Adaptive Modeling of the International Space Station Electrical Power System

DRAFT (5/4/2007)

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science Computer Science

By

Justin Ray Thomas

August 2007

Adaptive Modeling of the International Space Station Electrical Power System

Justin Ray Thomas

APPROVED:

Dr. Christoph Eick, Chairman

Dr. Carlos Ordonez

Dr. Glenn Webb
United Space Alliance

Dean, College of Natural Sciences and Mathematics

Acknowledgments

To Be Completed

Adaptive Modeling of the International Space Station Electrical Power System

---

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science Computer Science

By

Justin Ray Thomas

August 2007

Abstract

Software simulations provide NASA engineers the ability to experiment with spacecraft systems in a computer-imitated environment. Engineers currently develop software models that encapsulate spacecraft system behavior. These models can be inaccurate due to invalid assumptions, erroneous operation, or system evolution. Increasing accuracy requires manual calibration and domain-specific knowledge.

This thesis presents a method for automatically learning system models without any assumptions regarding system behavior. Data stream mining techniques are applied to learn models for critical portions of the International Space Station (ISS) Electrical Power System (EPS). We also explore a knowledge fusion approach that uses traditional engineered EPS models to supplement the learned models. We observed that these engineered EPS models provide useful background knowledge to reduce predictive error spikes when confronted with making predictions in situations that are quite different from the training scenarios used when learning the model.

Evaluations using ISS sensor data and existing EPS models demonstrate the success of the adaptive approach. Our experimental results show that adaptive modeling provides reductions in model error anywhere from 80% to 96% over these existing models. Final discussions include impending use of adaptive modeling technology for ISS mission operations and the need for adaptive modeling in future NASA lunar and Martian exploration.

Table of Contents

Figure 1. The International Space Station (ISS). *An artist rendering of the ISS after assembly completion (illustration courtesy NASA)*

1.0 Introduction

Simulations are the imitation of some real world entity. These imitations allow for experimentation and exploration that would otherwise be impossible due to costs, technical limitations, or risk to humans or materials. Military personnel can use fake laser weapons in live combat simulations without risk of injury. Flight simulators allow pilots to respond to malfunctions that are too risky to introduce during real flight. Crashes only damage the pilot's pride.

Computers can simulate environments and real world entities, which allows for interactions between objects within a virtual world. Video games such as SimCity™ [1] and The Sims™ [2] allow people to design and create a custom virtual environment. A user's decisions affect the virtual world as defined by the game designers. Using an example from SimCity, the user's popularity might plummet after the decision to raise taxes on the virtual townsfolk. The simulation evaluates the many different decisions a virtual mayor makes, all without the need to climb the political ladder in real-world society.

Computer simulations require a model of each simulated entity, including the environment itself. These software models are typically a mathematical definition that defines the behavior of a simulated entity. At the foundation, software models consist of a set of variables (state) and a relationship between states (transitions). A flight simulator must contain models for the Earth's gravitational force, the effect of atmospheric drag on solid bodies, and models for the aircraft's response to pilot controls. Modelers use physics to define the first two models, but the latter depends on the aircraft system engineering design.

Simulation plays a crucial role in NASA spaceflight. Our experience rests in the realm of human spaceflight, so this thesis focuses on simulation for human-rated spaceflight systems. Software models recreate hardware behavior when using the real system is impossible due to costs, safety, or operational constraints. System simulation is necessary for mission planning, training, systems engineering, and mission operations.

Unfortunately, human spaceflight modeling and simulation is an intricate process. Human-rated spacecraft contain highly complex systems. High fidelity models are expensive

[1] http://simcity.ea.com/
[2] http://thesims.ea.com/

to create and expensive to adjust to match true system behavior. Models are typically comprised of numerical methods based on the system's physical properties. Engineers tediously update these models by analyzing system data and selecting parameters that provide an optimal fit [1].

Adaptive system modeling and simulation techniques provide the solution. These techniques automatically adjust software models to reflect current system behavior. Machine learning (ML) and data mining (DM) techniques combine to form the answer. Instead of hard-coded static models, ML algorithms allow dynamic model creation using actual system data.

ML and DM techniques allow models to constantly adapt to match current system behavior. There is no need for tedious manual inspection and cumbersome tuning of model parameters. Adaptive models are highly accurate and dynamically adjust when assumptions about the system change. We defined these changes in system behavior as concept drift. Concept drift in hardware systems can be due to degraded performance, evolution over time, or unexpected changes in behavior.

Our primary contribution is a methodology for automatically learning and adapting system models over time without any assumptions regarding system behavior. Sensor data provides the real-time feedback source allowing for automated adaptation. The process includes the following major components:

- Feature selection to choose optimal sensor sets (i.e., parameters from spacecraft telemetry)
- Data preprocessing to prepare sensor data for training and evaluation

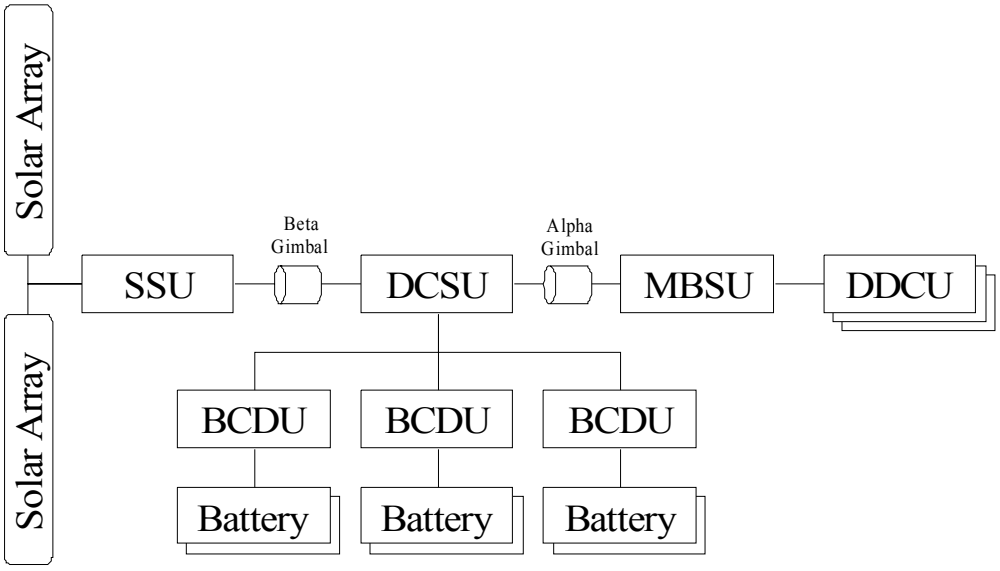

Figure 2. ISS Electrical Power System (EPS) Overview. *Solar arrays rotate to maximize solar intensity, as energy is stored in batteries for consumption when the ISS is on the dark side of the Earth.*

- Online clustering to build representative data sets across the system operating range
- Online predictor training to construct evolving system models that match current system behavior
- Model mixture that uses engineering models as background information for unexplored operating regions (i.e., areas in the input space)

We evaluate this technique using the International Space Station (ISS) Electrical Power System (EPS) [2] (Figure 2) and NASA historical data archives of spacecraft telemetry. Evaluation results demonstrate significant accuracy improvements over existing EPS engineering models. Engineering models are traditional static mathematical models created by system engineers using engineering specifications and physical properties.

Secondary contributions include:

- A knowledge fusion approach that uses traditional explicit mathematical models to supplement the learned models
- A technique for evaluating time-series forecasting models in the presence of frequent and/or long duration data loss

This thesis is organized into background information on modeling and simulation (Section 2.0), an overview of the technical approach (Section 3.0), the detailed approach and corresponding results for the battery model (Section 4.0), the detailed approach and corresponding results for the Battery Charge Discharge Unit (BCDU) model (Section 5.0), related work (Section 6.0), and finally, conclusions and future work (Section 7.0).

2.0 Background

The following section illustrates the need for autonomous adaptation of system models within the spaceflight domain. While spaceflight modeling possesses many challenging characteristics, the need for adaptive modeling exists in any domain requiring system simulation. The following sections provide the reader the necessary background information to understand the problem and rationale for our approach:

- Simulation in Modern Human Spaceflight
- Simulation Challenges
- Simulation in Future Human Spaceflight
- Introduction to the Electrical Power System

2.1 Simulation in Modern Human Spaceflight

Simulation plays an important role in NASA human spaceflight. Simulation is the cornerstone of spaceflight training for astronaut crews and flight control personnel. Simulation also allows for hardware and software testing when the use of real components is impossible or cost prohibitive. For Space Shuttle and Space Station mission operations, simulations of spacecraft systems assist engineers in pre-flight planning, in-flight monitoring, real-time mission planning, and post-flight analysis.

How long until the crew exhausts the oxygen supply during a cabin pressure leak? How can we orient the spacecraft to retain communication when mechanical antenna positioning fails? Is enough power available to run a science rack experiment for the required fourteen days? Simulations and the corresponding software models provide answers to all of these mission-critical questions.

Figure 3. NASA's Mission Control Center. *Simulations support flight controller decision-making during Space Shuttle and ISS operations. (photo courtesy NASA)*

The majority of these simulations currently operate in the Mission Control Center (MCC) (Figure 3). For example, NASA engineers currently use simulation to determine optimal vehicle maneuvers, analyze oxygen generation and consumption rates, generate power availability forecasts, and create contingency plans for system failures.

While analysts execute the majority of simulations on the ground, examples exist of simulations running on-board the spacecraft's computer systems. The Guidance, Navigation, and Control (GN&C) software on-board the Space Shuttle contains gravitational and atmospheric drag models to assist in launch and landing. On-board simulation is more prevalent in the autonomous systems found in vehicles such as Deep Space I and Earth Observing I. The flight software for such autonomous vehicles contains system models to simulate potential outcomes during decision-making.

2.2 Simulation Challenges

A critical requirement of spaceflight system models is that they reflect real-world system behavior. Inaccuracy hampers training and mission analysis capabilities. The analysts can deem the models unreliable in extreme cases.

Engineers find it difficult to model human-rated spaceflight systems for the following reasons:

- Human–rated spacecraft systems are complex
- True performance is only observable when the integrated system operates in space [1]
- Abnormal scenarios sometimes are not well understood
- System operation can evolve over time [1]

Exact behavior replication is difficult to achieve due to the sheer complexity of most human-rated spaceflight systems. A few examples of complex systems found in human-rated spacecraft are power [2], thermal, communications, environmental, and GN&C. These systems must operate and support human life in the most hazardous known environment. Physical system properties combined with reasonable assumptions drive system model development. Numerical methods and approximation techniques allow computer software to represent the necessary mathematical phenomenon.

In addition to complexity, a unique characteristic of spacecraft systems is that engineers often cannot perfectly predict true system performance until the system operates in space [1]. Engineers create models before they observe true performance during space operation. Even

when engineers reproduce comparable space thermal and gravitational environments on the ground, it is still difficult to test a fully integrated system. System emergent properties associated with a fully integrated system are hard to predict and understand.

System models must also reproduce both nominal operational scenarios and abnormal scenarios such as failures or performance degradations. Often, these scenarios are challenging to model and even more difficult to test due to the lack of supporting operational data. Even after the system behavior is well understood, spacecraft systems evolve due to age, design changes, and operational philosophies. Accuracy is a moving target requiring constant model evaluation.

Engineers create system models using hardware specifications, limited performance data, physical properties, and associated assumptions. Then engineers tune or calibrate the models based on true system performance [1]. This is a tedious process of analyzing historical system sensor data, evaluating the model, and determining the necessary software model adjustments. These adjustments often require specialized system knowledge and software model development expertise.

For these reasons, simulation model adjustments are manual, time-consuming, and expensive. Thus, engineers do not perform model adjustment on a continuous basis. Engineers make adjustments based on priority and budget availability. While this approach is sufficient for today's spaceflight missions, it is not optimal. A change in philosophy is required for future spaceflight back to the moon and then to Mars and beyond.

2.3 Simulation in Future Human Spaceflight

Future human spaceflight will increasingly depend on autonomous systems [3]. Many autonomous intelligent systems will rely heavily on system simulation for diagnostics and forecasting. Intelligent systems need system models and simulation environments in order to predict the outcome of potential decision paths. Simulation models must automatically adapt to fulfill future autonomy requirements. Manual adaptation would reduce the accuracy and efficiency of these autonomous systems due to infrequent manual updates. The sheer number of systems precludes such massive numbers of manual updates.

For future lunar exploration, autonomy is necessary to reduce the mission operations workload. Lunar excursions serve as preparation for future exploration or Mars. Scaling mission operations to support new expansive and complex NASA Constellation program[3] objectives requires minimization of human effort. Operation of the new NASA Orion Vehicle and lunar systems using manual techniques would overload mission support personnel, limiting mission size and complexity. While many lunar autonomous systems could operate on the ground, recent approaches are shifting towards on-board operation [4][5]. On-board operation benefits include crew autonomy during communications loss and real-time response for time critical tasks.

For Martian exploration, on-board autonomy is necessary due to communications limitations such as transmission latency and bandwidth. The lack of real-time communication for long-duration Mars flight exhibits the extreme case where astronaut crews must make timely decisions for mission-critical problems without any real-time ground support. The

[3] http://www.nasa.gov/mission_pages/constellation/main/index.html

face of ground support will change for these long-range missions. The speed of light makes continual real-time ground support communication infeasible. The flight crew must possess the analytical capabilities that currently exist in the MCC. To achieve this goal, future autonomous flight systems will require complex on-board simulation using highly accurate system models. Adaptive system modeling provides high accuracy while maintaining the flexibility to autonomously adjust to true system performance.

2.4 Introduction to the Electrical Power System

The ISS EPS [2] currently serves as the test bed for adaptive system modeling and simulation. The ISS absorbs sunlight through its massive solar arrays and converts the energy into a usable power source. An array of nickel-hydrogen batteries stores excess energy for use during orbital eclipse. The ISS orbit results in cyclic power charge/discharge behavior due to the periodic transition from eclipse to insolation (periods of solar radiation) as demonstrated in Figure 4.

We select this system due to a priori knowledge of the EPS and experience with an existing EPS simulation model. In-depth knowledge of the engineering model and the EPS facilitates research and development of the proposed adaptive system. The engineering model uses traditional mathematical modeling techniques based on engineering specifications and

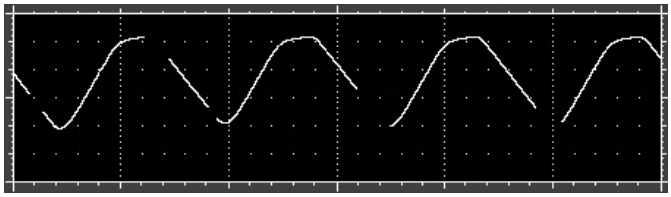

Figure 4. Battery Charge vs. Time. *The battery charges when sunlight is available and discharges as the Space Station requires energy reserves.*

physical properties. The engineering model generates forecasts of power system performance and available power for ISS flight controllers. The engineering model serves as the baseline for evaluating the adaptive system accuracy.

It is important to recognize a significant quality of the EPS. An EPS model used for mission support must predict future performance in a temporal manner. An EPS model must produce battery charge profiles as illustrated in Figure 4 given only initial conditions and a time-series of input data. Flight planners need to know if power will be available for the next week, not just the immediate future. We define this approach as *forecasting*. Forecasting adds complexity to the model. The model must also account for any behavior that can cause future system failure in addition to immediate failure.

3.0 Solution Overview

Our solution consists of up-front analysis components and on-line components that operate continuously to generate accurate system models without any assumptions or domain knowledge. The analysis phase drives the design of the on-line phase. The on-line phase consumes a data stream of sensor data (spacecraft telemetry) and generates adapted system models. The simulation system uses these adapted system models to generate the final predictions.

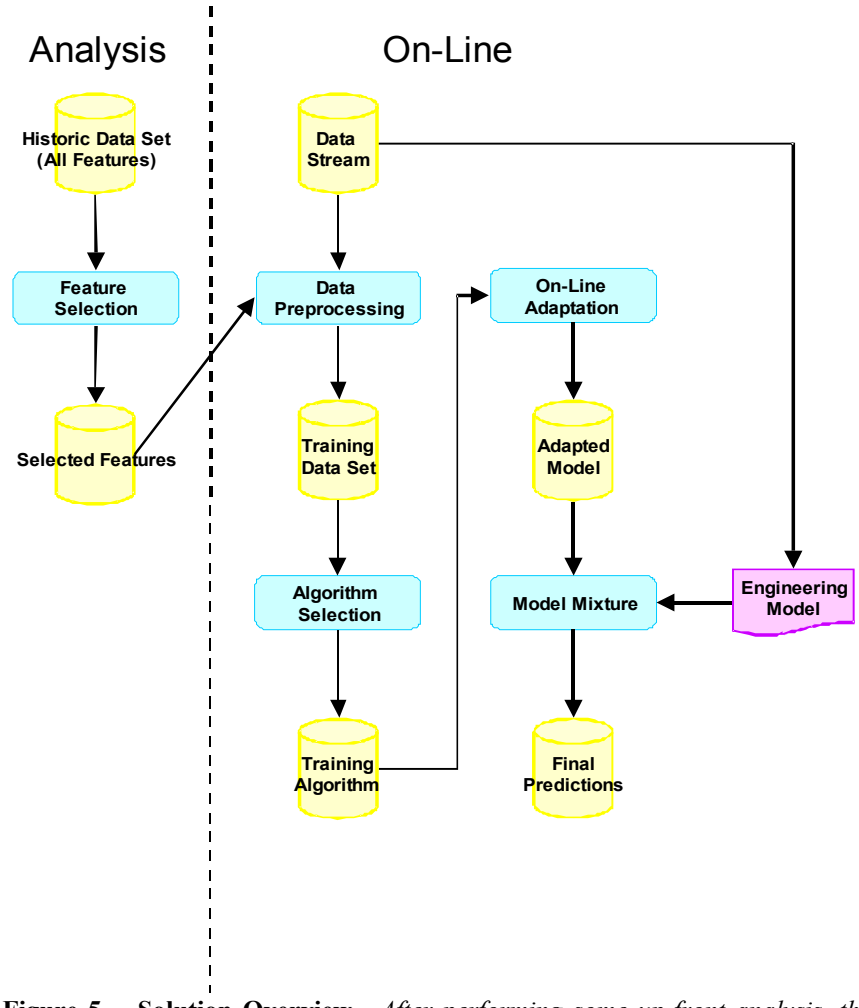

Figure 5. Solution Overview. *After performing some up-front analysis, the adaptive modeling system performs tasks in an on-line manner, consuming sensor data (spacecraft telemetery) in real-time.*

All components harness new and existing ideas from the fields of Machine learning and Data mining to solve our system modeling problem. The solution consists of the following primary components:

- Feature Selection
- Data Preprocessing
- Algorithm Selection
- Online Adaptation
- Model Mixture

Figure 5 illustrates each component in the operational data flow. We describe each component in more detail in the following sections. Feature selection occurs during initial analysis and adaptive model design. Data preprocessing, algorithm selection, adaptation, and model mixture occur during online system model operation.

