Date: 05/08/2016

From: Hendrik Brutsaert, Yuan Gao, Yosub Rim, and Yunfei Xie

hb31@cornell.edu, yg373@cornell.edu, yr96@cornell.edu, yx343@cornell.edu

To: Dr. X. Yang & Prof. S. Volgushev

Department of Statistical Science

Cornell University

Project Client: Dr. R. Martin

Project Advisor: Prof. S. Volgushev

Research Field: Adverse Event Prediction

Project Title: Using NASA ACCEPT (Adverse Condition and Critical Event Prediction Toolbox) for Adverse Thermal Events in a Smart Building

1. Project Description

1-1. Abstract:

Smart buildings strive to create an optimal environment for occupants through the active management of lighting, temperature, water, airflow, security, sanitation, and electricity through multiple connected systems and devices which interact dynamically using integrated algorithms and sensors. Sometimes there are discrepancies between occupants' real thermal comfort and measurements from sensors. Thus thermal discomfort is one of the highest priorities when designing and implementing these algorithms.

1-2. Background:

NASA Ames Sustainability Base (SB) is a smart and green building that provides researches for sustainable technologies and concepts. It is one the greenest constructions that built by the government (Hull, 2012). With the pleasant location in Santa Clara Valley, San Francisco Bay, the building is created to be "native to place" ("Sustainability Base", 2012). Using the system of nature sources (the sun's arc and wind power from the Bay) and on-site electricity generation, the building can generate more electricity than it actually consumes and also provide a comfortable environment for the residents ("Energy Dieting", 2015). There are many advanced technologies included in the building. It is equipped with 2636 thermal sensors inside the building, which perform physical or logical measurements. The building is run using state-of-the-art Siemens Apogee Insight software and detections systems.

About the air conditioning system, the building does not install any air conditioners. It mainly uses circulation of cold water in the copper pipes embedded in the ceiling to maintain the cool environment (Hull, 2012). Starting from 2012, about 200 occupants are working in the building (Hull, 2012). NASA cares about their physical health and emotional feelings when these employees are working in the building. From November 2014 to May 2015, occupants in SB proposed many “cold complaints”. Simply speaking, “cold complaints” were originated from an unexpected cool environment in the rooms or an abnormal drop in temperature in the building.

This paper introduces novel techniques using a generic framework developed in Matlab by NASA called Adverse Condition and Critical Event Prediction Toolbox (ACCEPT). The paper analyzes data types of the NASA Sustainability Base smart building, discusses variable selection techniques for the ACCEPT algorithm, and goes into the specific algorithms and features of ACCEPT which will minimize thermal discomfort adverse events.

1-3. Project Goal:

Our objective is to compare the performance of different machine learning and early warning detection models produced by ACCEPT upon predicting thermal adverse events.

2. ACCEPT

2-1. ACCEPT Architecture:

The prediction of anomalies or adverse events is a challenging task, and there are a variety of methods which can be used to address the problem. ACCEPT (Adverse Condition and Critical Event Prediction Toolbox) is an architectural framework designed to compare and contrast the performance of a variety of machine learning and early warning algorithms, and tests the capability of these algorithms to robustly predict the onset of adverse events in any time-series data generating systems or processes (Martin et al. 2015a).

Accept borrows ideas from MSET (multivariate state estimation technique) methodology (Bickford 2000). This was originally used for nuclear as well as aviation and space application, and later on, it was also used for adverse event prediction by NASA. This architecture consists of two toolboxes: the regression toolbox and the detection toolbox. The regression toolbox initially creates a model using various different machine learning algorithms. The detection algorithm then uses a Kalman filter and ROC curve analysis and different alarm systems to predict adverse events.

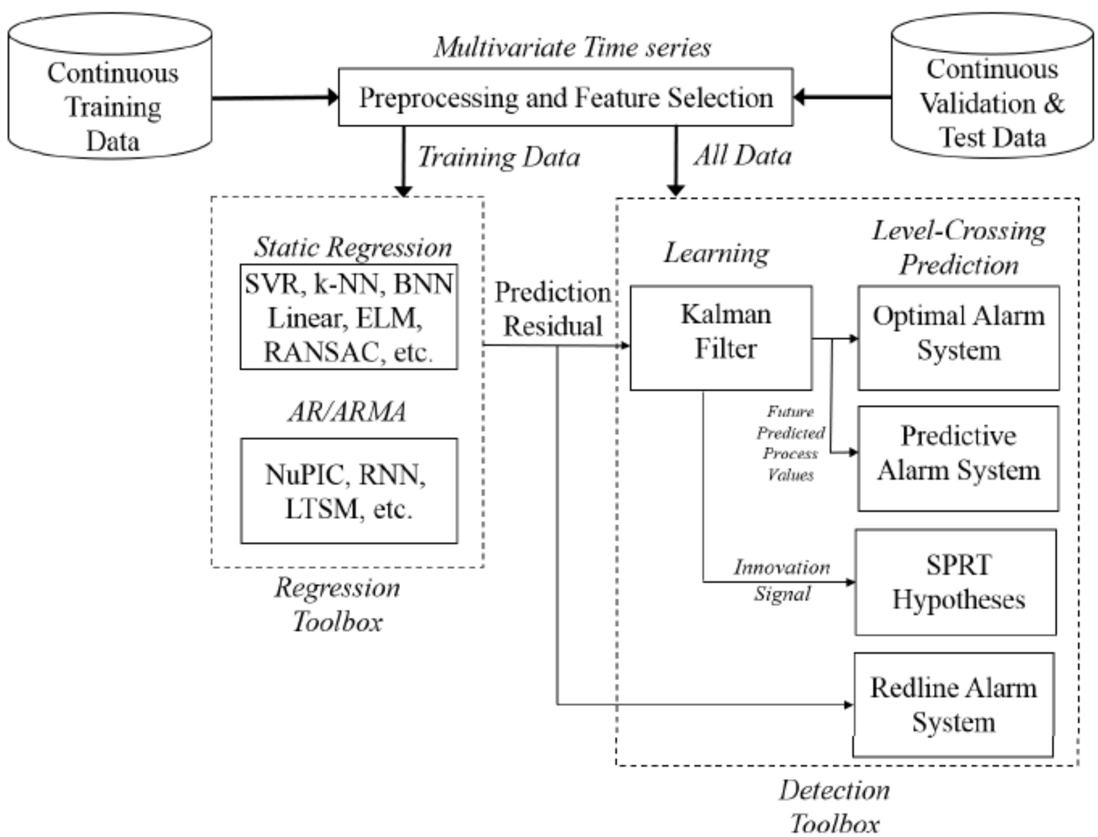


Figure-2.1: ACCEPT Process Flow Diagram

Figure-1 shows an explicit diagram showing the general work-flow of ACCEPT.