Feature selection is the process of determining the set of sensors, or features, that generate the best possible model. Data preprocessing prepares sensor data for model training and evaluation. Algorithm selection chooses the ML algorithm that learns the best model. Online adaptation constantly learns system models using the most recently available sensor data and a representative data set of past system behavior. Model mixture fuses the knowledge contained in traditional engineering models to prevent error rate spikes. These spikes occur when the learned model must make predictions in situations that are quite different from the training scenarios used when learning the model.

3.1 Feature Selection

Feature selection is the process of determining the set of inputs, or features, that generate the best possible model. In our case, we wish to generate the model that will accurately predict real-valued numeric outputs. Numeric prediction contrasts with another approach in ML that classifies inputs into discrete-valued outputs, or classes. Feature set size reduction often results in increased model accuracy, reduced model complexity, and reduced training time. Fewer features also reduce dimensionality for data visualization. For example, we can easily visualize a data set containing only three numerical features using techniques such as 3-D scatter plots found in numerous existing software packages.

We identify candidate feature sets using the corresponding system's telemetry specification. These specifications provide the telemetry identifiers for all EPS sensors onboard the ISS. We ignore additional features not available to an isolated EPS model such as thermal system characteristics. Between the ISS and the MCC, software derives additional features from on-board sensors such as summaries and averages. We include these features in the initial candidate sets when available.

We employed the following techniques when performing feature selection:

- Domain Knowledge
- Correlation Analysis
- Subset Evaluation

Domain Knowledge – We selected the EPS due to prior knowledge of the system and prior experience creating an EPS software model. Manual selection of telemetry parameters

is possible using this knowledge. In addition, the engineering model identifies inputs and outputs for each system component. A valid starting point is to simply use the same features designated as inputs and outputs in the corresponding engineering model.

Correlation Analysis – One use of correlation analysis is to identify redundant parameters. Redundant parameters do not offer any additional insight into learning system operation since they always maintain the same numeric relationship. For example, EPS telemetry includes summarized values for low-level sensors. Since summary features are highly correlated with the low-level features, we can often ignore low-level features without sacrificing model accuracy.

The correlation between two features X_i and X_j is defined as:

$$Corr(X_i, X_j) = \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

Where σ_i and σ_j are the standard deviations of X_i and X_j respectively. The standard deviation is simply the square root of the variance, provided below.

$$\sigma_i^2 = \frac{\sum_{t=1}^{N} (x_i^t - m_i)^2}{N-1}$$

σ_{ij} is the covariance between the two features as we've defined in the following equation:

$$\sigma_{ij} = \frac{\sum_{t=1}^{N} (x_i^t - m_i)^2 (x_j^t - m_j)^2}{N-1}$$

In addition, for completeness, we define the mean of each feature using the following equation:

$$m_i = \frac{\sum_{t=1}^{N} x_i^t}{N}$$

Table 1. Correlation Matrix for EPS Battery Features. *Correlation values near +1 or –1 represent a perfect linear correlation between two battery features.*

	A	B	C	D	E	F	G	H	I	J	K	L
A	1	0.55	-0.43	0.67	-0.13	0.55	0.29	-0.37	-0.44	-0.44	-0.42	-0.42
B	0.55	1	-0.92	0.81	-0.1	0.99	0.2	-0.9	-0.93	-0.93	-0.91	-0.92
C	-0.43	-0.92	1	-0.62	0.1	-0.91	-0.16	0.91	**1**	**1**	**1**	**1**
D	0.67	0.81	-0.62	1	-0.18	0.81	0.44	-0.52	-0.63	-0.64	-0.59	-0.6
E	-0.13	-0.1	0.1	-0.18	1	-0.1	-0.26	0.02	0.1	0.1	0.09	0.1
F	0.55	0.99	-0.91	0.81	-0.1	1	0.2	-0.9	-0.92	-0.92	-0.9	-0.91
G	0.29	0.2	-0.16	0.44	-0.26	0.2	1	-0.08	-0.16	-0.17	-0.15	-0.16
H	-0.37	-0.9	0.91	-0.52	0.02	-0.9	-0.08	1	0.9	0.91	0.91	0.91
I	-0.44	-0.93	**1**	-0.63	0.1	-0.92	-0.16	0.9	1	**1**	**1**	**1**
J	-0.44	-0.93	**1**	-0.64	0.1	-0.92	-0.17	0.91	**1**	1	**1**	**1**
K	-0.42	-0.91	**1**	-0.59	0.09	-0.9	-0.15	0.91	**1**	**1**	1	**1**
L	-0.42	-0.92	**1**	-0.6	0.1	-0.91	-0.16	0.91	**1**	**1**	**1**	1

Table 1 contains a subset of the correlation matrix for the candidate EPS battery features. The diagonal always contains values of one since each feature perfectly correlates with itself. Correlation values approaching positive or negative one represent a high linear correlation. In this example data set, features C, I, J, K, and L perfectly correlate with each other. We expected this result since feature C is simply the average of features I, J, K, and L. In this case, the summary feature is often sufficient for learning tasks. We can ignore these low level features, as they will only increase training time and model complexity. B and F are also highly correlated with a value of 0.99. This suggests that possibly only one of the two features is necessary for learning.

The limitation of this approach is that we only identify linearly correlated features. We cannot identify exponential or logarithmic relationships using this technique defined above. The correlation ratio [24] is more appropriate to identify such nonlinear correlation. We did not explore the correlation ratio since we discovered the expected feature set based on domain knowledge.

Subset Evaluation – Subset evaluation analyzes combinations of features from the candidate set to identify those features likely to produce the most accurate predictor. The WEKA software toolset [9] contains the Classifier Subset Evaluator algorithm. This evaluator constructs feature subsets, builds a classifier/predictor, and then measures the accuracy on an independent test set.

For example, if we have the candidate feature set {1, 2, 3} then this technique will construct sets {1, 2, 3}, {1, 2}, {1, 3}, {2, 3}. {1}, {2}, {3}, and {∅}. Then Classifier Subset Evaluator learns a model for each feature subset and identifies the subset used to construct the most accurate model.

Unfortunately, Classifier Subset Evaluator requires a priori selection of a ML algorithm when selecting features. The ML community defines this approach as a *wrapper* feature selection technique [23]. Classifier Subset Evaluator iteratively extracts a subset of the candidate features based on a search heuristic, builds a model using a user specified ML algorithm, and then measures the accuracy on an independent test set. Classifier Subset Evaluator then chooses the feature subset that generates the most accurate model. Since ML algorithm parameters can affect model performance, we must search many combinations of algorithms and algorithm parameters to find the best possible feature set. Instead of using

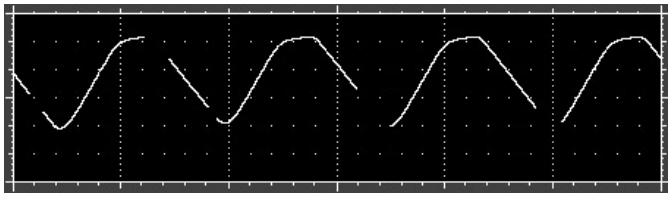

Figure 6. Battery Charge Level with Missing Values. *The gaps in the plot line represent areas where telemetry is missing due to shared resource limitations with the orbital satellite communications system.*

intrinsic properties of the data set, we introduce bias by predetermining algorithm and parameter choices. Since we have predetermined our candidate ML algorithms, the wrapper feature selection approach suffices.

3.2 Data Preprocessing

Data preprocessing prepares sensor data for model training and evaluation. We construct training and evaluation data sets using historical ISS telemetry data. The ISS on-board computers sample most EPS sensors in the feature set every 10 seconds (0.1 HZ rate). This rate provides ample training data.

ISS telemetry does not contain data during communication loss, expected or unexpected. The gaps in the charge level plot (Figure 6) represent communication loss. The communications satellite system is a shared resource and we expect communication outages due to priority-based sharing. These extended durations of missing values complicate the learning and evaluation tasks. In the extreme case, data can be unavailable for up to 30 minutes. Such large durations of missing data complicate the tasks of learning system models. The following sections describe the approach to address missing data.

3.2.1 Training Set Construction

We construct training sets by simply removing all data points with missing data from the full data set. The occurrence of missing data is completely independent of EPS system operation. Our approach of discarding missing data does not negatively impact model accuracy since ample data is available for training. At a 0.1 HZ rate, there is a maximum of 8460 data samples for a single 24-hour period.

The preprocessing for generic training sets is straightforward. We must convert the historical telemetry data into a format understood by WEKA. Then we simply delete missing data from the data set. When creating a forecasting model, the preprocessing task becomes more complex. We describe the approach for the forecasting battery model in more detail in Section 4.2.1.

We manually performed the training set construction process for early evaluation and later automated the processing when performing evaluations over extended timeframes.

3.2.2 Evaluation Set Construction

Standard Model Evaluations – We construct the standard evaluation set in the exact same manner as the training set. Traditional evaluation techniques such as n-fold cross validation suit our needs.

Forecasting Model Evaluations – To evaluate forecasting model accuracy, we could use standard evaluation techniques such as n-fold cross validation with data sets constructed similar to the training data sets. Unfortunately, this simple approach does not provide a realistic evaluation of time-series forecasting. The EPS model must forecast behavior over time. The EPS model uses a time-series of inputs and initial system conditions and then

generates the corresponding time-varying output. We cannot evaluate each data point independently. Data points are dependent on the previous data points. We refer to this dependency as a time-series sequence of data points. We must evaluate our approach on the entire time-series data sequence.

The primary problem is finding a lengthy ISS data set without any missing data. Missing data forces us to use assumptions when completing missing values. These assumptions can possibly bias the evaluation.

We desire a data set without any significant gaps for at least 90 minutes for realistic evaluation. This minimum duration represents one ISS orbit around the Earth. Data evaluation less than one Earth orbit do not fully demonstrate model performance. The optimal case is to use an evaluation data set for the same forecasting duration expected during operations. NASA engineers forecast EPS operation for several weeks into the future. Unfortunately, a perfect 2-week data set for forecasting evaluation does not exist. Additionally, high-quality data sets are only available for up to approximately four hours. Detailed searches through historical data archives can discover longer duration, high quality evaluation sets, but the nominal sets are sufficient for evaluating our technical approach.

Time	Value
1	1.25
2	?
3	?
4	2.00

Linear Interpolation →

Time	Value
1	1.25
2	1.50
3	1.75
4	2.00

Figure 7. Filling Missing Values with Linear Interpolation. We use *simple linear interpolation (blue) to update two missing values (yellow).*

We complete periods of missing data using linear interpolation (Figure 7). Visual inspection methods initially verified acceptable data quality when completing from 1 to 10 consecutive missing values. For longer durations, the resulting data set contains unrealistic data due to the approximation effects of linear interpolation. Missing data can result in erroneous input data (Figure 8) that biases the evaluation. We desire a completely unbiased evaluation to properly compare techniques, but this is not feasible for our problem.

To provide complete evaluation, we must evaluate the accuracy of competing techniques over historical ISS EPS operation. Such a full evaluation requires automated evaluation set construction. We desire quality evaluation sets for at least every 24-hour period over the ISS EPS lifespan (approximately 6 years). To automatically determine these data sets, we construct simple histograms based on the number of acceptable consecutive missing values,

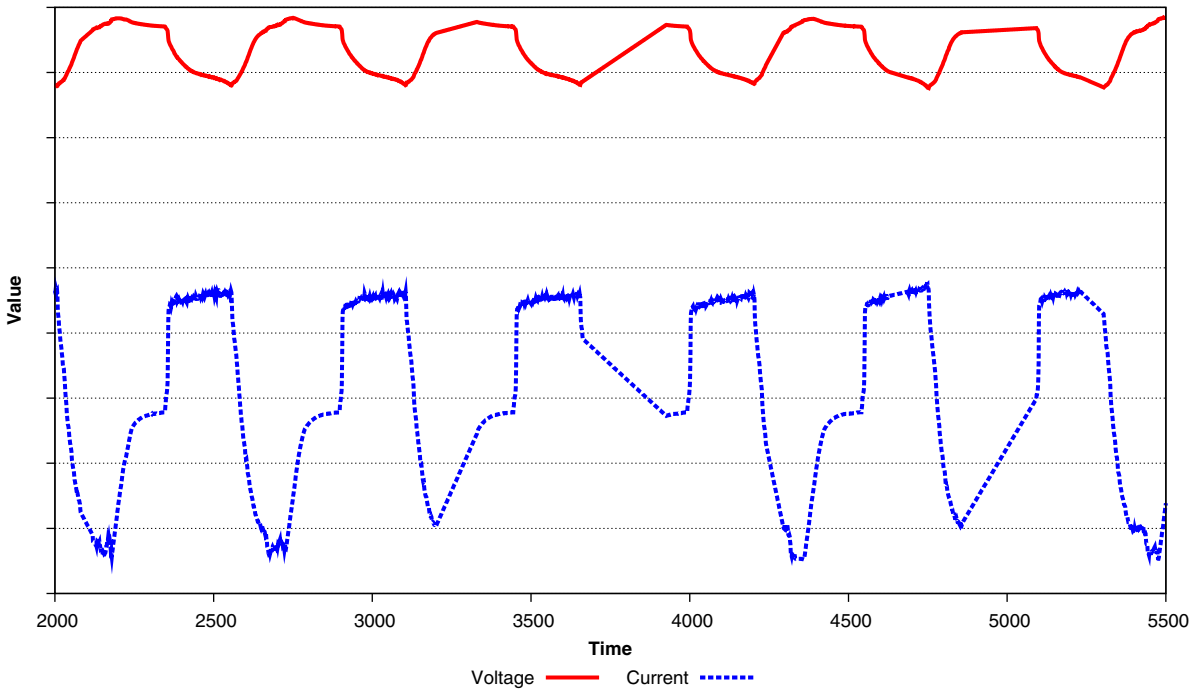

Figure 8. Input Error using Linear Interpolation. *Linear interpolation erroneously breaks the cyclic pattern in both inputs between time points 3600 and 3900 and then again between time points 4800 and 5100.*

Original Data

Time	Value
1	1.25
2	?
3	?
4	2.00
5	?
6	5.50
7	?
8	?
9	?
10	4.25

$\mu = 1$

Time	Value
4	2.00
5	4.75
6	5.50

$\mu = 2$

Time	Value
1	1.25
2	1.50
3	1.75
4	2.00
5	4.75
6	5.50

$\mu = 3$

Time	Value
1	1.25
2	1.50
3	1.75
4	2.00
5	4.75
6	5.50
7	5.25
8	5.00
9	4.75
10	4.50

Figure 9. Evaluation Data Set Construction with Missing Values. *We construct longer duration evaluation sets when the system is willing to accept more consecutive missing values (μ). When increasing μ from2 to 3, the evaluation set size is increased from 6 to 10, with some additional data quality sacrifice.*

μ, and the corresponding data set lengths.

Figure 9 illustrates the concept of filling missing values in time-series data sets using the number of acceptable consecutive missing values, μ. The original data set contains gaps of one, two, and three data points. We can only use three data points from the original set when we accept one consecutive missing data point ($\mu = 1$). When setting $\mu = 2$, the resulting data set length is five. Finally, when setting $\mu = 3$, the resulting data set length is 10. This is the same length as the original data set. We must explore the balance between μ and the resulting data set lengths in order to create lengthy evaluation data sets with minimal bias.

For model evaluation, we must analyze a historical data set to find all lengthy data sets with minimum need for linear interpolation. For example, in Table 2, without accepting any missing values, the length of the longest data set is 30 minutes, but when setting $\mu = 1$, the maximum data set length is now 1 hour. Linear interpolation with only one missing value is sufficient in our domain where data points are sampled every 10 seconds.

When constructing evaluation sets, our system automatically analyzes a historical data set. Our system presents the analyst with a histogram similar to Table 2. The analyst then determines how many data sets are desired given the μ value. So in Table 2, the user can select $\mu = 2$ to receive nine evaluation data sets with at most two consecutive interpolated data points. However, the user can select $\mu = 3$ to receive a much longer duration (147 minute) data set. Once the user chooses μ, the system automatically creates the resulting interpolated evaluation sets.

The μ parameter is available to weigh evaluation results. Results from a data set with minimal missing data are the best indicator of true system performance. Our system currently does not use the μ parameter for automated analysis. We noticed no strong correlation between μ and the resulting error rates for the adaptive and engineering models, and suspect

Table 2. Automated Evaluation Set Construction using Histograms. *When accepting one consecutive missing data point, the maximum evaluation data set length drastically increases from 30 to 60 minutes.*

μ	Resulting Data Set Lengths
0	30 minutes, 22 minutes
1	60 minutes, 54 minutes, 54 minutes, 32 minutes, (3 more)
2	66 minutes, 59 minutes, 54 minutes, 38 minutes, (5 more)
3	147 minutes, 66 minutes, 59 minutes, 54 minutes, (6 more)
4	147 minutes, 66 minutes, 59 minutes, 54 minutes, (6 more)

the reason is that we used such small μ values.

Quantitative Technique Evaluation – We analyzed our technique for filling missing data for forecasting evaluations. Our evaluation supports the two following conclusions:

- Linear interpolation successfully completes missing EPS sensor values with minimal error.
- The completion of up to ten consecutive missing values effectively creates high quality data sets that will not overly bias forecasting model evaluation.

First, we want to evaluate linear interpolation for completing missing data. We chose two additional techniques to complete missing data:

- Last Valid Value (Figure 10) - Using the sensor value before the sequence of missing values to complete all missing values

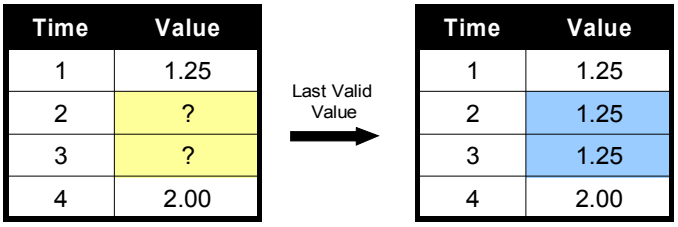

Figure 10. Completing Missing Values with the Last Valid Value. *Our system uses the sensor value before the missing data sequence (1.25) to complete the missing values.*

- Mean Value (Figure 11) - Using the sensor mean value as a constant to complete all missing values

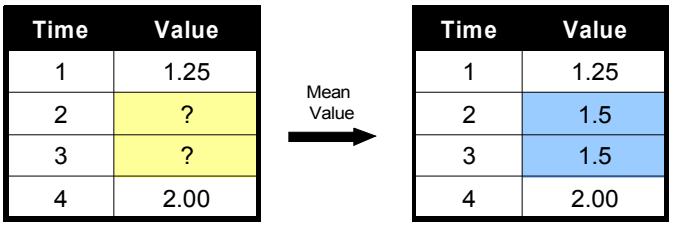

Figure 11. Completing Missing Values with the Mean Value. *Our system uses the sensor mean value (calculated externally as 1.5) to complete the missing value.*

To perform the evaluation, we performed the following procedure using 30 days worth of highly cyclic EPS sensor data:

1. Randomly sample 1000 data sequences of length μ + 2 that do not contain any missing values; the two additional data points are necessary to perform linear interpolation.
2. Delete all values between the beginning and the end values of the sequence.
3. Apply each of the three missing value completion techniques (linear interpolation, last valid value, mean value) to the sequence.
4. Calculate the sum of absolute error – defined as the difference between the true and the predicted values – for each sequence and average them across all 1000 sampled sequences.
5. Repeat the procedure 100 times with μ varying from 1 to 100.

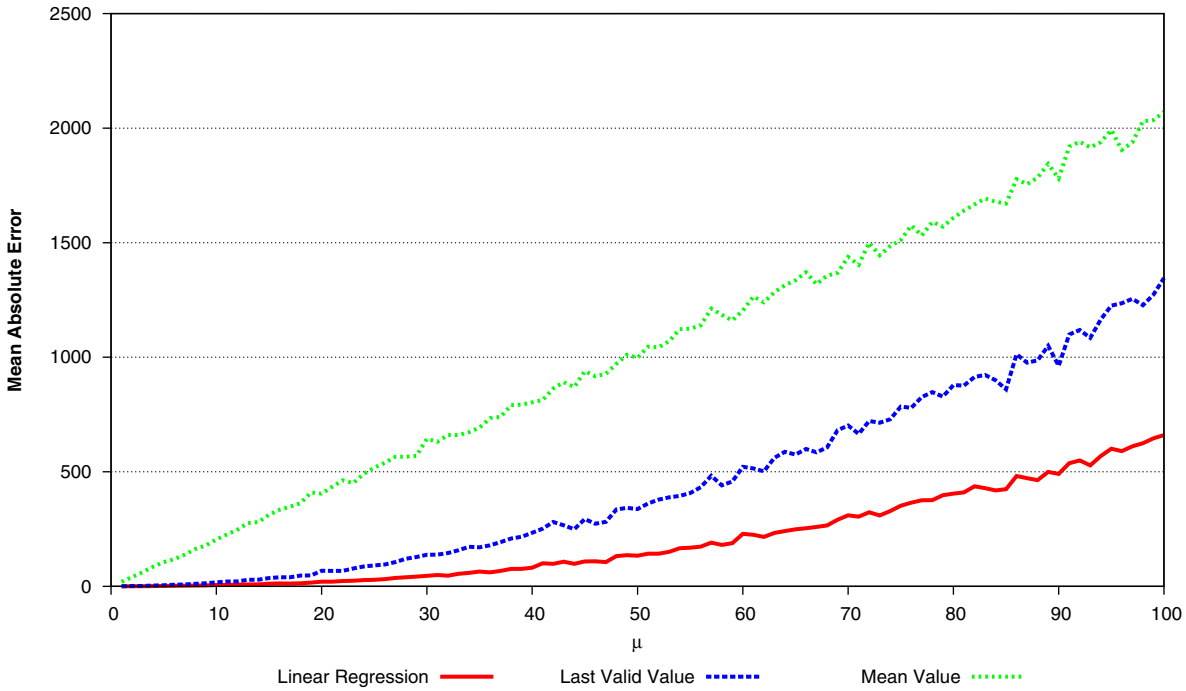

Figure 12. Forecasting Evaluation Set Construction Comparison. *Linear interpolation always outperforms the last valid value and mean value techniques. Error quickly increases with the number of consecutive missing values.*

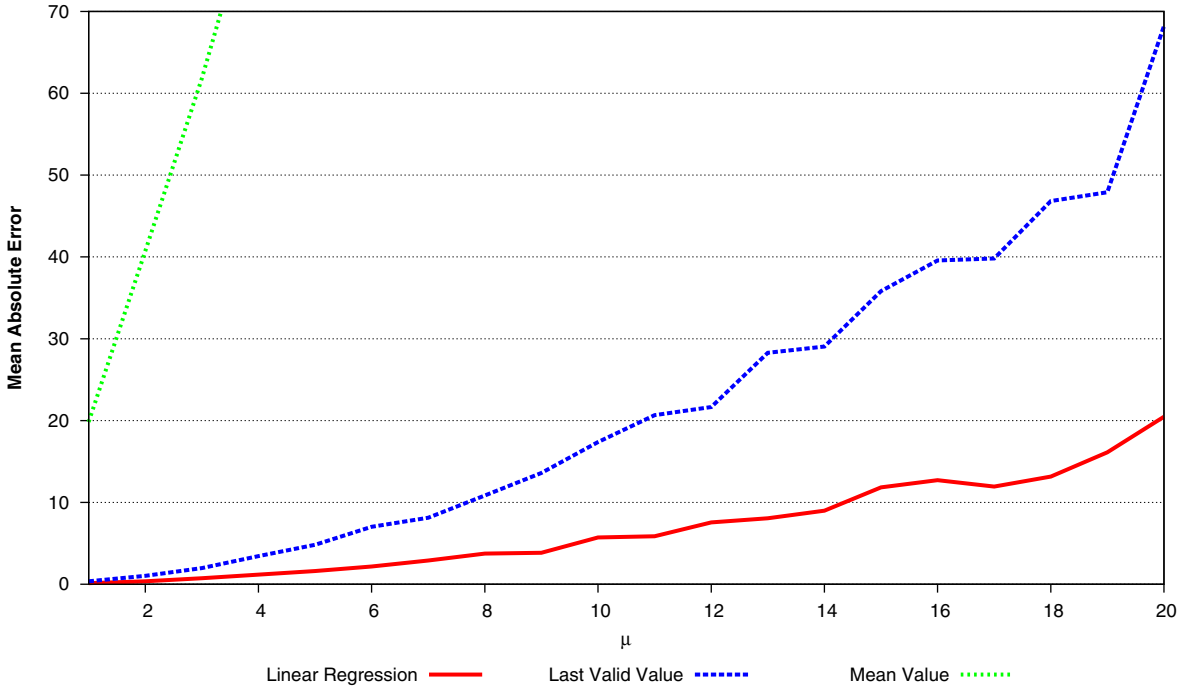

Figure 13. Forecasting Evaluation Set Construction Comparison (Zoomed). *This plot zooms in on the origin of Figure 12. Linear interpolation outperforms the last valid value technique for all values of μ and does not cross our threshold of five until μ=10.*

Figure 12 shows the mean absolute error plotted against μ. The purpose of this plot is to display the error growth trends as μ reaches unreasonable values. Figure 13 zooms into the origin of the plot in Figure 12 using μ from 1 to 20. The purpose of this plot is to compare each technique within ranges of plausible error. We removed the mean value technique results from Figure 13 to appropriately scale the plot so that small details are discernable.

Linear interpolation is the most accurate of these three straightforward methods for all μ values. We expected this result due to the cyclic nature of the data. A constant value will always introduce the highest amount of error since there is high variation in the true value. The error rates for the linear interpolation technique and the last known value technique were slightly exponential. The error rate for the mean value technique was linear, but substantially worse than the other two techniques for μ less than 100.

We found the amount of error introduced by sequences where μ was greater than nine too severe. The use of these sequences breaks our threshold of acceptable error for meaningful evaluation. For the experiments in this thesis, the actual value chosen for μ was four (Section 4.2.2), which introduces minimal error.

Other potential solutions include more complex regressive forecasting models (e.g. Auto-Regressive Moving Average (ARMA) [20]) that predict time-series patterns. Linear regression is suitable for our experiments as it provides reasonable and simple estimates without overly biasing the data.

3.3 Algorithm Selection

Using data sets that contain actual inputs and outputs, ML algorithms train a representative model of the device. Engineers have no need to write any mathematical equations to model physical system properties. The ML algorithm learns the necessary function to create an accurate model. The effective selection of a ML algorithm is imperative to achieve optimal accuracy.

A few popular ML algorithms capable of numeric prediction include:

- Artificial Neural Networks (ANNs)
- Regression Trees (RT)
- Support Vector Machines (SVMs)
- Linear Regression (LR)
- k Nearest Neighbors (kNN)

In addition to these algorithms, we explored ensemble approaches that increase the power of ML algorithms by creating multiple models that work together during prediction.

Each of these algorithms tends to perform optimally under specific conditions. However, ANNs provide a promising general solution for adaptive system modeling. For this reason, a detailed discussion of ANNs follows.

3.3.1 Artificial Neural Networks

ANNs model the scientific understanding of the biological neurons found in the brain. Practical applications prove ANN success in many domains (control systems, handwriting

recognition, image classification), including aerospace. Researchers have used ANNs to model the Space Shuttle Main Engines [6].

ANNs are an excellent candidate for this particular solution. ANNs exhibit the following characteristics desired for spaceflight system simulation:

- Forecasting capability for system performance prediction [7]
- Modeling of continuous and arbitrary numeric functions that are the essence of a system model
- Support for a large number of inputs useful for complex system models
- Immunity to the noise typically found in sensor data
- Online training techniques for real-time adaptation
- Efficient prediction performance during model use

ANNs can train in batch mode or incrementally using a variation of the stochastic backpropagation training technique [8]. Backpropagation is a supervised algorithm used to train ANNs using known input/output combinations.

3.3.2 Ensemble Methods

Combinations of multiple ML models result in a powerful predictor that typically outperforms each individual learner. The ML community refers to the approach of combining learners as ensemble methods. Ensemble methods combine multiple models built using base learning methods, such as those described in Section 3.3.3. The final model is an aggregate of these multiple models.

The ensemble methods examined in this experiment are:

- Boosting
- Bagging

Boosting – Boosting can take a weak learner (a learner with no worse than 50% error) and turn it into a powerful, accurate learner. Boosting iteratively builds multiple models using the same ML algorithm. After each iteration, boosting assigns more significance to incorrectly classified data points when training the next model. Each successive model focuses on the previous model's mistakes. The boosting process stops building models after it reaches a maximum defined model count or when predictor error is greater than 50%.

Boosting typically increases the significance of a data point by adjusting a weight factor for each data point. In this case, boosting requires a ML training algorithm that supports weighted instances. Alternatively, an algorithm can increase the probability of choosing the data point while creating a training set using random sampling with replacement. The popular AdaBoost [22] algorithm uses the latter technique.

To generate predictions, each generated model participates in voting. Boosting weighs each model's output based on its performance on either the training set or an independent validation set.

Bagging – Bagging combines models trained using the same algorithm while introducing variation in the training set. Bagging generates training sets in an iterative fashion using random sampling with replacement. This means that a single data point can appear more than once in the base algorithm's training set.

Bagging offers the best performance when using an unstable algorithm. An unstable learner can generate a significantly different model with slightly different training set. Most decision trees are unstable learners. Numeric predictions are a simple average of the output from all generated models. Unlike boosting, Bagging weighs all models equally. For numeric prediction, bagging simply averages the output of all values.

We did not evaluate meta-learners such as stacking and cascading in this experiment because we need an approach that uses a default model when learned models are not confident. The model mixture approach described in this thesis is a type of meta-learner. We will discuss this idea further in Section 3.5.4.

3.3.3 Online Algorithm Selection

Our system evaluates a set of candidate algorithms at a fixed frequency rather than choosing a fixed algorithm prior to system operation. The optimal algorithm can change when the system faces concept drift. During concept drift, the target function that replicates current system behavior changes. In this case, a different ML algorithm may better approximate the target function. Online algorithm will determine which algorithm provides the most accurate model on a periodic basis.

Online algorithm selection simply performs 10-fold cross validation on the training data set using a set of candidate algorithms. The system selects the algorithm that generates system models with the lowest average absolute error. For the purposes of our experiments, online algorithm selection occurs every 24 hours.

The system generates models using the best possible learner from a list of highly suitable algorithms. The algorithms are most applicable to the problem of creating a global model for system hardware. In many cases, the algorithms are complimentary, increasing the chances that at least one learner will perform well. We used the following list of candidate algorithms in our experiments:

- Multi-Layer ANNs
- Boosted Multi-Layer ANNs
- Linear Regression
- Reduced Error Pruning (REP) Regression Trees [9]
- Bagged REP Regression Trees

Table 3 lists our default parameter selections. In some cases, we will override specific parameters to increase performance on a specifically modeled system. Even without parameter optimization, we expect this list to be generically applicable to most spacecraft system models.

Unfortunately, the system does not use the k Nearest Neighbors predictor due to the associated run-time performance. The costly lookup of nearest neighbors causes simulation time to increase drastically, which our simulation users would find unacceptable. We did not use SVMs due to the amount of detailed parameter tuning involved and scalability limitations as the training set size increases. We provide additional details in Section 4.3.1.

Table 3. Default Algorithm Parameters. *Unless otherwise specified, we used the following values for each training algorithm.*

Algorithm	Default Parameters
Multi-Layer ANNs	• Single hidden layer (nodes = number of attributes / 2) • Learning Rate = 0.3 (constant during training) • Momentum = 0.2 • Attribute Normalization • Maximum Training Epochs = 500 • Validation Set Size = 10% training data • Validation Threshold = 20 (# of iterations of decreasing performance before halting)
Boosted Multi-Layer ANNs	• Default ANN Parameters (above) • Shrinkage Rate = 0.9 • Maximum Model Count = 10
Reduced Error Pruning (REP) Regression Trees	• Folds for Pruning = 5 • Minimum Total Leaf Weight = 2.0 • Minimum Variance Proportion = 0.001
Bagged REP Regression Trees	• Default REP Regression Tree Parameters (above) • Maximum Depth = 50 • Maximum Model Count = 10
Linear Regression	• None

3.4 Online Adaptation

Online adaptation is the autonomous and continuous learning of system models while the system is operating. The training process updates the model to more accurately mirror current system behavior.

There are two major components in our online adaptation approach:

- Online Clustering
- Online Training

In this proposed approach, the system creates the initial model using any available test data. Using online adaptation, the system will evolve the model based on environmental feedback. Over time, the adaptation process compensates for limited test data.

To demonstrate the need for online adaptation, we built a predictor using data after battery installation (November 2001). We then evaluated this predictor against the latest 2006 data. The learned model did not perform well. The engineering model outperformed the learned model by as much as 400%. This is not solely due to system evolution. After examining the flight data, we discovered that the battery was simply operating in a different region. We did not train the predictor on data in this unexplored region, which was the root cause of the observed inaccurate predictions. Online adaptation would create a model that handles these new operational regions in addition to past operating regions.

3.4.1 Online Clustering

We must use an incremental approach since it is simply not realistic to use the entire data stream for model training. At 10 HZ sensor sampling, one year's worth of operational

data results in a training set with over 3 million data points. Our approach uses clustering on a periodic basis to solve this problem.

Traditional online approaches either use only the most recent data point or maintain a sliding window of the most recent data points for some range of time *t*. The problem with these approaches for system modeling is that models forget unused operational regions over time. Once the system returns to a different operating region, the predictor will most likely provide unreliable output. Incremental approaches will relearn the operational region, but can take hours or days based on the sampling frequency and learning rate.

Rather than retaining the most recent data, we desire to retain a representative data set of all exercised operational regions. The predictor can use this representative data set to build a predictor for the full operating range and adapt to the latest data in an online fashion.

Clustering allows the creation of a representative data set. By selecting a large number of clusters, the clustering procedure replaces the numerous original data points by a representative data point for that operating region. This effectively creates a constant size training set that retains data points across the entire distribution of operational data. Then, re-clustering using a global clustering technique evolves the data distribution by slowly migrating to the most recent data. The clustering procedure retains the last known data for recently unused operating regions as the best last known estimate.

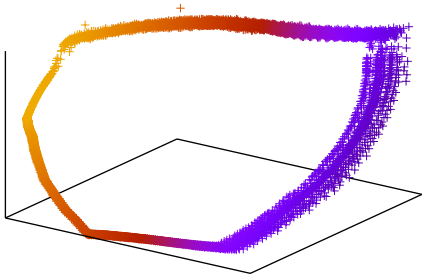

Figure 14. Full Data Set Before Clustering. *The scatter plot displays the tightly grouped data points in the operating range.*

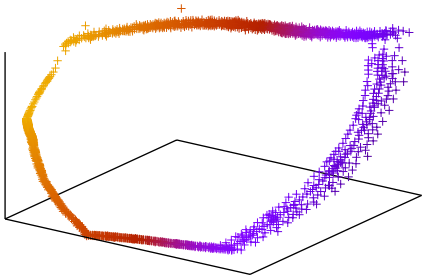

Figure 15. Reduced Data Set After Clustering. *Clustering preserves the data distribution across the operating while reducing the overall number of data points by 75%.*

Figure 14 illustrates 24 hours of battery data (voltage vs. current with a color coded output) as received from the ISS. Figure 15 shows the 2000 point representative data set we built using the k-means clustering algorithm. We preserve the data distribution while drastically reducing data set size. The representative data set size must remain constant to bound model training time.

3.4.2 Online Training

Performing traditional predictor training using the representative data set at a fixed frequency achieves online behavior. We can adjust this frequency. The worst-case complexity for clustering and predictor training limits the maximum frequency. Since the training set is a constant size, we can easily provide an estimate for worst-case execution time. In other words, if we receive data points every 10 seconds, but the adaptation process consumes 12 seconds, the adaptive system is unable to train before receiving the next data point. The system would lag behind the inputs, although this is not of serious consequence.

We would only waste CPU time if we trained multiple independent models using the same data sets. We only need to train a single model per physical device. All adaptive simulation systems can share this single model. We will dedicate computational resources to the sole task of model adaptation. Grid or agent architectures would allow distributed model sharing. There is no need for the system to adapt multiple system models for the same device.

The clustering and static training approach performs online adaptation for the EPS models. The steps are as follows:

- Initialize with n sequential data points
- Determine k clusters using k-means
- Build a new predictor (ex. ANN) using the cluster centers
- Repeat the clustering, adding the m new data points to the set of k cluster centers

We recommend $m < k$ to prevent new data from over-saturating the pre-clustered data set. Otherwise the new data points will have too great of an influence on the representative data set.

In most circumstances, it is possible for the system to perform clustering and build a new classifier for each incoming data point. However, such frequent model updates are not necessary in most cases.