There are some statistical concepts which may may need to be elucidated:

- Feature Selection (FS): also known as variable selection or attribute selection, a process of selecting relevant variables from the big picture of raw data for statistical modeling construction.
- Kalman Filter (KF): a statistical algorithm that uses time-based measurements and observations, including statistical noise, and to produce estimation of unknown variables.
- ROC Curve: receiver operating characteristics, which demonstrates the performance of a specific binary classifier when its threshold varies. The x and y axises are the true positive rate and the false positive rate at various threshold settings.
- Linear Dynamical System (LDS): dynamical systems whose evaluating function is linear. Unlike dynamical systems, the root of LDS can be exactly solved. Also, it is used to illustrate the properties of dynamical systems by solving the equilibrium points of the system.

In the regression step, after choosing a specific regression model, ACCEPT produces residual output that quantifies the difference between the actual value of the target parameter and the value predicted by regression model. The regression performance is represented and quantified by the Normalized Mean Square Error (NMSE) of residuals, and it also acts as a preprocessor for the transformation of data into a form that is amenable to modeling.

In the detection step, all detection methods will conform to a process which is based on the occurrence of adverse events contained in the validation datasets. Then in the design of the alarm system and final testing procedure with the testing datasets, in theory, all the adverse events should be drawn from the same distribution as the validation datasets.

2-2. Regression Toolbox:

ACCEPT provides different regression to be tested in the context of a rigorous analysis that uses nominal training data which is partitioned into several folds for the purpose of f-fold cross-validation analysis.

In our case, we only consider 3 regression methods, although ACCEPT can test more than these:

- Regularized Linear Regression (LR): Tikhonov regularization, which seeks to determine a useful approximation of regularization of ill-posed problems, is used and the regularization coefficient acts as the hyperparameter to optimize the NMSE.
- Extreme Learning Machine (ELM): ELM has a basic structure similar to that of a single layer feedforward neural network, yet the input layer parameters are assigned in a random sense, thus the only unknown parameters are the output layer parameters of the model, and can be determined solely by linear least square approach. The hyperparameter is the number of hidden neurons.
- Random Sample Consensus (RANSAC): in some cases, assumptions of typical regression methods do not hold, and a misleading result may occur if still using such regression approaches. RANSAC now becomes a robust regression method

that is not excessively affected when gross errors and extreme outliers exist in the data. It uses a hypothesis testing methodology to eliminate the effect of such gross errors and outliers towards the regression method. The hyperparameter is cost threshold.

2-3. Detection Toolbox:

Distinct from regression, all detection methods conform to a rigorous validation process that consider both nominal training data and the validation data containing adverse events as shown in Figure 1. There are two main detection techniques:

1. Methods that use a Monte-Carlo style implementation to empirically generate relevant alarm system statistics.
2. Methods that rely on a model-based approach to generate performance metrics.

During the detection stage, ROC curve analysis is used to design the tradeoff analysis between false alarm rate and missed detection rate. The resulting threshold from the ROC analysis will be used to implement the detection method using the testing datasets.

There are 3 different detection methods combined with training-base alarm and validation-base alarm, so we have 6 different detection method/alarm system to use:

- Redline w/ Training-base (RT)
- Redline w/ Validation-base (RV)
- Predictive w/ Training-base (PT)
- Predictive w/ Validation-base (PV)
- Optimal w/ Training-base (OT)
- Optimal w/ Validation-base (OV)

2-4. Model Comparison Standards:

After all work done by ACCEPT, 3 key rates will be exported and they are used to compare the performance of each combination of regression/detection method to predict adverse event. They are listed by formal definitions:

- False Alarm Rate (FAR): an alarm is triggered at a time point that does not contain an example of a confirmed anomalous event in at least one time point in the next d time steps.
- Missed Detection Rate (MDR): no alarm is triggered at a time point where an example of a confirmed anomalous event exists in at least one time point in the next d time steps.
- Detection Time (DT): d time steps prior to the occurrence of a future adverse event, which is detected by the prediction system.

NOTICE: ACCEPT is a high-level integrated packages developed by scholars. Throughout the whole process of doing this project, our team did not try to fully understand the specific and detailed methodology behind ACCEPT, instead, what more important is the use and application of this technical toolbox, and to explore models which produces better results upon prediction adverse events.

3. Data Manipulation

3-1. Sustainability Base Dataset:

The sustainability base dataset consists of over 2544 variables. These variables represent both the output of various sensors and as well as user and system determined inputs. This environment lays a predictive groundwork which can be used to optimize occupant environment using state-of-the-art predictive algorithms for adverse events.

There are 7 variable type categories in the dataset and the distribution of them is shown in Figure-2:

1. A floating point precision variable describing a user or system inputted value to set parameters for one of the many systems in SB.
2. Analog Output- A floating point precision variable describing the output of a system within SB.
3. Analog Value- A floating point precision variable describing the state of a system within SB.
4. Binary Input- A binary variable describing a user or system inputted value to set parameters for one of the many systems in SB.
5. Binary Output- a binary value describing the output of a system within SB.
6. Binary Value- A binary variable describing the state of a system within SB.
7. Multistate value- A multi-state variable describing the state of a system within SB.

From these data types the variables relevant to the ACCEPT framework are those of analog values only (unless part of a variable transformation process).

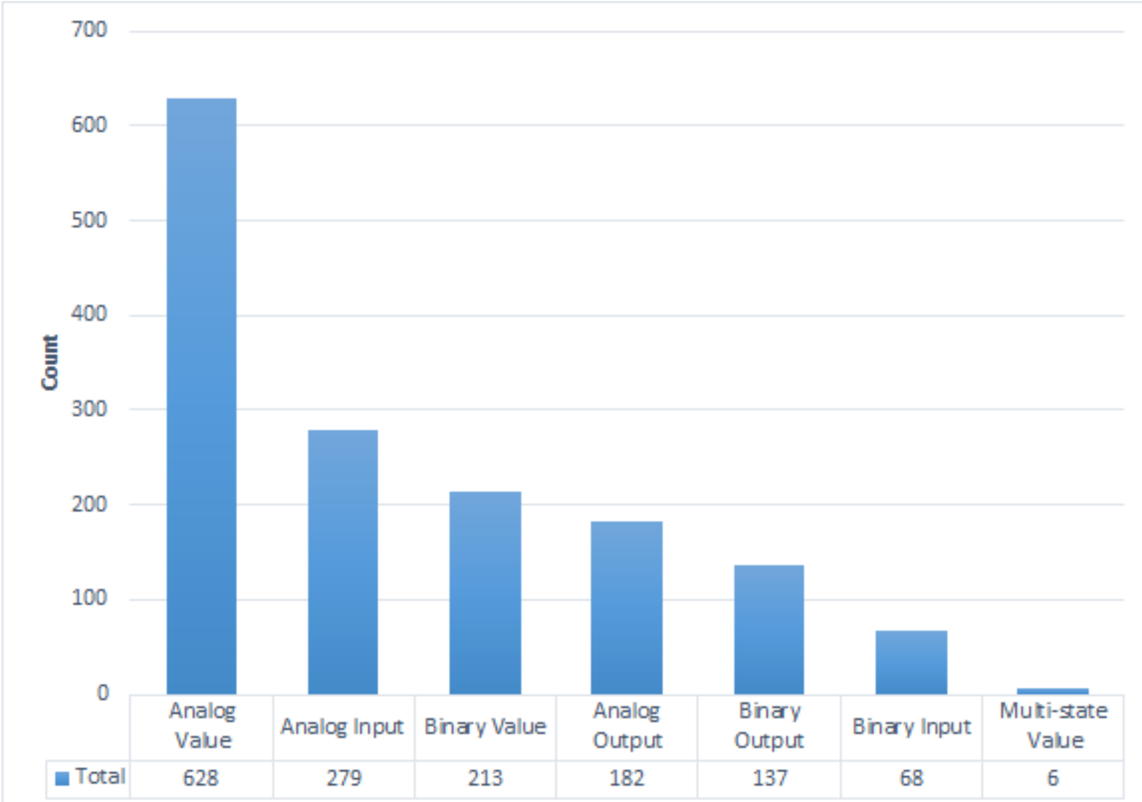


Figure-3.1: Types of variables in the SB Building 232 Dataset

3-2. Random Feature Set:

The Random Feature set consists of 10 predictors named A to J, which were chosen at random from the SB Building 232 dataset. These variables were selected at random and represent a variety of different variable types. From Figure 3.2, it is clear that B, D, I, J, and response variables are non-normally distributed whereas the others are.

Index	Description	Object Type
A	232 RF1 HWS VALVE 14	Binary Output
B	232 A1 DX CAP SIGNAL	Analog Output
C	232 RSB P1 START/STOP	Binary Output
D	232 CRCP VALVE S28A	Analog Output
E	232 GWRV LOOPOUT	Analog Value
F	232 M1 AVG FLOW	Analog Value
G	232 ZONE N121 N125 AVERAGE TEM	Analog Value
H	232 S1 DPT AVG C	Analog Value
I	232 HP3 HEAT STAGE TIMER	Analog Value
J	232 N1 COOLING OFF	Analog Value

Figure-3.2: Feature Variable Types

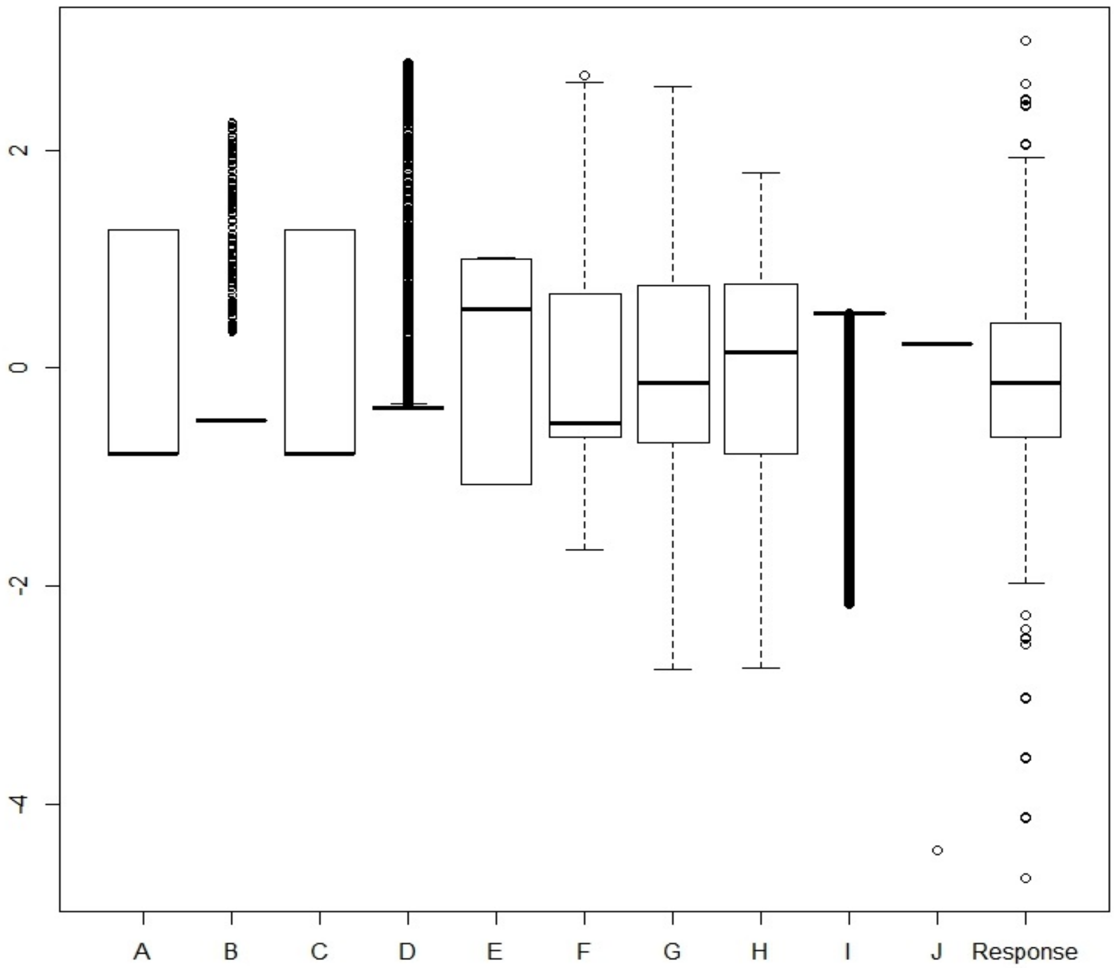


Figure-3.3: Box Plot of Scaled Features and Response for Random Dataset

3-4. Response Variable:

Our response is the "Building 232 Zone N240 room temperature" as measured in degrees Fahrenheit, named DCTN240T. This is an analog input variable, a floating point precision variable describing a user or system inputted value to set temperature parameters for one of the many systems.

Index	Description	Object Type
K	232 ZONE N240 ROOM TEMP	Analog Input

Figure-3.4: Response Variable Type

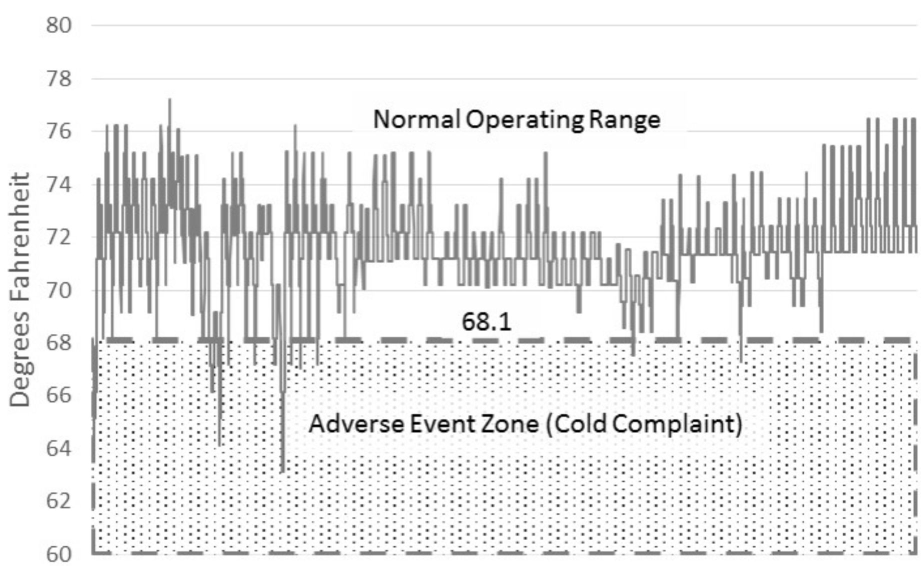


Figure-3.5: Response Variable: Building 232 Zone N240 room temperature, shaded area captures adverse event region