We designed this adaptive approach to:

- Maintain a representative training set that captures the data distribution across the operating range

- Reduce the training set to a manageable size for online predictor training
- Retain outliers that represent abnormal operation by reusing historic clusters
- Account for concept drift by shifting cluster means based on the most recent data
- Smooth the training data set using k-means (averaging numeric values) rather than PAM [21]

3.4.3 Discussion

We chose the above adaptation strategy from a set of candidate approaches. The other approaches we considered were:

- Open-ended incremental predictor training
- Online clustering with static predictor training

Open-ended incremental predictor training – In this approach, there is no clustering or data set reduction. The predictor trains incrementally only on new incoming data. As previously mentioned, the predictor can forget output for recently unused operating regions. While this approach trains immediately, the learner requires many new data points to accurately predict new or forgotten operating regions. In addition, it would be difficult for a system to mix the engineering model with the adaptive model using this strategy. There must be a strategy for estimating the accuracy of the learned model on a new data point. However, this strategy is not straightforward. With the previous approach, the distance from the training set is available. To use this approach, the systems must maintain some sort of data set, which already occurs in our current online adaptation strategy.

Online clustering with batch predictor training – Instead of using a batch cluster, online clustering techniques are available to update clusters for each incoming data point. Online clustering offers a potential performance improvement if the batch clustering in the pseudo-incremental technique is inefficient. The faster clustering can take place, the quicker the model can adapt to the latest sensor data. We did not explore this approach due to the minimal computational requirements of our selected approach.

3.5 Model Mixture

Model mixture is a knowledge fusion approach that uses traditional engineering models to supplement the learned models. We observed that engineering models provide useful background knowledge to reduce predictive error spikes when confronted with making predictions in situations that are quite different from the training scenarios used when learning the model. We want to limit the maximum error. Also, we wish to protect against the chance of the system generating an unstable model.

This section explores the following topics to introduce our novel approach:

- Motivation
- Approach
- Online Threshold Selection
- Discussion

3.5.1 Motivation

Model mixture is the use of engineering models with the new adaptive models to provide accurate output when facing newly explored operating regions. Even though ML algorithms

can learn behavior quickly and efficiently, the simulation cannot always use adaptive models in place of the engineering model. There are two primary reasons:

- Abnormal scenario prediction (failures and unexpected behaviors)
- Safety

The following discussion outlines the need for traditional engineering models with new adaptive models to create a single synthesized simulation. After describing the need, we present the model mixture technical approach.

Abnormal scenario prediction – An adaptive model will quickly learn nominal behavior, but simply cannot always accurately predict scenarios that have yet to occur. These abnormal scenarios may not have applicable test data since this data is usually difficult or expensive to create. In this case, the adaptive model risks having insufficient supporting operational data for the learning task.

The adaptive model's inaccurate output would initially hamper the effectiveness of any simulation system using the system model. A decision support system might provide invalid analysis to the flight crew. A spacecraft control system can take the incorrect action, potentially producing catastrophic results.

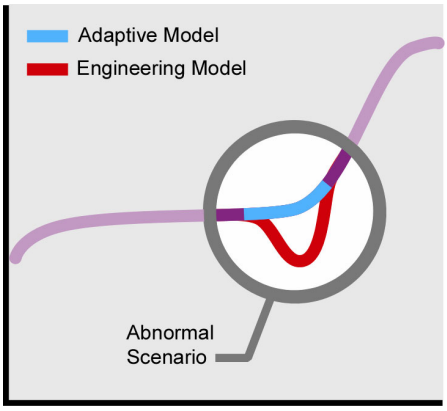

Figure 16. Abnormal Scenario Prediction. *Static engineering models are necessary to initially predict abnormal scenarios the adaptive model has yet to learn.*

We must initially rely on engineered mathematical models to generate the best possible output estimate. Traditional system models provide background

knowledge to the adaptive model. This information represents an understanding of the expected behavior in nominal and abnormal situations. We illustrate an example comparison of both models in Figure 16. This comparison shows that an adaptive model can incorrectly assume a smooth function output if no training data is available for that operational region. The engineering model can contain the equations that account for abnormal scenarios. As sufficient operational data arrives, the engineering model becomes unnecessary as the adaptive model learns to correctly predict the abnormal scenario.

Safety – Model mixture provides a safety net to ensure that adaptive model extrapolation does not introduce bizarre results. Unusual output will hurt the user's confidence and decrease their trust in this intelligent system. Model mixture also allows for side-by-side comparison of adaptive and engineering models.

Engineers find it difficult to solely trust a learned model of a complex system. A cultural transition is necessary to embrace advanced ML techniques for spaceflight. Since the adaptive model is very efficient, the existing engineering models and the new adaptive models can run simultaneously. The independent execution of engineering models and adaptive models allows for real-time comparison of both approaches. The user is able to manually select which results to use. The user will become more comfortable with the new technology as the adaptive model demonstrates increased accuracy over existing engineering models. Confidence in the model will increase when the adaptive model is consistently more accurate than the engineering model and always within reasonable error bounds.

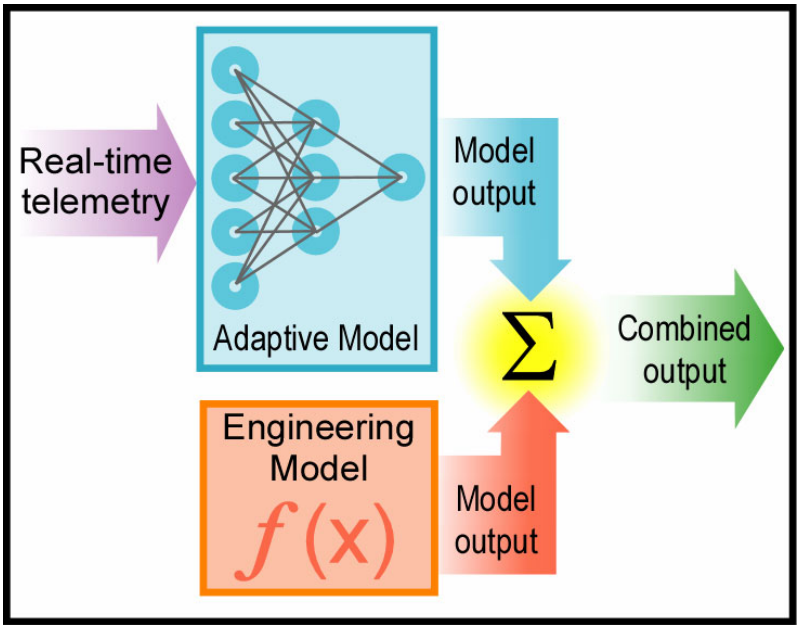

Figure 17. Simulation Output Mixture. *The adaptive system selects from the adaptive model output or engineering model based on the estimated amount of extrapolation error from the adaptive model.*

3.5.2 Approach

The overall EPS model switches output between the engineering model and the adaptive model (Figure 17). We base the switching criterion on the distance of the new data point to the nearest data point in the current training set. Our system needs a threshold distance to make the switching decision in this hard boundary mixture.

We chose Euclidean distance for all experiments, although the system allows any distance function. Our system scales each attribute value between 1 and 0 to ensure no feature dominates the distance. We define Euclidean distance between two data points P and Q with n dimensions as:

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad \text{where } P = \langle p_1, p_2, \ldots, p_n \rangle \text{ and } Q = \langle q_1, q_2, \ldots, q_n \rangle$$

Before the adaptive model has a chance to learn new or anomalous operational scenarios, the static engineering model provides the only reliable output estimate. The engineering model provides a type of background information to the overall simulation system. For new and unique operational data, model mixture assigns priority to the static engineering model. As the adaptive model observes operational data for a particular scenario, model mixture assigns priority to the adaptive model. Over time, the system will rarely use the engineering model during model mixture.

If needed, the user can control model mixture manually in order to build trust in this new intelligent system. Since both the engineering model and the adaptive model predict outputs, the user can view a real-time side-by-side comparison of both models. Over time, the user will gain comfort with the adaptive modeling approach and allow the automated mixture to occur.

3.5.3 Online Threshold Selection

In early experiments, we selected the threshold manually by visualizing a time-series of the adaptive model error, the engineering model error, and the distance of the input data from the nearest training set data point. Later we devised an automated approach. This automated approach selects the threshold that minimizes total error based on a hard boundary that switches between both models.

The basic approach is a linear search with a constant step size. The system calculates the step size using the minimum, maximum, and step count parameters. The pseudocode is provided below:

```
step_size = (maximum – minimum) / step_count

for i = 0 to step_count

   threshold_i = minimum + (i * step_size)

   error = evaluateMixedModel(threshold_i)

   if error < minError

         minError = error

         optimal_threshold = threshold_i

return optimal_threshold
```

The threshold selection process occurs at a fixed frequency. For the purposes of our experiment, we selected 24 hours as the threshold selection frequency. This frequency is very convenient to implement.

Our system measures performance on an independent test set to provide a realistic estimate of the threshold parameter. If we use the training set, the distance to the nearest data point will always be zero. In this case, model mixture always selects the engineering model. This protects our system against using an erroneous adapted model. Our system reverts to using the engineering model if any serious problems occur during training, such as overfitting.

3.5.4 Discussion

There were two major concepts that influenced our approach:

- Comparison to Cascading
- Hard-Boundary vs. Soft-Boundary Mixture

Comparison to Cascading – Cascading [17] is a type of meta-learner (ensemble method) that creates a series of models of increasing complexity using a given learning algorithm. During prediction, cascading selects a single model rather than combining the output of all of the models like bagging and boosting. Cascading selects a model based on the model's confidence in correctly predicting the current data point. If the first model is not confident enough, cascading serially passes the data point down the line of models until it finds one with sufficient confidence.

In the model mixture performed in this experiment, the adaptive model represents the first model and the engineering model is the second model. Our system uses the engineering model when the adaptive model is not confident enough that it can correctly predict from the given system inputs. This is a type of cascading with only two models.

We measure confidence as the distance of a data point from known data points in the training data set based on prior performance. If the model extrapolates well for all data points, then the confidence is always high. This approach assumes that increased extrapolation will increase prediction error. Using this basic assumption, one can approximate predictor accuracy simply using this distance.

We cannot use a previous implementation of cascading for two reasons. First, the engineering model is not learned. It is determined a priori. Second, the engineering model represents a default model that is always the backup when the adaptive model cannot confidently predict a given data point. We could make minor adjustments to the cascading

algorithm to provide these capabilities, but our model mixture approach is simple enough to implement.

Hard-Boundary vs. Soft-Boundary Mixture - Finally, we chose between a hard-boundary mixture and a soft-boundary mixture. In the hard-boundary mixture, we treat the outputs from the adaptive model and the engineering model as mutually exclusive. Either the adaptive model output is used or the engineering model output is used. In a soft-boundary mixture, we blend the outputs from both models based on weighting factors.

For a soft-boundary mixture, a potential approach is to weigh the model outputs using the estimated output accuracy. In the case where the system expects the adaptive model to be 80% accurate and the engineering model is 60% accurate, the adaptive model output will have more influence on the mixed model output. We will explore soft-boundary mixture in future work since the hard-boundary mixture provided sufficient results.

4.0 Battery Model

This section walks through the applications of technical approach to the EPS battery. We provide experimental results and battery-specific details.

We selected a single EPS component for early prototyping and analysis. The ISS battery represents a basic case as it only has a single connection to the EPS. Focusing on a single device allows us to demonstrate of the end-to-end adaptive modeling approach. Prototyping lowers technical risk by evaluating the entire adaptive modeling approach (Section 3.0) and identifying any technological deficiencies as early as possible. We will model the remaining EPS devices using the same approach.

4.1 Feature Selection

The initial candidate feature set contained 54 parameters. We discovered these battery features from engineering specifications. We employed the following techniques to discover the final battery model feature set:

- Domain Knowledge
- Correlation Analysis
- Subset Evaluation

4.1.1 Domain Knowledge

Table 4. Domain Knowledge-Derived Training Set Features. *Using domain knowledge, the training features include the same features found in the engineering model equation with the addition of a key system input. We added the new feature with the chance of increasing model accuracy.*

Feature	Type	Exists In Engineering Model
Voltage	Input	No
Current	Input	Yes
Charge Level	Input	Yes
Future Charge Level	Output	Yes

Using knowledge of the EPS, we selected a candidate feature selection set of size 4 (Table 4) for early analysis. We did not include the output feature during feature selection since we cannot remove it. Of the three input features in the set, only the current and charge level actually appear in the engineering model equations. Electrical voltage is a key system input. We added electrical voltage to the candidate feature set with the belief that it would increase predictor accuracy.

The future charge level feature requires more elaboration. The output from the forecasting battery model is the future battery charge level. The battery's future charge level is dependent on the current charge level. Thus, the charge level represents the battery temporal state. Adding this temporal aspect, we trained the model on a time-series sequence of data. With this knowledge, we added a lagged variable [20][25] (Section 4.2) to the training set. This lagged variable is simply the output from the future sample with some Δt between data points, where Δt represents the sampling period. The following equation demonstrates this concept given that x_t is the voltage (Input 1) and y_t is the current (Input 2) for the current data sample:

$$f(x_{t+1}, y_{t+1}) = f(x_t, y_t) + \delta(x_{t+1}, y_{t+1})$$

where $\delta(x_{t+1}, y_{t+1})$ is the amount of change in the charge level between time t and t + 1.

We discovered experimentally that the lagged variable is necessary to generate an accurate predictor. Without the lagged variable, model error increased by a factor of 100. These results verify our stateful model assumption.

We verified this feature set from the full candidate set using DM feature selection techniques. While we possess detailed EPS knowledge, DM techniques allow feature selection without in-depth domain knowledge. We provide the details in the following sections. We strongly believe our automated feature selection approach will allow engineers to create system models without detailed system expertise.

4.1.2 Correlation Analysis

The full candidate feature set contains 54 input features: 45 are low-level features and three are summaries of the low-level features. These summary features are the average of several internal battery sensors. The remaining six features are miscellaneous battery sensors.

We performed correlation analysis on a sample ISS data set. The analysis demonstrated a perfect correlation between the low level and summary features. Based on these results, we removed all 45 low level features from the feature set. The resulting feature set contained only nine features; the three summary features and the six miscellaneous battery sensors. Table 1 provides sample data from our analysis.

If an analyst does not have the domain knowledge of the presence of summary features, correlation analysis will automatically identify such redundant parameters.

4.1.3 Subset Evaluation

We performed an exhaustive subset search using the Classifier Subset Evaluator algorithm with a LR predictor. The exhaustive subset search builds all possible combinations of features from the candidate set. Given the nine remaining input features, we evaluated all 512 (2^9) subsets. The evaluation chose the same three features (voltage, current, charge level) as our use of domain knowledge (Table 4), but with an additional thermal system feature. Since we did not construct a thermal model for this experiment, we had to ignore this additional feature. Rerunning the analysis using the remaining eight input features generated the same features as the use of domain knowledge (Table 4). The future introduction of an ISS battery thermal model would likely enhance battery model performance even further.

Our results support our decision to add voltage as an input feature (Section 4.1.1). Subset evaluation shows that model accuracy improves when using this feature. This value does not appear in the original engineering model equations. We discovered this new information using unbiased data mining techniques rather than existing biased domain knowledge (engineering model equations). We recommend that engineers analyze all potential features, rather than manually pruning sensor values using domain knowledge.

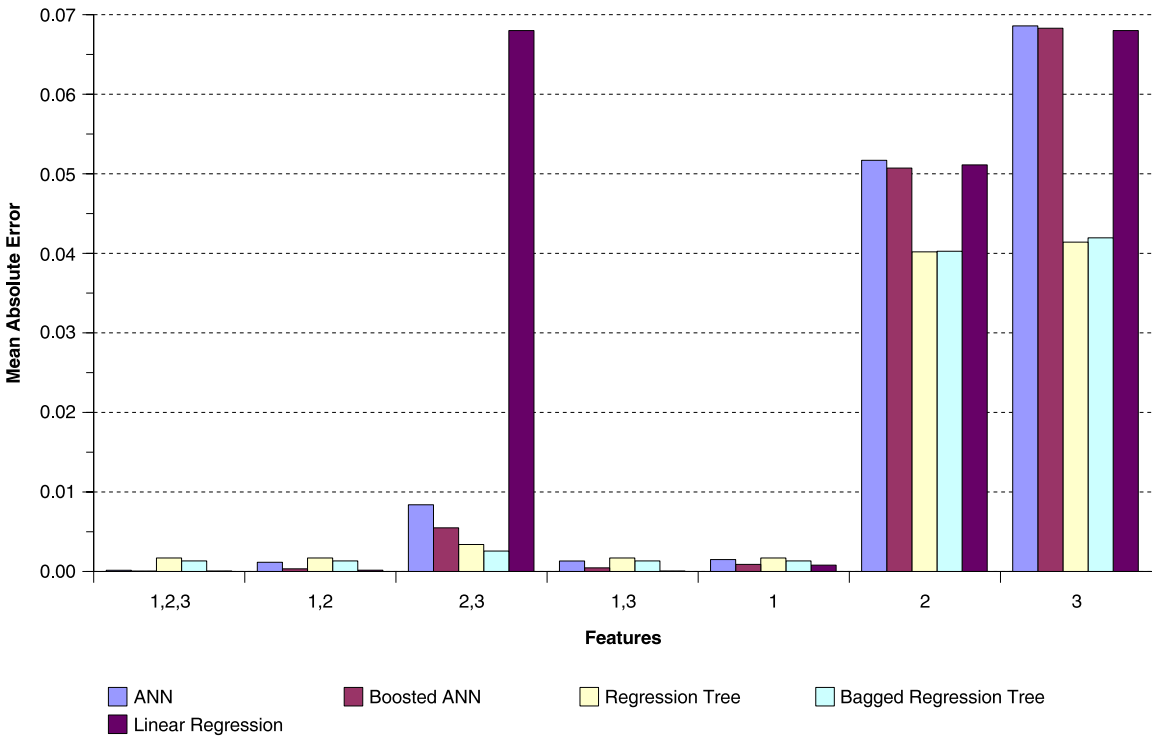

Figure 18. Feature Subset Comparison. *Feature 1 provides the greatest increase in model accuracy. The combination of all three features allows construction of the most accurate model. We provide a better view of the top performing feature subsets in Figure 19.*

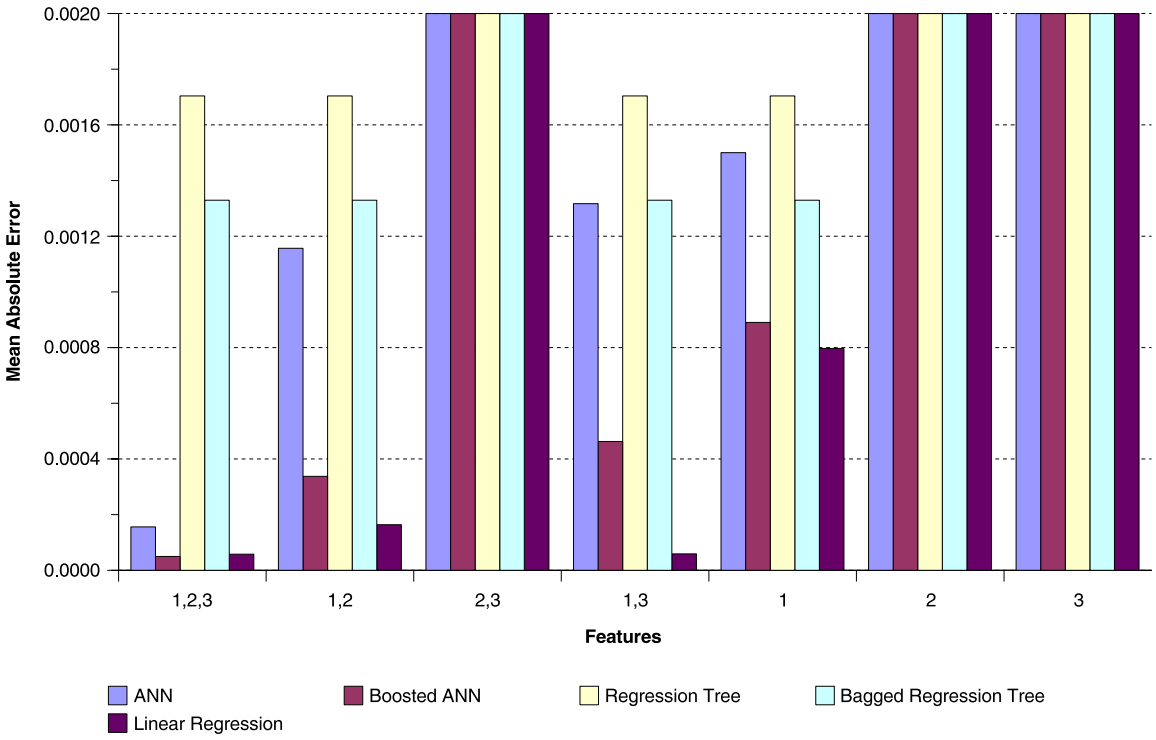

Figure 19. Feature Subset Comparison Zoomed. *This chart provides detail for the best performing feature subsets in Figure 18.*

Figure 18 displays the error rate comparison of the candidate ML algorithms built using all subsets of the three automatically selected features. Figure 19 displays the same data with a different scale to distinguish the difference of the best feature subsets. The battery state of charge parameter (feature 1) provided the largest reduction in error. The addition of either the voltage (feature 2) or current (feature 3) parameter to charge level actually slightly increased error over LR models built with using only the charge level. We created the most accurate model for all candidate ML algorithms only when we combined all three features.