3-5. Project Scope:

The project scope was 92 calendar days beginning on 03 November, 2014 at midnight and ending on 02 February, 2015 at 11:45 pm. The days were split up into 5 minute intervals. Only Testing and Validation Data had adverse events as per ACCEPT adverse event prediction paradigm. There were 3 missing 5 minute interval data points which were extrapolated from the point before. The data was split into Training/Testing/Validation Datasets according to the following splits:

- Testing Data- Days 14 and 15.

- Validation Data- Days 1, 16, 19, 21, 22, 23, 24, 26, 61, and 73.
- Training Data- All other days (no adverse events).

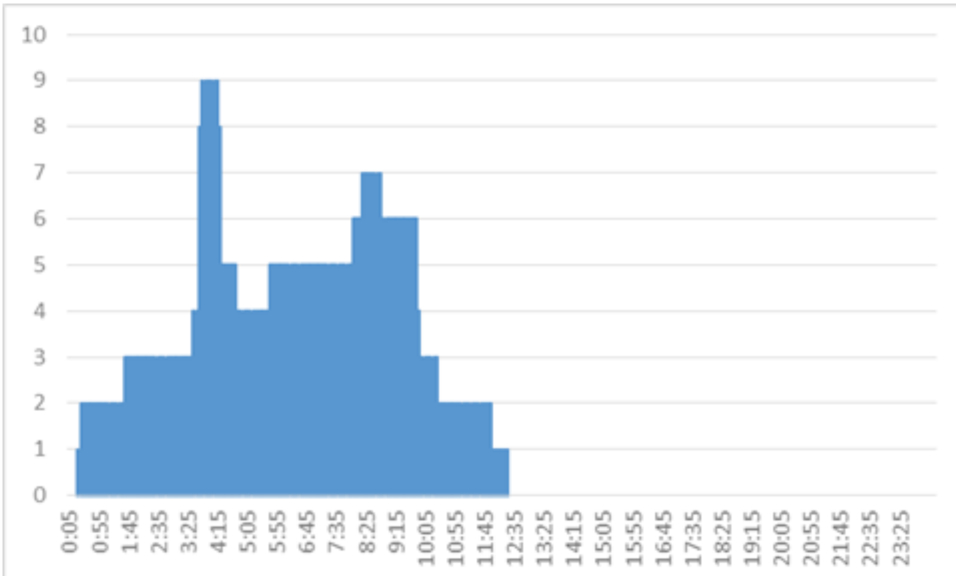


Figure-3.6: Empirical Distribution of adverse event count for all 92 days by time of day, notice all adverse events occur before 12:15

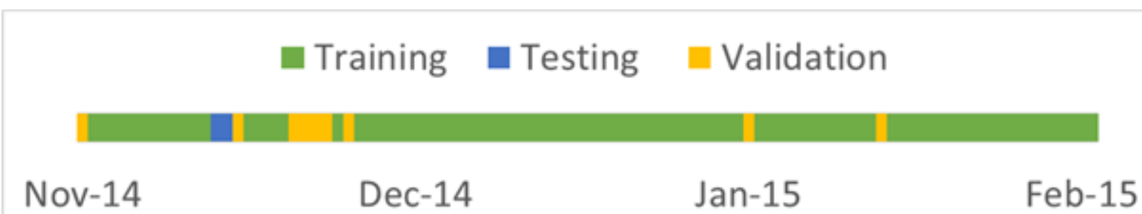


Figure-3.7: Data Split. 80 days for Training, 10 Days for Validation, 2 Days for Testing

3-6. Adverse Event Definition:

An adverse event for the Thermal Discomfort Problem is defined as any 5 minute temperature reading which registers below 68.1 degrees Fahrenheit.

An empirical approach was taken to determine the ground truth for the cold complaints prediction scenario. We estimated the distribution of the target room temperature sensor

(x) and found that it was a unimodal distribution with mean 71.7 and standard deviation 1.8. Hence, a 95% confidence interval around the mean corresponds to 68.1F and 75.3F. Considering this range as nominal room temperature values, we established 68.1F as the upper threshold for cold regions. In our problem, we are only concerned with anomalous drops in temperature. Thus, we considered any temperature value below 68.1F as an adverse event (cold).

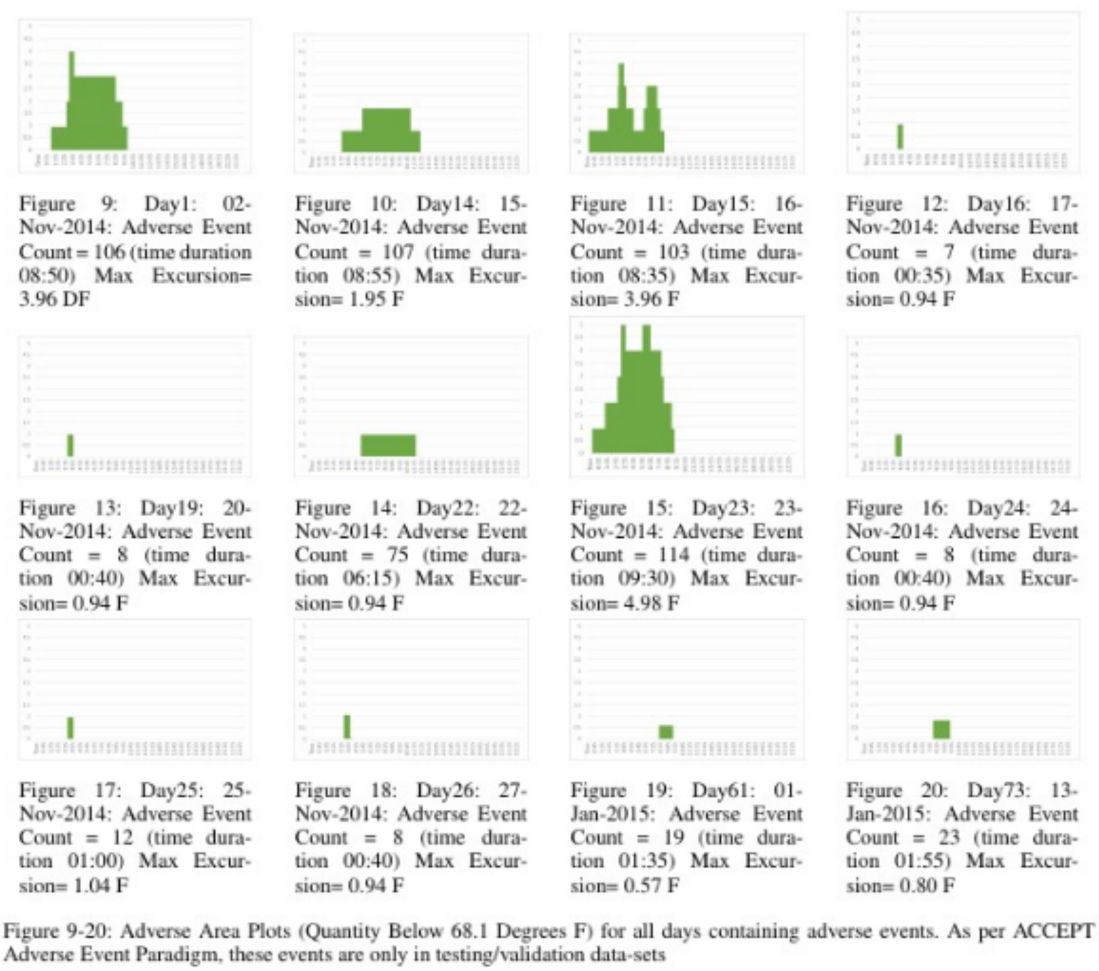


Figure 9: Day1: 02-Nov-2014: Adverse Event Count = 106 (time duration 08:50) Max Excursion= 3.96 DF

Figure 10: Day14: 15-Nov-2014: Adverse Event Count = 107 (time duration 08:55) Max Excursion= 1.95 F

Figure 11: Day15: 16-Nov-2014: Adverse Event Count = 103 (time duration 08:35) Max Excursion= 3.96 F

Figure 12: Day16: 17-Nov-2014: Adverse Event Count = 7 (time duration 00:35) Max Excursion= 0.94 F

Figure 13: Day19: 20-Nov-2014: Adverse Event Count = 8 (time duration 00:40) Max Excursion= 0.94 F

Figure 14: Day22: 22-Nov-2014: Adverse Event Count = 75 (time duration 06:15) Max Excursion= 0.94 F

Figure 15: Day23: 23-Nov-2014: Adverse Event Count = 114 (time duration 09:30) Max Excursion= 4.98 F

Figure 16: Day24: 24-Nov-2014: Adverse Event Count = 8 (time duration 00:40) Max Excursion= 0.94 F

Figure 17: Day25: 25-Nov-2014: Adverse Event Count = 12 (time duration 01:00) Max Excursion= 1.04 F

Figure 18: Day26: 27-Nov-2014: Adverse Event Count = 8 (time duration 00:40) Max Excursion= 0.94 F

Figure 19: Day61: 01-Jan-2015: Adverse Event Count = 19 (time duration 01:35) Max Excursion= 0.57 F

Figure 20: Day73: 13-Jan-2015: Adverse Event Count = 23 (time duration 01:55) Max Excursion= 0.80 F

Figure 9-20: Adverse Area Plots (Quantity Below 68.1 Degrees F) for all days containing adverse events. As per ACCEPT Adverse Event Paradigm, these events are only in testing/validation data-sets

Figure 3.8: Adverse Area Plots (Quantity Below 68.1 Degrees F) for all days containing adverse events

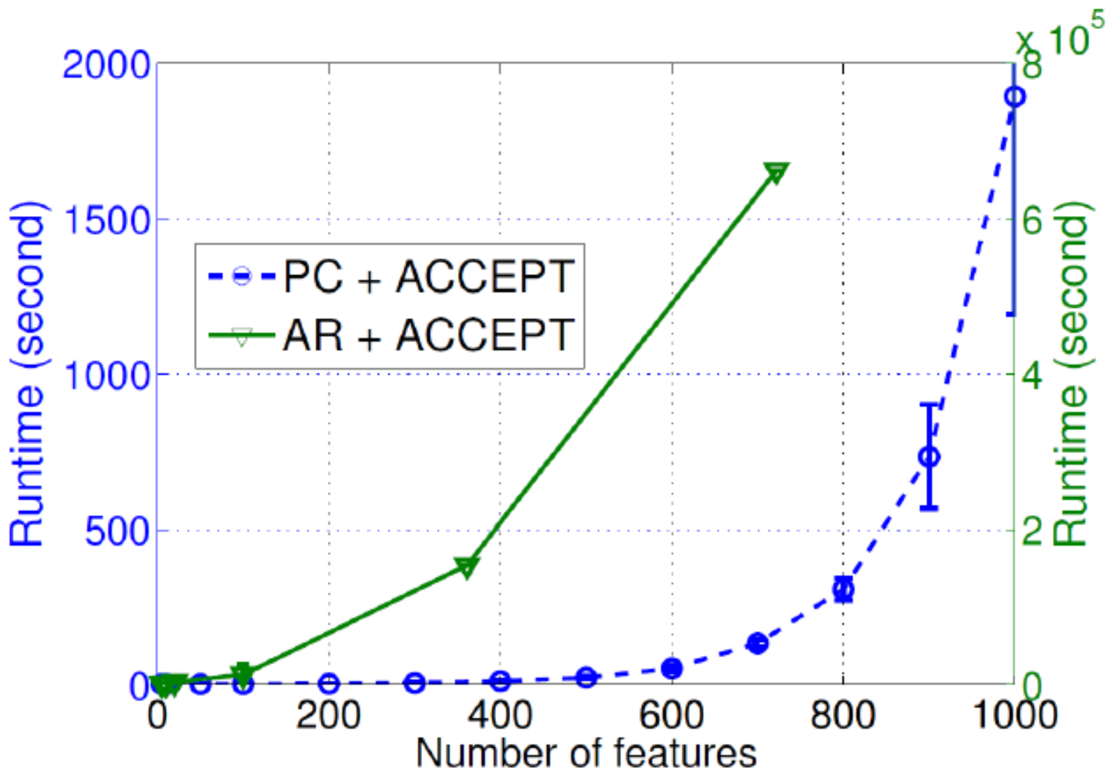


Figure 3.9: Analysis of ACCEPT Runtimes as per (Martin et al. 2015c)

3-7. Variable Reduction:

As described by (Martin et al. 2015c) at the 2015 AAAI conference, the ACCEPT algorithm takes exponential longer using a larger number of variables (see figure 3.9). To combat this, (Martin et al. 2015c) proposed as main variable reduction algorithm Lasso regression, namely

$$\hat{\beta}_L = \underset{\beta\in\mathbb{R}^{FT}}{argmin} \sum_{T=\tau}^{T} Y_{t-1,t-\tau} + \lambda||\beta||_L$$

However for this task we used stepwise regression in order be able to use a number of different variable counts with one iteration of the algorithm. This algorithm is described in (James et al. 2013, chap. 6.1.3):

Algorithm 1: Forward stepwise selection

Let M_0 denote the *null* model, which contains no predictors. ;

for $k = 0, \ldots, p-1$ **do**

  Consider all $p-k$ models that augment the predictors in M_k with one additional predictor. ;

  Choose the *best* among these $p-k$ models and call it M_{k+1}. Here *best* is defined as having the smallest RSS or highest R^2. ;

Select a single best model from among $M_0, \ldots, M_p$ using cross validated prediction error, C_p (AIC), BIC, or adjusted R^2. ;

4. Execution & Results:

We will demonstrate and compare the performance of different models (LR, ELM, RANSAC) in ACCEPT through 4 main steps:

1. Estimate hyperparameter for each regression methods.
2. Set alarm constraints and observe changes in performance.
3. Explore best performing methods and further refine the alarm design separately by training-base alarm and validation-base alarm.
4. Run all splits among val. & test sets and observe behaviour of alarm detection.