4.2 Data Preprocessing

The ISS on-board software samples the battery sensors every 10 seconds (0.1 HZ). After converting the raw data to the WEKA format, we create a lagged variable for the output. Creating a lagged variable from the successive data sample output requires special care. Each data point must contain the future device output in 10 seconds. Figure 20 shows the raw sensor data at time *t*. To construct the training and evaluation sets, we must copy the output for time point $t + 1$ back to the data point at time *t*. The right-hand side of Figure 20 demonstrates this copy process.

4.2.1 Training Set Construction

As previously mentioned, we removed all data points with missing values from the candidate data set to create the training set.

Predictions generated from the battery model forecast into the future at the same period as the training data. Specifically, the output is the estimated charge level 10 seconds into the future. For the battery model, 0.1 HZ is the maximum allowable frequency. A lower frequency is achievable by adjusting the training data set to contain predictions for the desired period. In this experiment, no changes were made to the 0.1 HZ forecasting

Sensor Data

Time	Input 1	Input 2	Output
1	0.542	2.123	12.346
2	0.567	2.234	13.234
3	0.789	2.089	13.098
4	0.649	2.105	15.726

Lagged Variable Training Set

Time	Input 1	Input 2	Output	Output @ Time + 1
1	0.542	2.123	12.346	13.234
2	0.567	2.234	13.234	13.098
3	0.789	2.089	13.098	15.726
4	0.649	2.105	15.726	etc..

Figure 20. Data Set with Lagged Variable. *We copy the output for successive data points back to the previous data point to train a time-series forecasting model. The last column of the Lagged Variable Training Set is the target prediction value.*

frequency.

4.2.2 Evaluation Set Construction

We constructed battery model evaluation sets using the technique discussed in Section 3.2.2. For the 90-day evaluation in 2006, there were approximately 130 data sets with at most four consecutive missing data points (μ=4). Linear interpolation over a 40 second period is sufficiently accurate for this evaluation. We verified data quality through manual visual inspection and automated quantitative analysis (Section 3.2.2).

4.3 Algorithm Selection

Initially we used the EPS battery to build a candidate list of ML algorithms. We detail this preliminary algorithm evaluation activity below. We used these algorithms in our initial studies to measure to feasibility of our technical approach. We feel the reader will find the results from this activity interesting. Therefore, those results are included.

Over the duration of the project, we revised the algorithm list for the purpose of online algorithm selection. Online algorithm selection is a much more thorough and complete mechanism for algorithm evaluation. We wrote custom software to fulfill our ideas. We provide discussion and results below in Section 4.3.3.

4.3.1 Preliminary Algorithm Evaluation

First, we constructed a single training set and a single corresponding independent evaluation set to evaluate learned models. The training set contained 24 hours of data from 2006. We used this data set to construct the learned model. The evaluation set contained the 3 hours of data following the training set timeframe. This is the most realistic evaluation since

the final system will contain a model that predicts near-term system operation after training on the most recent historical data.

Second, we constructed a candidate list of algorithms. We based our candidate selection on the problem domain and known characteristics. The candidate list contained the following algorithms as provided by the WEKA toolset:

- Linear Regression (LR)
- Artificial Neural Networks (ANNs)
- Support Vector Machines (SVMs)
- k Nearest Neighbors (kNN)

We provide the results in Figure 21 and provide details discussion and analysis below.

Linear Regression – This technique provided the best performance. This is likely due to

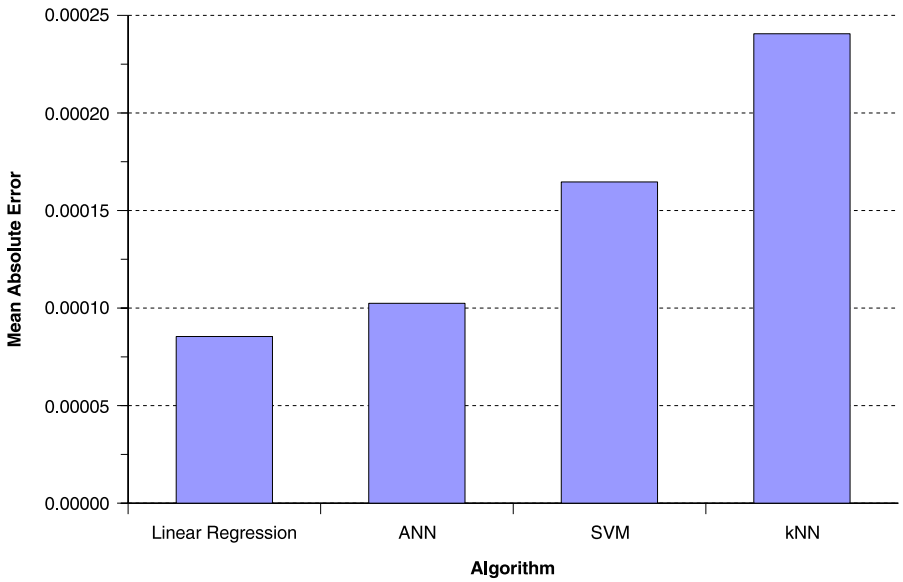

Figure 21. Algorithm Comparison. *Linear Regression and Artificial Neural Networks offer the best prediction performance for ISS battery modeling.*

the linear behavior of an electrical battery.

Artificial Neural Networks – ANNs performed almost as well as LR. We trained a network with a fully interconnected hidden layer using standard backpropagation and the default parameters outlined in Table 3. It seems ANNs are a more flexible approach than LR. For example, when the lagged variable was removed from the training set in Section 4.2, ANNs drastically outperformed LR. We expect this inherit robustness to resurface when modeling other systems.

Support Vector Machines – An SVM with the default parameters did not perform as well as LR and ANNs. SVMs also required extended training times and detailed parameter tuning. Since LR and ANNs were successful, we did not explore SVMs any further for the battery model.

K Nearest Neighbors – First we evaluated varying values of k from 1 to 10. Setting k=3 offered the best accuracy. Using three nearest neighbors, the other algorithms still outperformed kNN. In addition, kNNs were an order of magnitude slower at predicting values. We did not explore kNNs any further since prediction performance is of the utmost importance for our simulation system.

4.3.2 Simulation Model Performance

Sensor Data

Time	Input 1	Input 2	Output
1	0.542	2.123	12.346
2	0.567	2.234	13.234
3	0.789	2.089	13.098
4	0.649	2.105	15.726

Time-Series Predictions

Time	Input 1	Input 2	Output	Predicted Output @ Time + 1
1	0.542	2.123	12.346	13.534
2	0.567	2.234	13.534	13.284
3	0.789	2.089	13.284	15.065
4	0.649	2.105	15.065	etc..

Error Rates

Time	System Output	Predicted Output	Absolute Error
2	13.234	13.534	0.300
3	13.098	13.284	0.186
4	15.726	15.065	0.661

Figure 22. Evaluating Time-Series Forecasting. *Given the two inputs and the current device output, the model predicts the device output for the next time interval. This example demonstrates open-loop forecasting where the predicted output is an input value for the next prediction. Evaluation data only provides the true output value is as an initial condition.*

We performed the previous algorithm evaluation using the traditional hold out method. This method assumes data point order independence. We've established that we must measure battery model simulation performance using time-series forecasting (Figure 22). We base each successive output on the output at the previous time point. Thus, forecasting models can slowly compound any error over time. We produced two time-series of outputs using a full 3-hour set of input values: one for the learned model and one for the engineering model. We compared these two time-series against the actual battery output sensor data.

We trained the ANN using a validation set to prevent overfitting. The results in Figure 23 and Figure 24 demonstrate the success of the ANN-based model over the traditional static engineering model. In Figure 23, the engineering model output slowly drifts from the actual value as forecasting time increases. The ANN model stays in synch with the peaks and valleys of the battery output signature. Figure 24 shows the corresponding squared error (difference between the predicted value and actual value). The ANN model error always returns to a near zero value while the engineering model error almost monotonically increases.

Experiments during simulation evaluation offered an interesting observation. We managed to construct an ANN that was less accurate than the engineering model when predicting single data points in time (time independent). We then generated a time-series forecast over the 3-hour period of input data. In this case, the ANN outperformed the engineering model in the simulation evaluation (time dependent). Therefore, ANNs preserve time-series characteristics that allow better forecasting than actual instantaneous predictions.

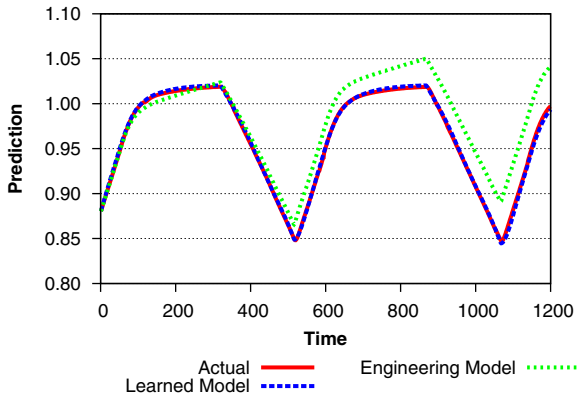

Figure 23. Battery Simulation Output vs. Time. *The learned model more accurately simulates true EPS battery performance, while the engineering model slowly drifts away.*

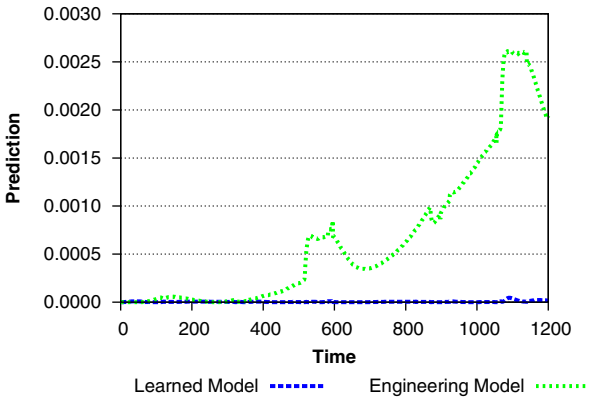

Figure 24. Battery Simulation Squared Error vs. Time. *As simulation error quickly expands for the engineering model, the learned model always reflects true performance.*

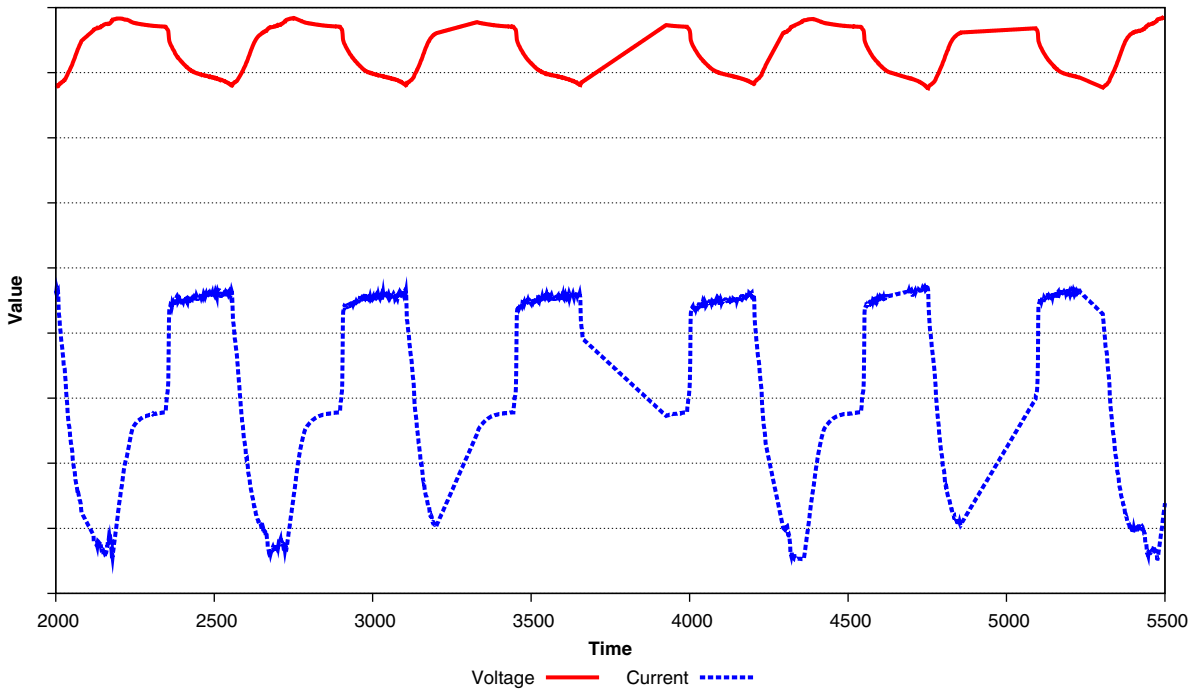

Figure 25. Input Error due to Linear Interpolation. *Between times 3600 and 3900, linear interpolation breaks the cyclic behavior in both voltage and current.*

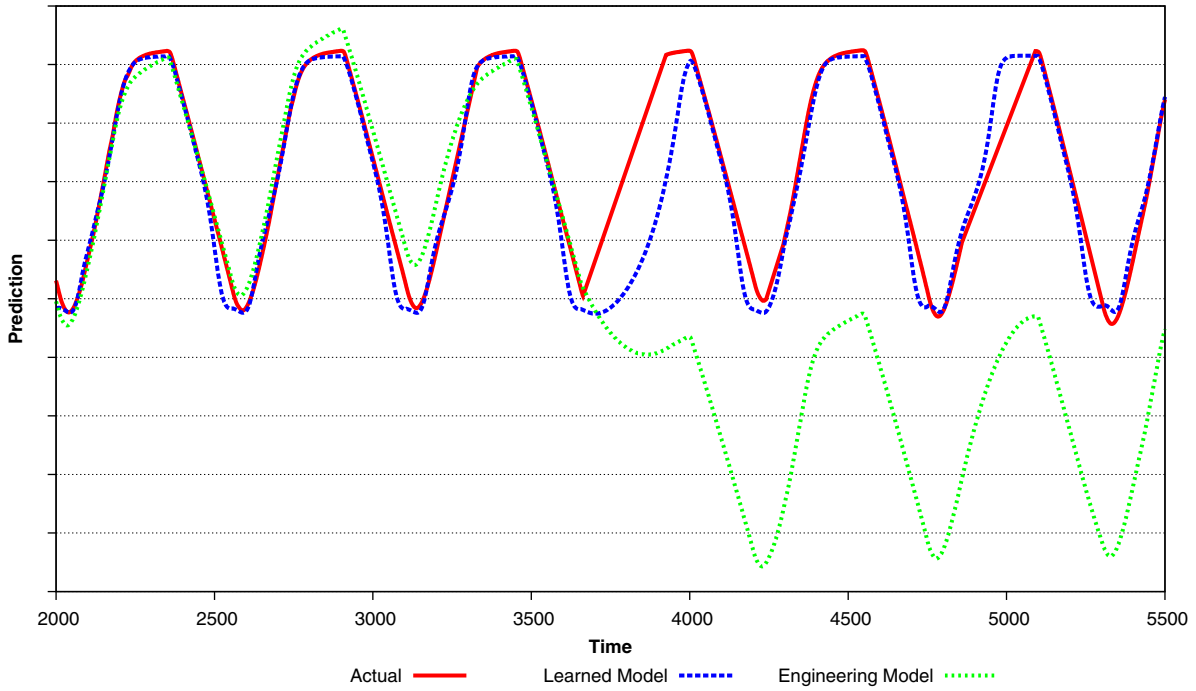

Figure 26. Robust Artificial Neural Network Performance with Erroneous Input. *The ANN forecasts the output accurately while the engineering model cannot handle the input error (*Figure 25*).*

We also observed that an ANN was able to handle erroneous input data more gracefully than the engineering model. The plot in Figure 25 clearly demonstrates erroneous input data. We generated this plot in early phases of this thesis when manually searching for high quality evaluation sets. Linear interpolation does not preserve the cyclic behavior as it cuts through hills and valleys in the sensor's signature. The following plot in Figure 26 compares performance of an ANN and the engineering model when using interpolation on large periods of missing data. The error causes significant deviation in the engineering model. The engineering model is slow to recover while the ANN stays in synch with the true output.

Most simulation systems assume noiseless input data, but we can see that an ANN is more robust and can perform outperform LR in the face of noisy/inaccurate data.