4-1. Step 1: Hyperparameter Optimization

Remember we have 3 regression methods (LR, ELM, RANSAC) and 6 detection methods (RT, RV, PT, PV, OT, OV).

ACCEPT Input for Step 1:

- Maximum state order = : 10

- What is the maximum design (and validation) prediction horizon ? : 12
- Enter resolution (number of points) for Monte Carlo-based integration (smoothness factor) : 3600
- Resolution of ROC curve (bits) : 10
- 1 - Closed form approximation, 2 - Root-Finding Approximation : 1

Regularized Linear Regression Output (LR)

	lin
Global Optimum	0.0385
Optimized Values	0.4506

Missed detection results...

	lin
Redline - Training	0.0000
Redline - Validation	0.0000
Predictive - Training	0.0000
Predictive - Validation	0.0000
Optimal - Training	0.0000
Optimal - Validation	0.0000

False alarm results...

	lin
Redline - Training	0.2176
Redline - Validation	0.2102
Predictive - Training	0.2176
Predictive - Validation	0.2102
Optimal - Training	0.2166
Optimal - Validation	0.2123

Detection time results...

	lin
Redline - Training	20.0000
Redline - Validation	20.0000
Predictive - Training	20.5000
Predictive - Validation	20.0000
Optimal - Training	20.5000
Optimal - Validation	20.0000

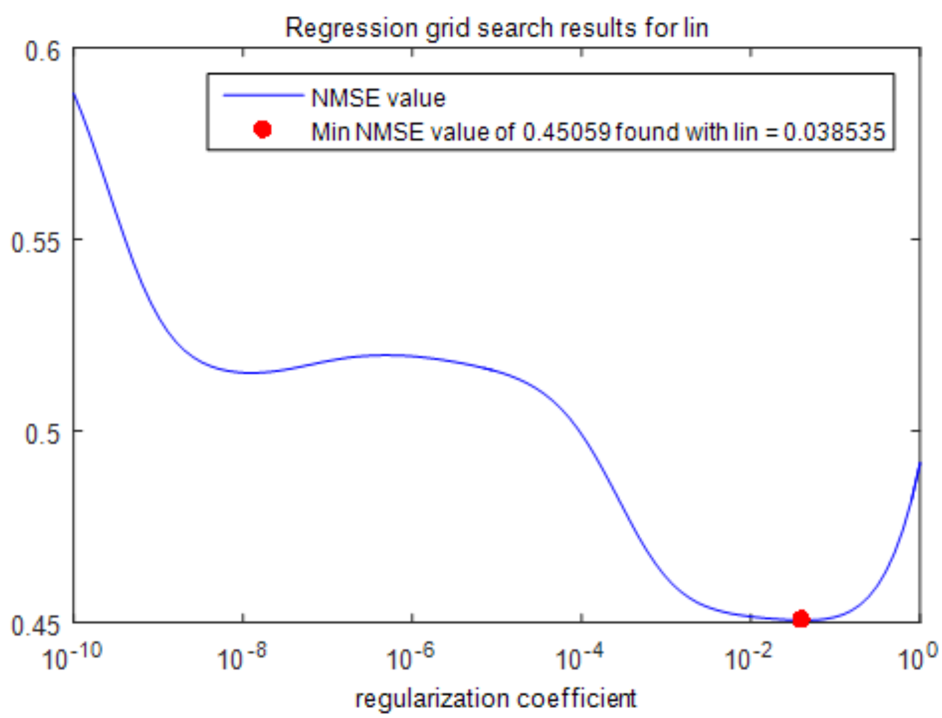


Figure - 4.1: Step 1 ACCEPT Output for LR

LR Interpretation 1: NMSE is minimized when the hyperparameter is at 0.038535. Since missed detection rate is 0, empirically we know that, alarm constraint on missed detection rate around 0.15 rather than trade off will give a bit higher missed detection rate but lower false alarm rate.

Extreme Learning Machine Output (ELM)

elm	
Global Optimum	146.0000
Optimized Values	0.4159
Missed detection results...	
elm	
Redline - Training	0.0000
Redline - Validation	0.0000
Predictive - Training	0.0000
Predictive - Validation	0.0000
Optimal - Training	0.0000
Optimal - Validation	0.0000
False alarm results...	
elm	
Redline - Training	0.2335
Redline - Validation	0.2335
Predictive - Training	0.2420
Predictive - Validation	0.2335
Optimal - Training	0.2420
Optimal - Validation	0.2335
Detection time results...	
elm	
Redline - Training	20.0000
Redline - Validation	20.0000
Predictive - Training	20.5000
Predictive - Validation	20.0000
Optimal - Training	20.5000

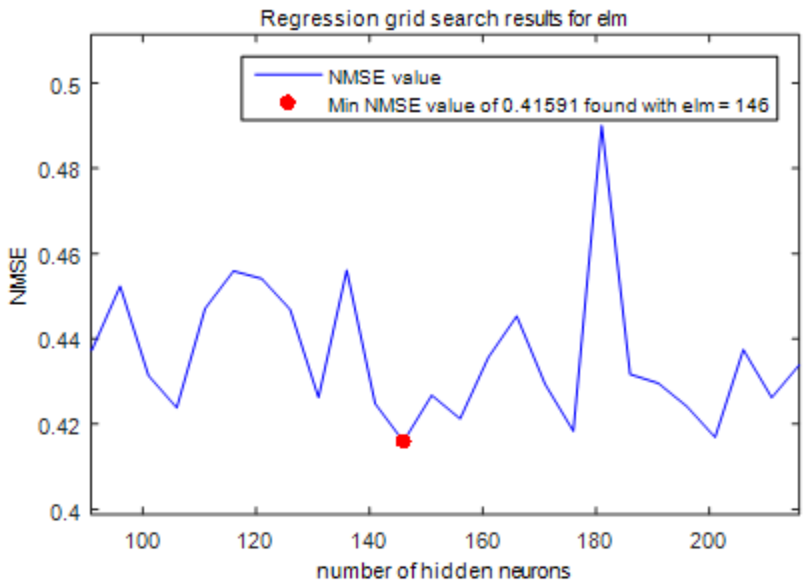


Figure - 4.2: Step 1 ACCEPT Output for ELM

ELM Interpretation 1: NMSE is minimized when the hyperparameter is at 146. Again, missed rate is 0. We will add alarm constraint on missed rate 0.15.

Random Sample Consensus Output (RANSAC)

ransac

Global Optimum	0.5554
Optimized Values	0.7972

Missed detection results...

	ransac
Redline - Training	0.0048
Redline - Validation	0.0714
Predictive - Training	0.0571
Predictive - Validation	0.0048
Optimal - Training	0.0333
Optimal - Validation	0.0048

False alarm results...

	ransac
Redline - Training	0.1953
Redline - Validation	0.1932
Predictive - Training	0.1921
Predictive - Validation	0.1943
Optimal - Training	0.1943
Optimal - Validation	0.1964

Detection time results...

	ransac
Redline - Training	20.0000
Redline - Validation	20.0000
Predictive - Training	10.5000
Predictive - Validation	10.5000
Optimal - Training	18.0000
Optimal - Validation	10.5000

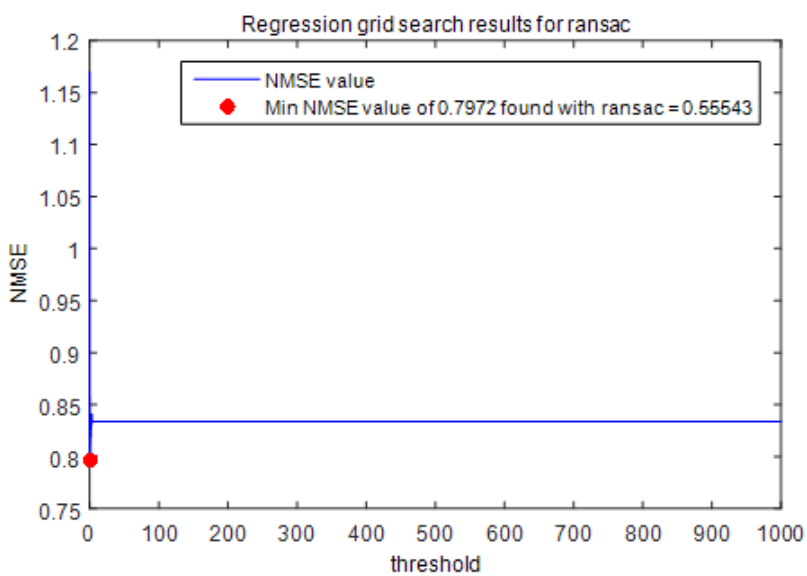


Figure - 4.3: Step 1 ACCEPT Output for RANSAC

RANSAC Interpretation 1: NMSE is minimized when the hyperparameter is at 0.55543. As in the case of Linear, it tends to have higher false rate and lower missed rate.

Summary from Step 1: optimized hyperparameters for LR, ELM, and RANSAC are 0.038535, 146, and 0.55543.

4-2. Step 2: Set Alarm Constraints:

LR Inputs and Outputs:

Design Alarm System by constraint on 1 - False Alarm Rate, 2 - Missed Detection Rate, 3 - Equal Tradeoff : 2

Enter max allowable missed detection rate : 0.15

Use ASOS approximation for LDS learning ? 1 - Yes, 2 - No : 2

Maximum state order = : 10

What is the maximum design (and validation) prediction horizon ? : 12

Enter resolution (number of points) for Monte Carlo-based integration (smoothness factor) : 3600

Resolution of ROC curve (bits) : 10

1 - Closed form approximation, 2 - Root-Finding Approximation : 1

lin	
Global Optimum	0.0385
Missed detection results...	
lin	
Redline - Training	0.0190
Redline - Validation	0.0714
Predictive - Training	0.0190
Predictive - Validation	0.0714
Optimal - Training	0.0190
Optimal - Validation	0.0714
False alarm results...	
lin	
Redline - Training	0.0520
Redline - Validation	0.0011
Predictive - Training	0.0403
Predictive - Validation	0.0011
Optimal - Training	0.0382
Optimal - Validation	0.0011
Detection time results...	

Figure 4.4: Step 2 ACCEPT Output for LM

LR Interpretation 2: easily one can observe now the training-base alarms perform well whereas validation-base alarms have detection time of zero. The LM model works well.

ELM Inputs and Outputs:

Design Alarm System by constraint on 1 - False Alarm Rate, 2 - Missed Detection Rate, 3 - Equal Tradeoff : 2

Enter max allowable missed detection rate : 0.15

Use ASOS approximation for LDS learning ? 1 - Yes, 2 - No : 2

Maximum state order = : 10

What is the maximum design (and validation) prediction horizon ? : 12

Enter resolution (number of points) for Monte Carlo-based integration (smoothness factor) : 3600

Resolution of ROC curve (bits) : 10

1 - Closed form approximation, 2 - Root-Finding Approximation : 1

elm	
Global Optimum	146.0000
Missed detection results...	
elm	
Redline - Training	0.0143
Redline - Validation	0.0143
Predictive - Training	0.0143
Predictive - Validation	0.0238
Optimal - Training	0.0143
Optimal - Validation	0.0190
False alarm results...	
elm	
Redline - Training	0.0743
Redline - Validation	0.0552
Predictive - Training	0.0764
Predictive - Validation	0.0446
Optimal - Training	0.0764
Optimal - Validation	0.0531
Detection time results...	
elm	
Redline - Training	10.5000
Redline - Validation	10.5000
Predictive - Training	10.5000
Predictive - Validation	10.5000
Optimal - Training	10.5000
Optimal - Validation	10.5000

Figure 4.5: Step 2 ACCEPT Output for ELM

ELM Interpretation 2: the output is surprising as it performed better than Linear case and we can see there could be possible improvement through adjusting max allowed missed detection rate. Note that the detection times almost halved comparing to the tradeoff case. As missed rate is still lower and false rate is a bit higher than that, we will give a bit higher max missed rate, 0.16 or 0.155.

RANSAC Inputs and Outputs:

Design Alarm System by constraint on 1 - False Alarm Rate, 2 - Missed Detection Rate, 3 - Equal Tradeoff : 2

Enter max allowable missed detection rate : 0.1

Use ASOS approximation for LDS learning ? 1 - Yes, 2 - No : 2

Maximum state order = : 10

What is the maximum design (and validation) prediction horizon ? : 12

Enter resolution (number of points) for Monte Carlo-based integration (smoothness factor) : 3600

Resolution of ROC curve (bits) : 10

1 - Closed form approximation, 2 - Root-Finding Approximation : 1

	ransac
Global Optimum	0.5554
Missed detection results...	
	ransac
Redline - Training	0.1333
Redline - Validation	0.1333
Predictive - Training	0.0905
Predictive - Validation	0.1333
Optimal - Training	0.1095
Optimal - Validation	0.1333
False alarm results...	
	ransac
Redline - Training	0.0902
Redline - Validation	0.0531
Predictive - Training	0.1051
Predictive - Validation	0.0552
Optimal - Training	0.0924
Optimal - Validation	0.0648
Detection time results...	
	ransac
Redline - Training	0.0000
Redline - Validation	0.0000
Predictive - Training	20.0000
Predictive - Validation	0.0000
Optimal - Training	0.0000
Optimal - Validation	0.0000

Figure 4.6: Step 2 ACCEPT Output for RANSAC

RANSAC Interpretation 2: we have some improvement of lowering the false rate but in return, we have too higher missed rate and detection time of zero. Thus we will not select RANSAC for Step 3 as it performs poorly.