4.3.3 Online Algorithm Selection

Every 24 hours worth of data, we evaluate the candidate algorithms from Section 3.3.3 against the representative data set using 10-fold cross validation. Our system derived the following results using the first 90 days of data from year 2006. We used the established default algorithm parameters (Table 3) in this experiment.

Figure 27 indicates that linear regression offers the greatest average accuracy although it is not always the best performer. Our system selected linear regression 58% of the time and the boosted ANN the remaining 42% of the time. The boosted ANN offers a lower mean error and reduced error variance over the standard ANN. Regression trees and bagged regression trees are not competitive although it is noteworthy that the bagged regression tree always outperforms the plain regression tree.

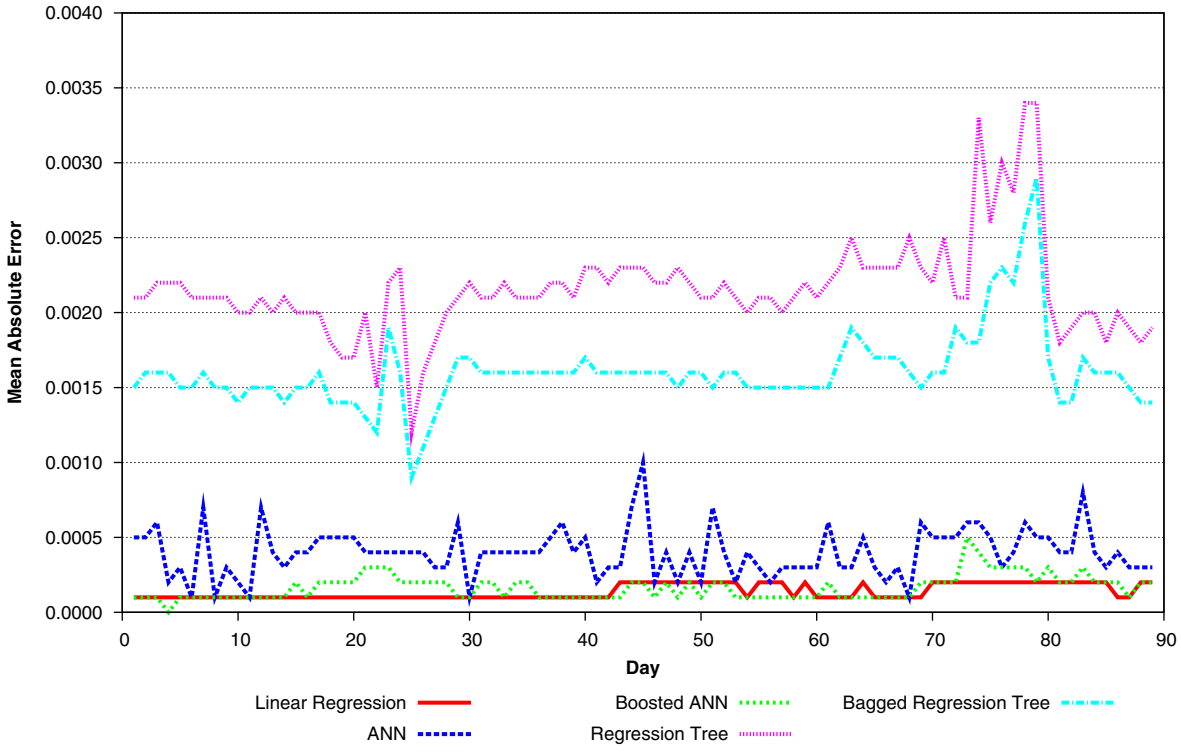

Figure 27. Algorithm Error Comparison - Battery Model (2006 Day 1 to Day 90). *Boosted ANNs and Linear Regression provide comparable predictive accuracy for the ISS battery model.*

4.4 Online Adaptation

During online adaptation, our system trained a model every 24 hours using the ML algorithms with the lowest error rate in Figure 27. A 90-day historical evaluation using ISS data from 2006 (Figure 28) demonstrates high predictor accuracy after each training cycle. Our system measures the model accuracy on an independent test set. The independent test set contains the next 24 hours worth of data after the training data set. The adaptive model drastically outperforms the engineering model by reducing the mean error rate by 75%. In addition, the adaptive model reduces the variance in the mean error rate by 45%. The adaptive model is clearly more stable than the engineering model.

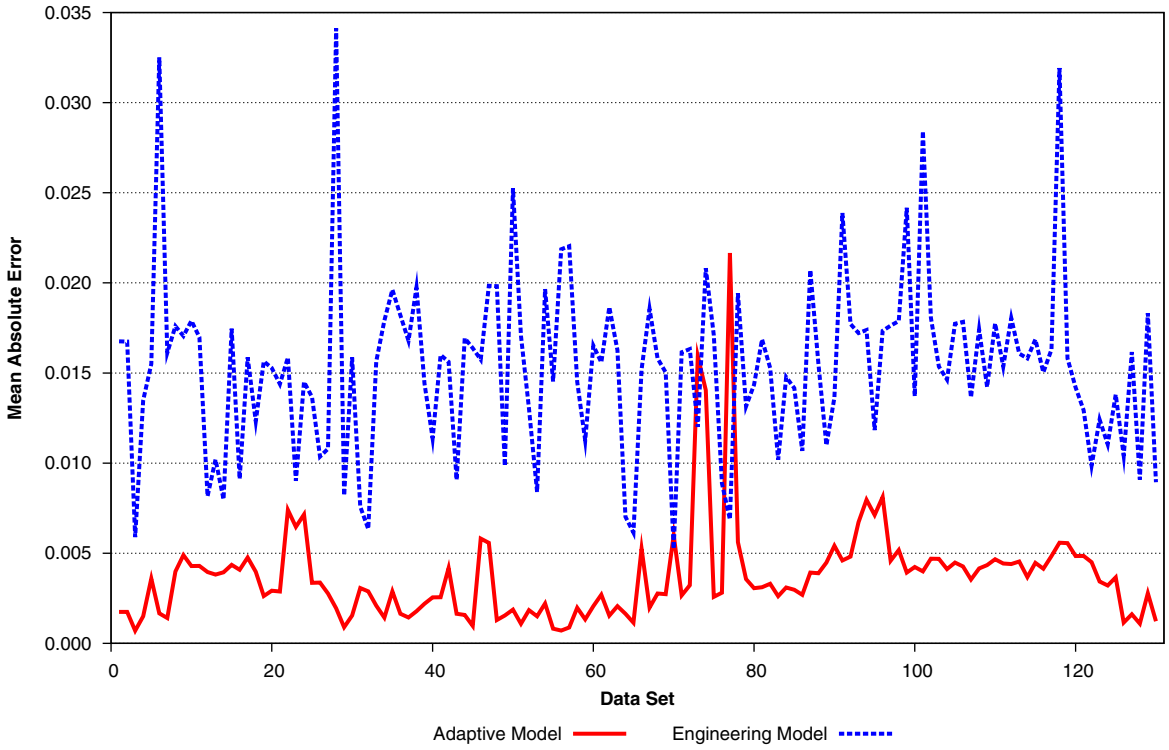

Figure 28. Adaptation Performance - Battery Model (2006 Day 1 to Day 90). *The adaptive model learned every 24 hours significantly outperforms the existing engineering model on average, but we must address spikes in the error (datasets 72 through 77) further.*

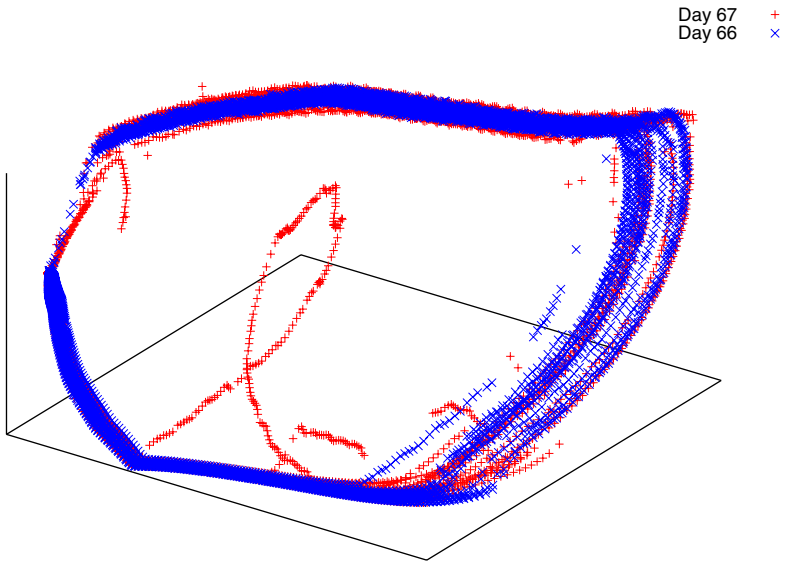

Figure 29. Newly Explored Battery Operation Region. *Day 67 explored new areas of the battery operating range (i.e., input space) reflected in the red loop in the middle of all day 66 data points. The adaptive model generated on day 66 data performs poorly on day 67 data. Model mixture solves this problem.*

The adaptive model error rate spikes above the engineering model error rate for a few data sets between 72 and 77. The system built all of these data sets from day 67 data. There were changes in the operating mode on day 67 (Figure 29). Model mixture is necessary in this case with newly explored operating regions. The following section provides an illustration of how we significantly reduce these spikes using the final mixed model.

This battery model adaptation approach used a representative data set of 2000 points every 24 hours. Each clustering and training session took approximately 10 seconds, which is the same as the sensor sampling frequency. This means it is possible to perform clustering and build a new classifier for each incoming data point if necessary.

4.5 Model Mixture

We performed model mixture in the same manner described in Section 3.5. On average, the mixed model outperforms the adaptive model (Figure 30). We see the greatest benefit in the range of data sets 72 and 77. Our system bounds the maximum error by using the engineering model for data points further away from the training data. Model mixture sometimes increases the error over the purely learned model, but our goal is to reduce the maximum error value. We want our simulation system to avoid harmful spikes. Model mixture performs the role of a safety net, ensuring good accuracy in the face of unexplored operating regions.

The evaluation of the first day uses zero as the mixture threshold. A mixture threshold of zero means the system uses the engineering model exclusively. We need this to initialize the

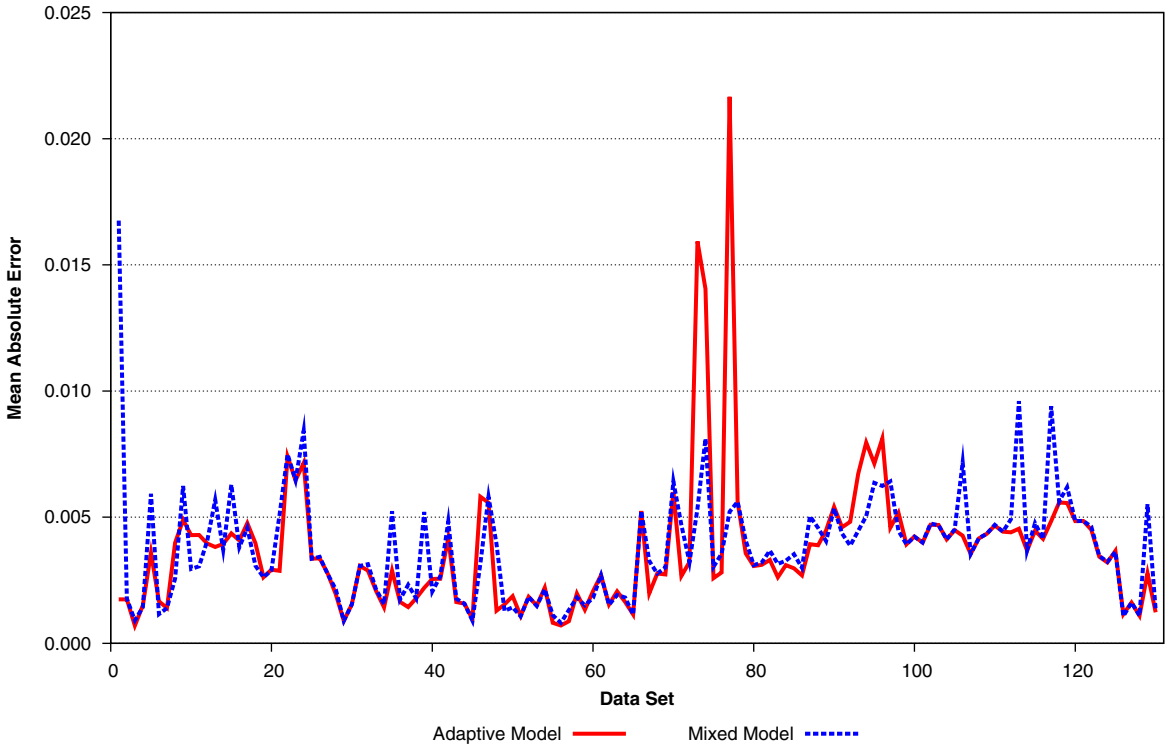

Figure 30. Model Mixture Performance - Battery Model (2006 Day 1 to Day 90). *Model mixture notably reduces the error rate spikes near days 72 and 77. This is the expected behavior since model mixture limits the amount of extrapolation performed by the adaptive model.*

mixture threshold to a reasonable value. We desire this behavior because the system will use the engineering model when we are not certain of adaptive model performance. Since zero is the mixture threshold for the first day, the error plots for the adaptive model always reflect the engineering model error for day one.

Our results provided in this section demonstrate the successful application of our proposed technical approach to the EPS battery. Online algorithm selection selects algorithms during online adaptation that generate models more accurate than existing engineering models. The model mixture component successfully reduces the error spikes when adaptive models are faced with making predictions in new, unexplored scenarios.

5.0 BCDU Model

This section walks through the applications of technical approach to the EPS BCDU. We provide experimental results and BCDU-specific details.

The Battery Charge Discharge Unit (BCDU) [2] is the device responsible for charging and discharging the battery. The BCDU is the next hardware device upstream from the battery (Figure 2), making it our primary candidate for the next modeling experiment.

The BCDU model has two critical output parameters necessary for EPS simulation: electrical current and electrical voltage. Our system learns two independent adaptive models to predict each of the output parameters. The following sections provide discussion and results for both the BCDU current and voltage models.

5.1 Feature Selection

The BCDU contains only five sensors available for learning. Three of these sensors are input features and the remaining two sensors are output features. This feature selection task is on a much smaller scale than the battery case (Section 4.1).

We employed our primary feature selection techniques to discover the final feature sets for the BCDU voltage and current models:

- Domain Knowledge
- Correlation Analysis
- Subset Evaluation

5.1.1 Domain Knowledge

The engineering model equations for BCDU voltage and current include variables that correspond to all input sensor values. Therefore, we did not reduce the candidate feature set using our domain knowledge.

5.1.2 Correlation Analysis

We found no linear correlation between the three candidate input features. We did not expect any redundant features.

5.1.3 Subset Evaluation

We manually performed subset evaluation since the BCDU only has three candidate features. This means we only have to analyze seven sets. We ignored the empty feature set

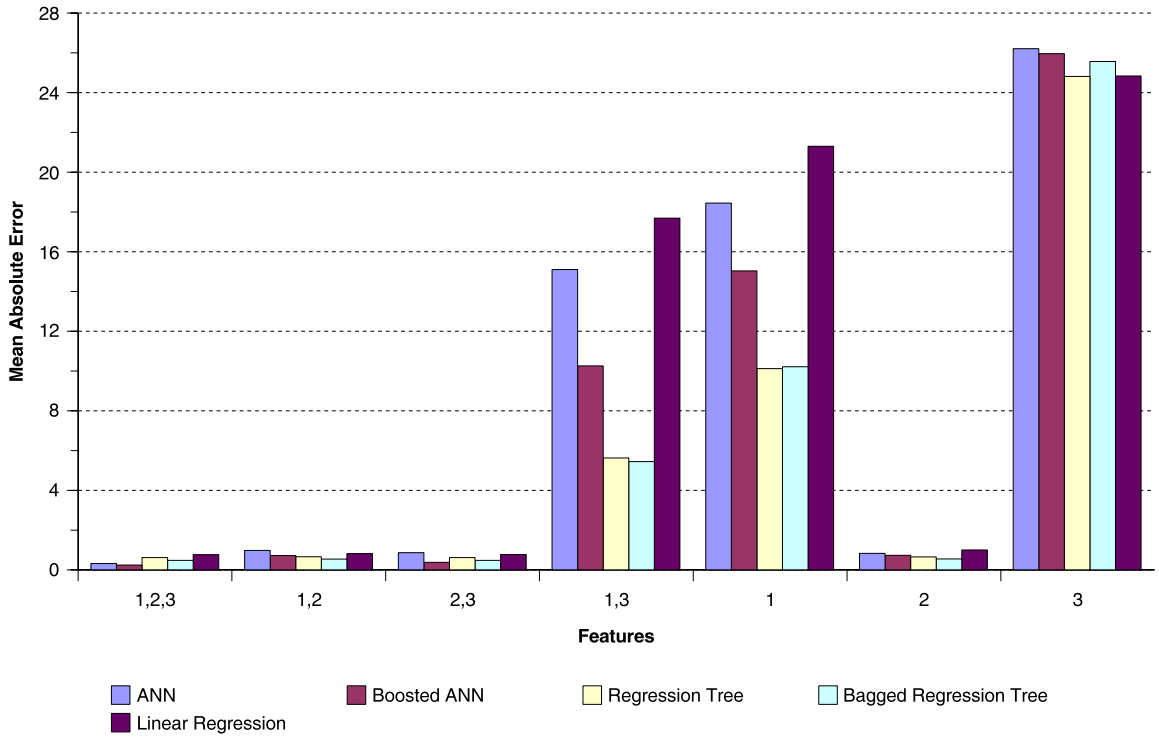

Figure 31. Feature Subset Evaluation Results - BCDU Current Model. *The combination of all three features offers optimal model accuracy for all candidate algorithms.*

since it is impossible for the system to learn an accurate model without any data. We evaluated each of the candidate algorithms on 24 hours of BCDU sensor data using 10-fold cross validation.

BCDU Current – Figure 31 shows the results for all combinations of feature subsets. The feature set containing all three candidate features (1, 2, and 3 for simplicity) performed the best for all candidate algorithms. Feature 2 is clearly the most significant of the three candidate features.

BCDU Voltage – Figure 32 shows the results for all combinations of feature subsets. The feature set containing all three candidate features performed the best for all candidate algorithms.

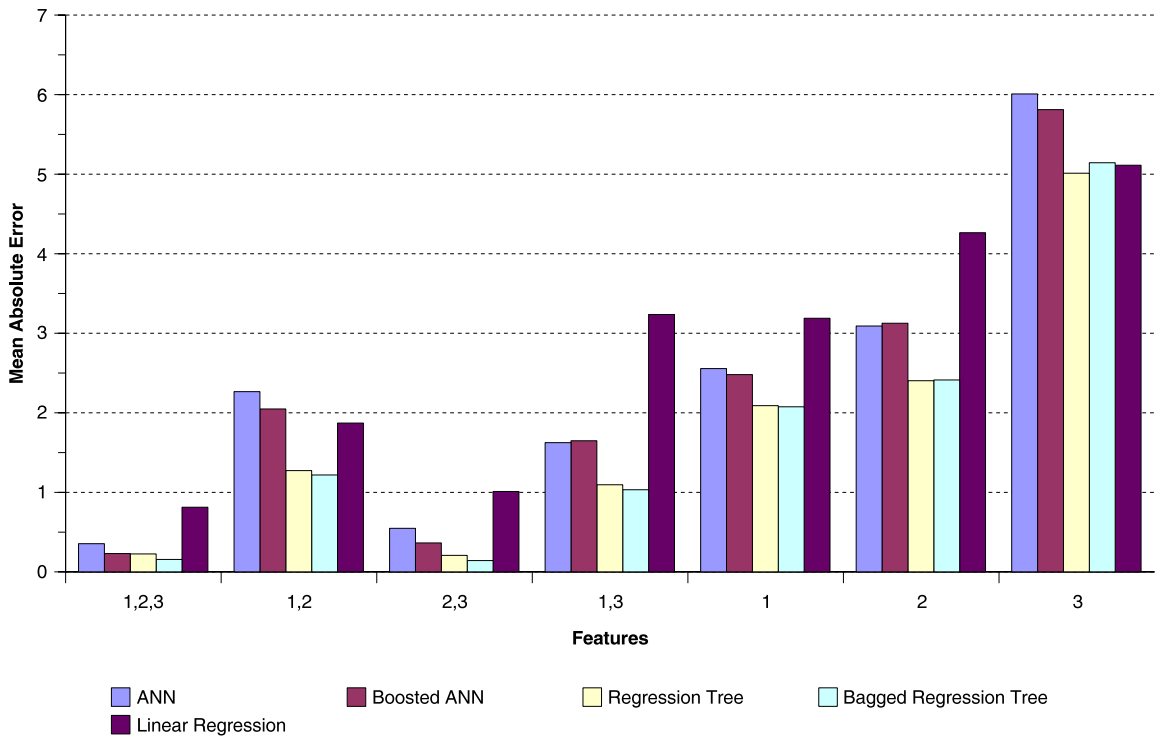

Figure 32. Feature Subset Evaluation Results - BCDU Voltage Model. *The combination of all three features offers optimal model accuracy for all candidate algorithms.*

In the end, we used the same features for both the BCDU voltage and current models. There were no surprises with our results. However, we validated our knowledge using established DM feature selection techniques.

5.2 Data Preprocessing

5.2.1 Training Set

We created training sets by simply throwing out data points with missing information. The BCDU is stateless, so there is no need to train the model against a time-series of data. We can treat each incoming data point as independent.

5.2.2 Evaluation Set

We constructed evaluation sets in the exact same manner as the training set. The BCDU is stateless, so there is no need to treat the input data as a time-series for evaluation purposes. As with the training set, we treated each incoming data point as independent. The benefit of using this approach is that traditional implementations of n-fold cross validation are directly applicable. This approach is much more straightforward than the time-series forecasting model evaluation technique.

When performing BCDU evaluation, we kept the set in time order to simply results analysis. Viewing prediction plots in time sequence order allows the identification of potential prediction trends to compare against the original time-ordered sensor data.

5.3 Algorithm Selection

Every 24 hours, our system evaluated the candidate algorithms against the representative data set using 10-fold cross validation. We derived the following results using the first 90 days of data from year 2006.

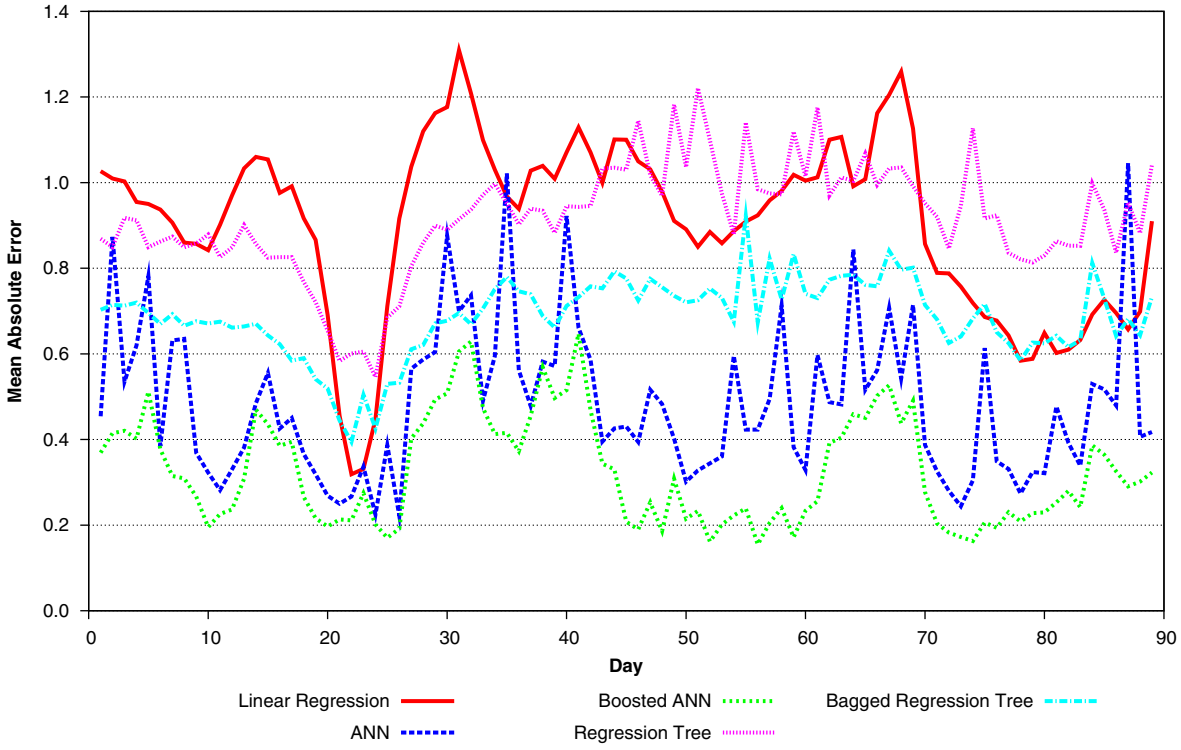

Figure 33. Algorithm Error Comparison - BCDU Current Model (2006 Day 1 to Day 90). *Boosted ANNs always provide the highest degree of accuracy and stability.*

5.3.1 Current Model

Figure 33 indicates that a boosted ANN always offers the lowest error. A plain ANN offers the second lowest error, but occasionally the error rate spikes above other algorithms. The boosted ANN appears to be much more stable. Interestingly enough, all algorithms seem to follow a similar error trend. We speculate this is due to each algorithms inherit ability to represent an approximately linear function

For BCDU current, a boosted ANN appears to be the optimal choice. The need for automated online algorithm selection is debatable. However, online algorithm selection offers the benefit of adjusting to a more suitable algorithm in the case of system evolution or an erroneous training section.

We used the default algorithm parameters (Table 3) except for the plain ANN. The ANN contains a single hidden layer with the same number of nodes as the number of input features. We derived the ANN network design experimentally using a sample of BCDU current data.

5.3.2 Voltage Model

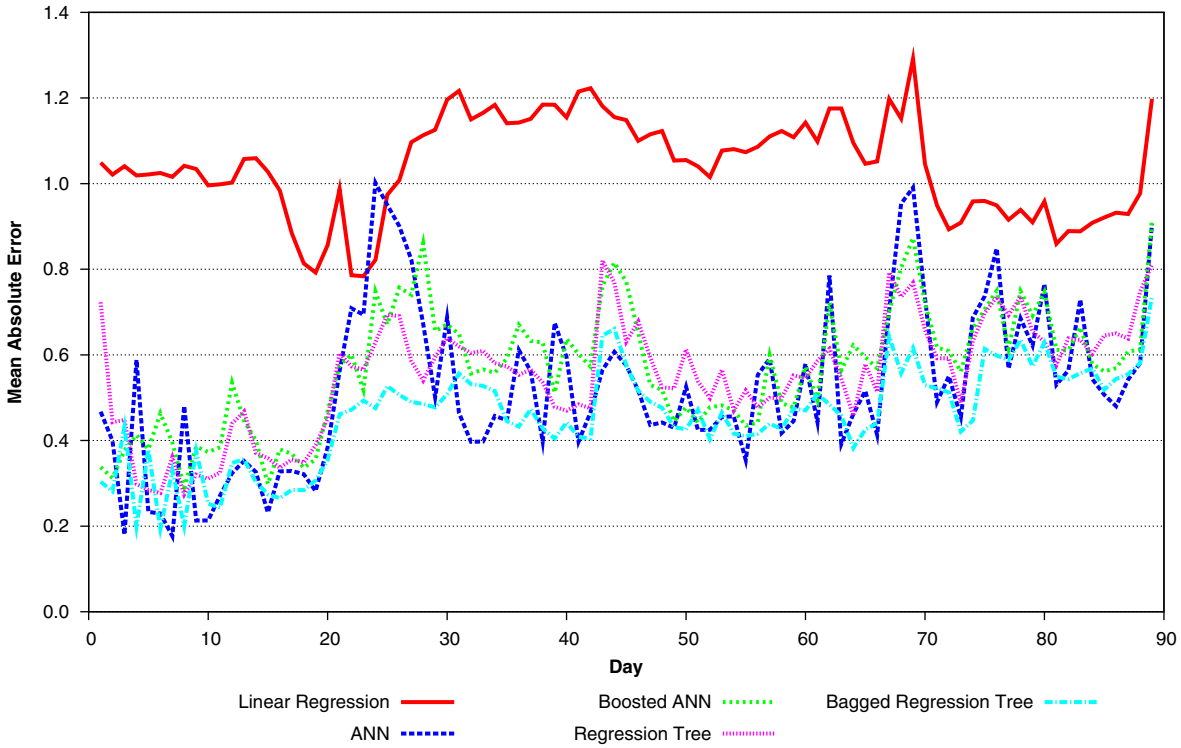

Figure 34. Algorithm Error Comparison - BCDU Voltage Model (2006 Day 1 to Day 90). *Bagged Regression Trees provide the best overall accuracy, but all algorithms with the exception of Linear Regression offer comparable results.*

Figure 34 indicates that a bagged regression tree offers the greatest average accuracy. Unlike BCDU current, a single algorithm is not always the best performer. Our system selects the bagged regression tree 60% of the time while it selects the ANN the remaining 40% of the time. The bagged regression tree offers an overall lower mean error and lower error variance over the ANN.

Interestingly enough, we also observed that the single ANN model outperforms the boosted ANN on average. In contrast, the bagged regression tree usually outperforms the plain regression tree model. We believe the boosting process overfits the BCDU voltage data.

Another interesting observation is that with the exception of Linear Regression, all algorithms provide comparable accuracy and follow the same error rate trend.

We used the default algorithm parameters (Table 3), except for the plain ANN. The ANN has hidden layers with five, three, and five nodes respectively. We experimentally derived the ANN network design using a sample of BCDU voltage data.

5.4 Online Adaptation

The system trains the adaptive model on the latest incoming data points using the most recent algorithm chosen during algorithm selection.

5.4.1 Current Model

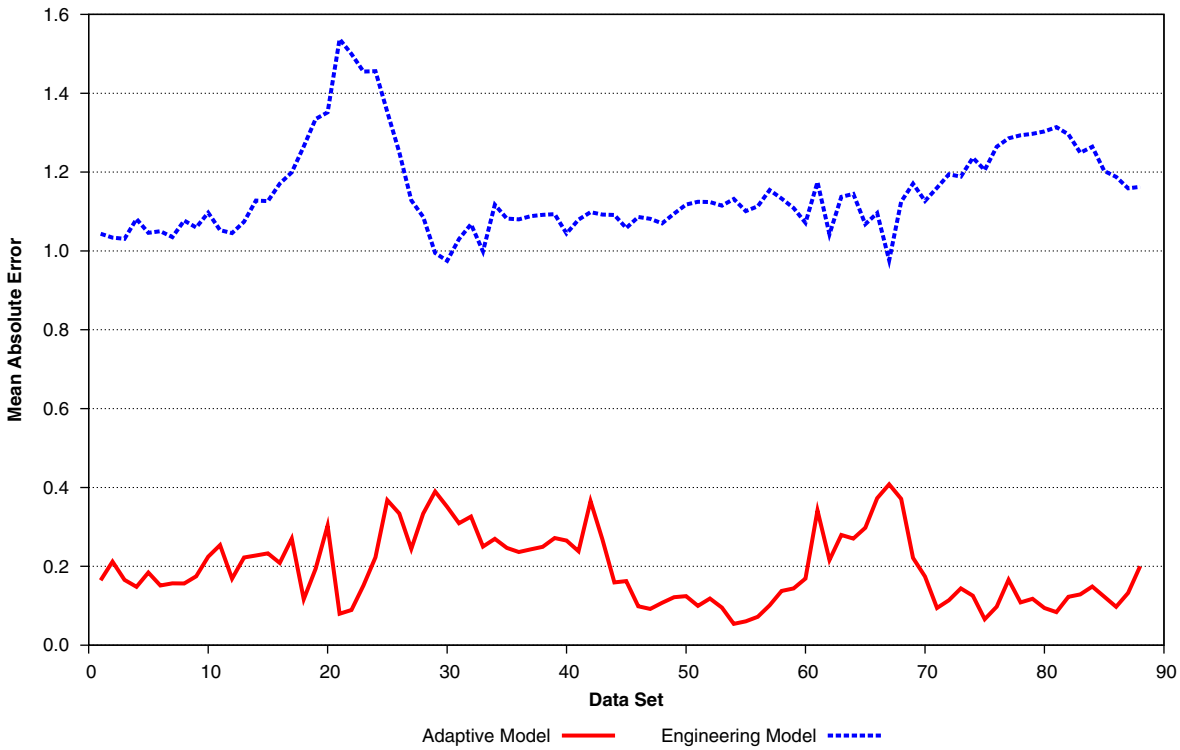

Figure 35. Adaptation Performance - BCDU Current Model (2006 Day 1 to Day 90). *The adaptive model, based on Boosted ANNs, significantly outperforms the static engineering model.*

In the case of BCDU current, the results shown in Figure 35 are from a boosted ANN model. On average, the adapted model boasts an 83% reduction in mean error over the static engineering model. The error bounds for the ANN is well within reason and never worse than the engineering model's maximum error.

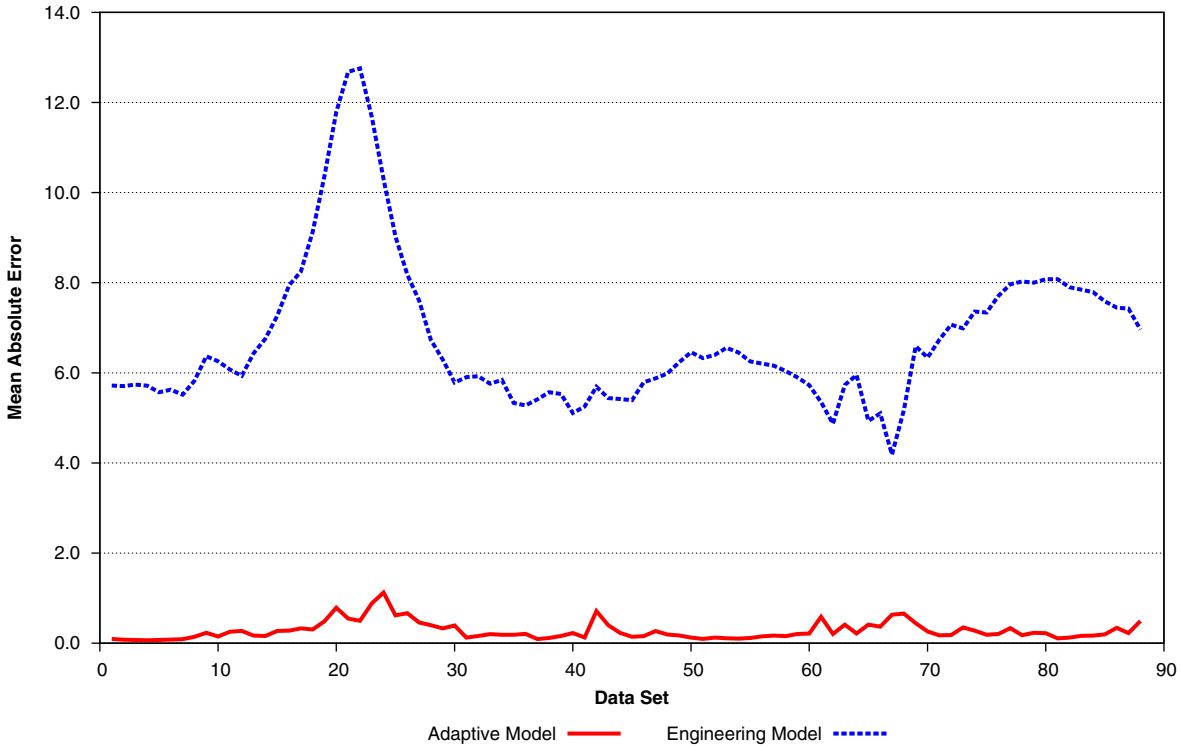

Figure 36. Adaptation Performance - BCDU Voltage Model (2006 Day 1 to Day 90). *The adaptive model significantly outperforms the static engineering model. There is also no correlation between the error rate of the adaptive model and the training algorithm used (Bagged Regression Trees and ANNs).*

5.4.2 Voltage Model

The increase in accuracy is even more significant when adapting models based on BCDU voltage data (Figure 36). On average, the adapted model boasts a 96% decrease in mean absolute error over the static engineering model. The boosted ANN is very successful as it also decreases the error variance by 88% and maximum error value by 37% over the engineering model.

Online algorithm selection switches between a plain ANN and a bagged regression tree. As illustrated in Figure 36, both algorithms performed extremely well. There is also no

noticeable correlation between the peaks in the error rate and the selected algorithm. Both algorithms are equally suitable for BCDU voltage prediction.

5.5 Model Mixture

The system performs model mixture using automated threshold adjustment in order to minimize the error from previous data sets. Model mixture did not provide a significant improvement for either BCDU current or voltage models. Either the adaptive models correctly induce (extrapolate) unknown data points or the BCDU operated in the same regions during this 90-day period. After analyzing the data, we noticed the BCDU operating region was not constant. In addition, our system always selected mixture thresholds near the maximum distance. We conclude that the adaptive models extrapolate extremely well. Model mixture still provides the safety net to bound poor extrapolation or erroneous training sessions. Without this safety net, user confidence in the adaptive system can diminish.

For both the voltage and current models, the evaluation of the first day uses zero as the mixture threshold. Refer to Section 4.5 for our rationale.

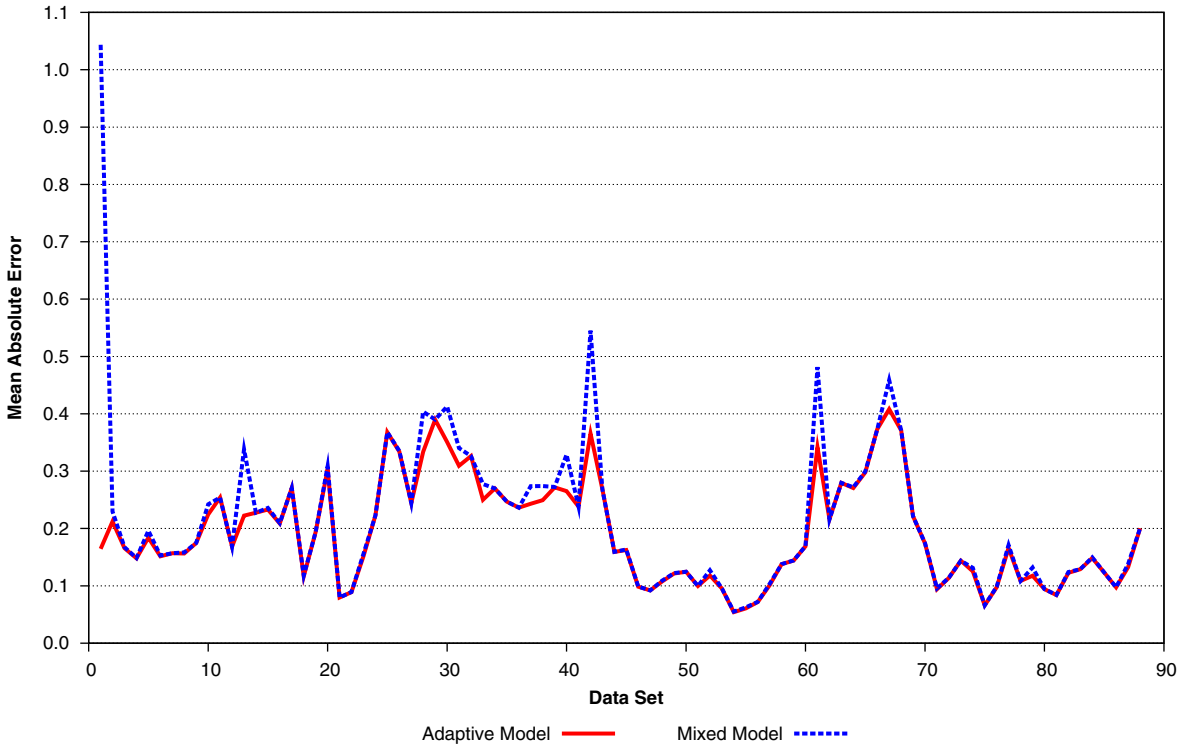

Figure 37. Model Mixture Performance - BCDU Current Model (2006 Day 1 to Day 90). *Model mixture offers no real benefits for the BCDU current model since the adaptive model does not have to extrapolate.*

5.5.1 Current Model

The system rarely uses the engineering model during model mixture. Thus, the error rates for the adaptive model and mixed model usually coincide (Figure 37). Model mixture slightly increases the predictive error for a few datasets, but the increased error rate does not approach the engineering model error rate. The mixed model only offers a minuscule accuracy improvement over the adaptive model for one of the 90 days. As portrayed in Figure 37, the improvement is insignificant since the difference is not even visually discernable.

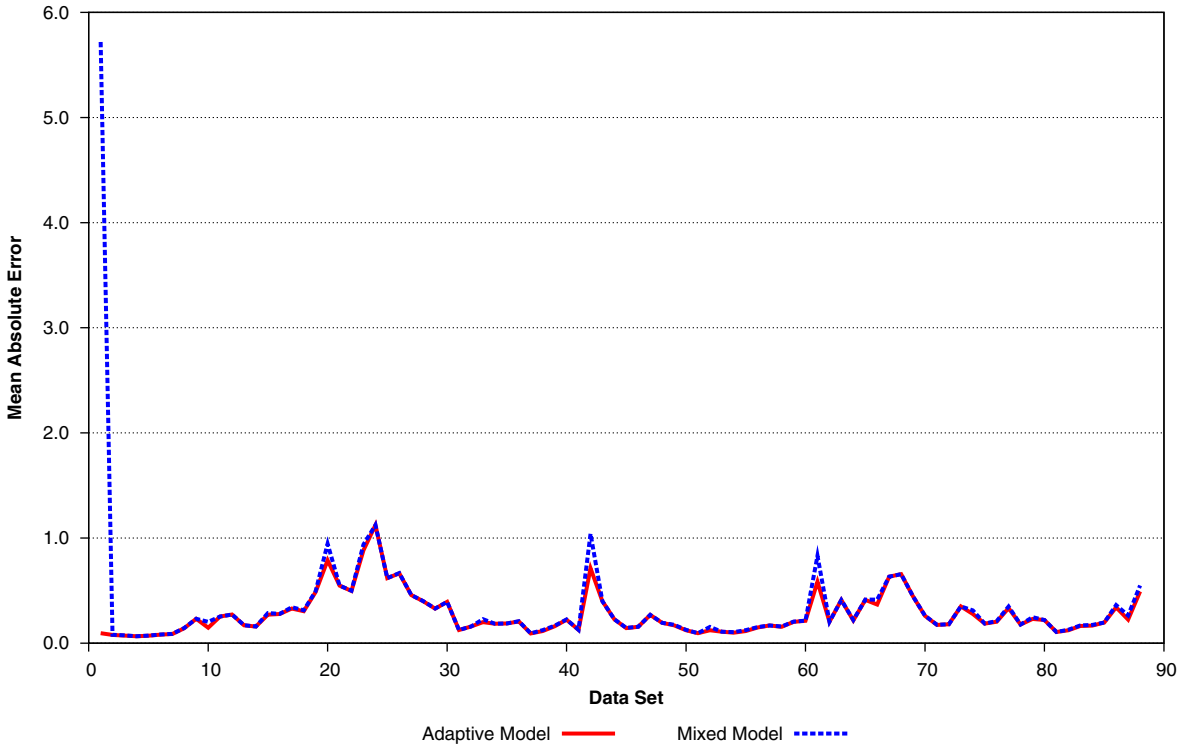

Figure 38. Model Mixture Performance - BCDU Voltage Model (2006 Day 1 to Day 90). *Model mixture offers no real benefits for the BCDU voltage model since the adaptive model does not have to extrapolate.*

5.5.2 Voltage Model

Similar to the results we witnessed for the BCDU current model, the system rarely uses the BCDU voltage engineering model during model mixture (Figure 38). Once again, model mixture slightly increases the predictive error for a few datasets, but the increased error rate does not approach the engineering model error rate. The mixed model only offers a minuscule accuracy improvement over the adaptive model for one of the 90 days. As portrayed in Figure 38, the improvement is insignificant since the difference is not even visually discernable.

Our results provided in this section demonstrate the successful application of our proposed technical approach to the EPS BCDU. Online algorithm selection selects algorithms during online adaptation that generate models more accurate than existing engineering models. The model mixture component is somewhat benign since the learned models are always more accurate than the existing engineering models. However, model mixture is still necessary as a safety net to prevent grossly inaccurate predictions.

6.0 Related Work

This work most relates to the field of data stream mining [19]. We build system models from real-time and real-valued streaming sensor data. Some stream mining complexities we addressed in this problem domain include concept drift [18] and forecasting ability [20]. Most data stream mining techniques use a sliding window [18] to train models for immediate prediction. Since simulation models must provide predictions across operating regions not currently in use, sliding windows can fail in preserving the necessary data to build accurate global models. Sliding windows are very effective for predicting near-term outputs, but ISS EPS power analysis must forecast weeks into the future. In this case, hardware devices can operate in drastically different modes.

As previously mentioned, the model mixture approach is similar to the cascading ensemble technique [17]. In this case, model confidence is based on the Euclidian distance from the nearest training set data point. The motivating factor in model mixture is to bind the amount of extrapolation error. We want to reduce the probability of any unacceptable spikes in predictive error.

Others have worked towards a similar goal of adapting modeling for simulation, but with a trivial discrete-valued problem and no data mining techniques [10]. Some have proposed the need for machine learning and data mining for the ISS EPS [13], but there are no corresponding research and experiment results.

Regarding spacecraft modeling, researchers have used ANNs to build simple spacecraft models [6], but flight engineers do not actively use these models. This thesis addresses a

more complex, predictive model. To date, no one has addressed the ISS electrical power system.

Attempts have been made to fuse models [11] or incorporate background knowledge [12], but not in the same fashion as proposed in this thesis. The work in [11] fused two ANNs [11]; one static and one actively updated. In the approach presented in this thesis, we fused the engineering model with a learned model.

Instead of incorporating background knowledge by using a mathematic representation, [12] uses a rules-based approach for incorporating background knowledge into an expert system. We could incorporate a similar approach where engineers define explicit mathematic equations for specific operating regions.

7.0 Conclusions and Future Work

7.1 Conclusions

Current spacecraft system modeling and simulation approaches require extensive domain knowledge and expensive calibration effort in order to build and maintain accurate models. We have proposed a new machine learning approach to automatically and continuously adapt systems models to accurately reflect current system behavior. Our approach includes new novel techniques for handling problems specific to the system modeling and simulation domain: online clustering for retaining representative training examples across the system operating range and model fusion for harnessing mathematical background knowledge when faced with newly explored scenarios.

We have performed complete and robust experiments using real ISS sensor data that demonstrate the success of our approach. Accuracy improvements result from 80% to 96% error reductions over existing techniques. Due to the focus on predictive stability and the experimental results, the NASA ISS electrical systems flight control community is requesting an operational prototype for MCC mission planning and analysis.

7.2 Future Work

Operational NASA Usage – Our next major step is to incorporate this adaptive system modeling approach into a NASA MCC software application currently used for ISS mission support. Flight controllers have already expressed a high degree of interest in our approach and are prepared to perform full evaluations. We plan to perform thorough evaluations using years of historical ISS data and then develop an integrated prototype during the summer of 2007. If the flight controllers accept our software, the new system is in a position to replace

existing tools as the primary ISS power planning platform. We expect to publish positive quantitative and qualitative results from this endeavor by the end of 2007.

Incremental Learning – The online techniques described in this thesis use traditional batch learning methods to training on an evolving representative data set. Each training session results in a completely new model. In contrast, incremental algorithms can train on each incoming data point to adapt the underlying model in real-time. While we plan to explore incremental models, we expect that models will forget unused operating regions very quickly. Such behavior is unacceptable for simulation models since they must be capable of predicting outputs for the full operating range. Flight planners need to predict ISS power weeks in advance, not just the immediate future.

Enhanced Representative Data Set Maintenance – The k-means clustering algorithm successfully maintains an evolving data set across the system operating range. However there is room for improvement. An incremental clustering strategy (e.g. online k-means [16]) could drastically improve run-time performance by only adjusting local clusters.

The k-means approach we used in this research also retains noise points as individual clusters. Most learning methods we used in this experiment provide some level of noise immunity. Due to the success of our results, we can conclude that this noise did not significantly impact model accuracy. We can use a clustering technique less susceptible to noise to help construct models that are even more accurate.

Finally, the introduction of weighting factors will allow for age-based discounting of data point influence. In some systems, we want the ability to determine how stale training

data can become. An age-based discounting scheme would either remove or reduce the influence of elder data points.

Online Feature Selection – Feature selection was the only portion of the data mining process the system does not perform in an online fashion. Using a similar approach as algorithm selection (Section 3.3.3) and model mixture threshold selection (Section 3.5.3), the system could perform automated feature selection on a periodic basis. The advantage of such an approach is the inclusion or exclusion of a feature due to concept drift. The current approach assumes that the same feature set will produce the most accurate model across the lifetime of the system.

The potential drawbacks of online feature selection include increased training time and potential reduction in stability due to variation or completely adding and deleting features instantaneously. The exclusion or inclusive of new features can have a major impact on training effectiveness.

Online Algorithm Selection Enhancements – Currently the system only analyzes the error rate during online algorithm selection. We can include other factors such as the prediction runtime performance. Runtime performance is a limiting factor in most simulation systems. Other potential metrics include:

- Maximum error (drastically affects the user's impressions of the system)
- Error variance (also affects the user's impressions of the system)
- Training time (system responsiveness to concept drift)
- Algorithm stability (constant value determined a priori)

- Extrapolation behavior (constant value determined a priori)

The system would use a weighted linear combination of these metrics to perform algorithm selection.

In addition, in some simulation systems, users wish to avoid over-prediction. In these cases, the user always prefers the conservative answer for safety reasons. Cost sensitive evaluation techniques would analyze over-prediction versus under-prediction when selecting algorithms. We could also use similar techniques to bias algorithms into learning conservative models by incorporating cost sensitive factors into the error function.

Application to Other System Models – If flight engineers adopt this adaptive systems modeling approach for EPS mission support, NASA can apply the approach to other ISS systems or new lunar exploration systems such as the Crew Exploration Vehicle (CEV) systems. Effort in applying the adaptive system modeling and simulation techniques would test the generic applicability of the approach outlined in this research. In addition, this technique is not specific to spacecraft systems. Other potential domains include aeronautics, ground vehicles, and industrial plants.

Runtime Architecture and Tool Support – We believe an integrated software architecture would be the primary method for documenting and injecting adaptive system modeling technology. A corresponding software toolkit would assist engineers when constructing adaptive models for existing and future systems. The software architecture must interface with real-time sensor streams and provide run-time coordination between the adaptive models and engineering models. Existing simulation systems might impose requirements for run-time interoperability within higher-level systems. The toolkit must

support these interfaces for adaptive simulations to participate in such higher-level systems such as the High Level Architecture (HLA) [14]. HLA is the IEEE standard for Modeling and Simulation high-level architectures primary used by the Department of Defense.

References

[1] Jannette, A.G., J. S. Hojnicki, D. B. McKissock, J. Fincannon, T. W. Kerslake, and C. D. Rodriguez. 2002. Validation of international space station electrical performance model via on-orbit telemetry. In *Proceedings of the 2002 Energy Conversion Engineering Conference* 45-50.

[2] Gietl, E. B., E. W. Gholdson., B. A. Manners, and R. A. Delventhal. 2000. The electric power system of the international space station - a platform for power technology development. NASA TM-2000-210209.

[3] NASA. *NASA Exploration Systems Architecture Study*. http://www.nasa.gov/mission_pages/constellation/news/ESAS_report.html.

[4] Bernard, D., et al. 1999. Spacecraft autonomy flight experience - the DS1 remote agent experiment. In *Proceedings of the 1999 AIAA Space Technology Conference and Exposition* AIAA-1999-4512.

[5] Chien, S., et al. 2005. Using autonomy flight software to improve science return on earth observing one. *AIAA Journal of Aerospace Computing, Information, and Communication* 2(2):196-216.

[6] Saravanan, N., A. Duyar, T. H. Guo, and W. C. Merrill. 1994. Modeling space shuttle main engine using feed-forward neural networks. *AIAA Journal of Guidance, Control, and Dynamics* 17(4):641-48.

[7] Zhang, G., B. Patuwo, and M. Hu. 1998. Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting* 14(1):35-62.

[8] Rumelhart, D., G. Hinton, and R. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323(6088):533-36.

[9] Witten, I. H., and E. Frank. 2005. *Data mining: practical machine learning tools and techniques*. 2nd ed. San Francisco: Morgan Kaufmann.

[10] Bosch, P.C., and M. Rajab. 2004. Autonomous predictive-adaptive simulation for operations support. In *Proceedings of the 2004 Winter Simulation Conference* 2: 2018-24.

[11] Zaknich, A., Y. Attikiouzel. 1997. A fast adaptive neural network system for intelligent control. In *Proceedings of the 1997 IEEE Conference on Systems, Man, and Cybernetics* 2:1023-27.

[12] Huang, S. H., R. Kothamasu, Y. C. Shiralkar, and D. Bogstad. 2003. Prediction of plastic preform temperature profile and modeling perspective. *International Journal of Manufacturing Science and Technology* 4(2):56-83.

[13] Bay S., P. Langley, M. W. Marker, J. Sanchez, and D. Shapiro. *Monitoring and modeling the space station electrical power system*. California: Stanford University. http://cll.stanford.edu/research/previous/monitoring/.

[14] IEEE Computer Society. 2000. IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules. IEEE Std 1516-2000.

[15] MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1:281-97. Berkeley, CA: University of California Press.

[16] Alpaydin, E. 2004. *Introduction to machine learning*. Cambridge, MA: The MIT Press.

[17] Kaynak, C., and E. Alpaydin. 2000. Multistage cascading of multiple classifiers: one man's noise is another man's data. In *Proceedings of the Seventeenth International Conference on Machine Learning* 455-62.

[18] Widmer, G., and M. Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* 23(1):69-101.

[19] Babcock, B., S. Babu, M. Datar, R. Motwani, and J. Widom. 2002. Models and issues in data stream systems. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* 1-16. New York: ACM Press.

[20] Gershenfeld, N. A., and A. S. Weigend. 1993. The future of time series. In *Time series prediction: forecasting the future and understanding the past*, ed. A. S. Weigend, and N. A. Gershenfeld, 1-70. Reading, MA: Addison-Wesley.

[21] Boryczka, U. 2006. Finding groups in data: cluster analysis with ants. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications* 1(October):404-9. Los Alamitos, CA: IEEE Computer Society.

[22] Freund, Y., and R. E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1):119-39.

[23] Han, J., and M. Kamber. 2000. *Data mining: concepts and techniques*. 2nd ed. New York: Morgan-Kaufman.

[24] Kenney, J. F., and E. S. Keeping. 1951. *Mathematics of statistics, Pt. 2*. 2nd ed. Princeton, NJ: Van Nostrand.

[25] Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. 1989. Phoneme recognition using time-delay neural networks. IEEE Transactions On Acoustics, Speech, and Signal Processing 37(3):328-39.

Acronyms

ANN	Artificial Neural Network
ARMA	Auto-Regressive Moving Average
BCDU	Battery Charge Discharge Unit
DM	Data Mining
EPS	Electrical Power System
GN&C	Guidance, Navigation, & Control
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
ISS	International Space Station
kNN	k Nearest Neighbors
LR	Linear Regression
MCC	Mission Control Cetner
ML	Machine Learning
NASA	National Aeronautics and Space Administration
REP	Reduced Error Pruning
SVM	Support Vector Machines
Weka	Waikato Environment for Knowledge Analysis