Key points from Step 1 & 2:

- For LR, we observe lower FAR and higher level-crossing threshold from Step 1 to Step 2.
- For ELM, we observe the same change as LR case but it performs very well.
- For RANSAC, since it performs poorly, we decide not to go further with this regression method.

4-3. Step 3: Refine Alarm System by Separating Training-base and Validation-base:

Remember from Step 2, we have eliminated RANSAC mode from our analysis. Now we only have LR and ELM. However, in this step, we separately test the training-base and validation-base alarm, shown in Figure-4.7. Since we have determined the optimized hyperparameters from step 1, and the best inputs and the alarm constraints in step 2, we will not modify any inputs or hyperparameters from now on.

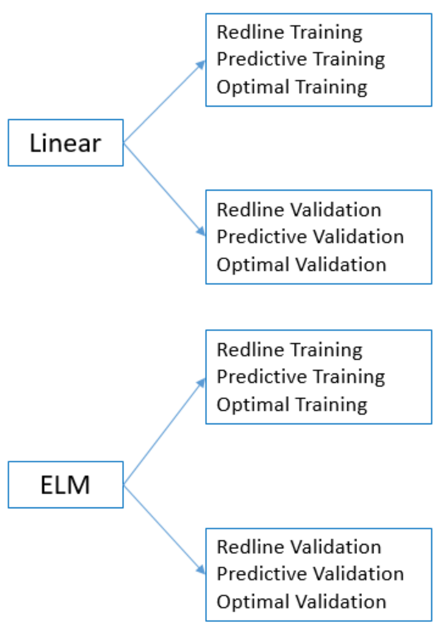


Figure 4.7: Step 3 Work Flow

LR w/ Training-Base Alarm Outputs:

	lin
Global Optimum	0.0385
Missed detection results...	
	lin
Redline - Training	0.0190
Predictive - Training	0.0190
Optimal - Training	0.0190
False alarm results...	
	lin
Redline - Training	0.0520
Predictive - Training	0.0350
Optimal - Training	0.0414
Detection time results...	
	lin
Redline - Training	1.0000
Predictive - Training	10.5000
Optimal - Training	10.5000

Figure 4.8: Step 3 ACCEPT Outputs for LR w/ Training

LR w/ Validation-Base Alarm Outputs:

	lin
Global Optimum	0.0385
Missed detection results...	
	lin
Redline - Validation	0.0714
Predictive - Validation	0.0714
Optimal - Validation	0.0714
False alarm results...	
	lin
Redline - Validation	0.0011
Predictive - Validation	0.0011
Optimal - Validation	0.0011
Detection time results...	
	lin
Redline - Validation	0.0000
Predictive - Validation	0.0000
Optimal - Validation	0.0000

Max allowed missed rate
0.15 -> 0.14

	lin
Global Optimum	0.0385
Missed detection results...	
	lin
Redline - Validation	0.0714
Predictive - Validation	0.0714
Optimal - Validation	0.0524
False alarm results...	
	lin
Redline - Validation	0.0011
Predictive - Validation	0.0011
Optimal - Validation	0.0011
Detection time results...	
	lin
Redline - Validation	0.0000
Predictive - Validation	0.0000
Optimal - Validation	0.0000

Figure 4.9: Step 3 ACCEPT Outputs for LR w/ Validation

LR Interpretation 3: Predictive Training is the best performing detection method. The result is satisfactory, so no further refinement will be made on LR w/ Training-base alarm model. On the other hand, LR w/ Validation-base alarm is not satisfactory, even though we change the maximum allowed MDR, the detection time is still 0, which does not necessarily make any predictions (remember the definition of MDR).

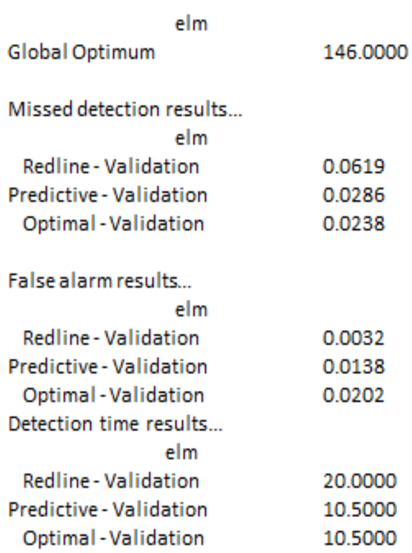
ELM w/ Training-Base Alarm Outputs:

	elm
Global Optimum	146.0000
Missed detection results...	
	elm
Redline - Training	0.0143
Predictive - Training	0.0143
Optimal - Training	0.0143
False alarm results...	
	elm
Redline - Training	0.0743
Predictive - Training	0.0764
Optimal - Training	0.0743
Detection time results...	
	elm
Redline - Training	10.5000
Predictive - Training	10.5000
Optimal - Training	10.5000

	elm
Global Optimum	146.0000
Missed detection results...	
	elm
Redline - Training	0.0143
Predictive - Training	0.0333
Optimal - Training	0.0238
False alarm results...	
	elm
Redline - Training	0.0552
Predictive - Training	0.0287
Optimal - Training	0.0414
Detection time results...	
	elm
Redline - Training	10.5000
Predictive - Training	10.5000
Optimal - Training	10.5000

Figure 4.10: Step 3 ACCEPT Outputs for ELM w/ Training

ELM w/ Validation-Base Alarm Outputs:



	elm
Global Optimum	146.0000
Missed detection results...	
	elm
Redline - Validation	0.0619
Predictive - Validation	0.0286
Optimal - Validation	0.0238
False alarm results...	
	elm
Redline - Validation	0.0032
Predictive - Validation	0.0138
Optimal - Validation	0.0202
Detection time results...	
	elm
Redline - Validation	20.0000
Predictive - Validation	10.5000
Optimal - Validation	10.5000

Figure 4.11: Step 3 ACCEPT Outputs for ELM w/ Validation

ELM Interpretation 3: we can see from the ELM w/ Training-base alarm, as we increase the maximum allowed MDR, the FAR decreases to the satisfactory level (under 0.05). On the other hand, ELM w/ Validation-base alarm produces satisfactory results.

Metric	Method	Lin
Missed Detection Results	Redline – T	0.0190
	Predictive – T	0.0190
	Optimal – T	0.0190
False Alarm Results	Redline – T	0.0520
	Predictive – T	0.0350
	Optimal – T	0.0414
Detection Time Results	Redline – T	1.0
	Predictive – T	10.5
	Optimal – T	10.5

Metric	Method	ELM
Missed Detection Results	Redline – T	0.0143
	Predictive – T	0.0333
	Optimal – T	0.0238
False Alarm Results	Redline – T	0.0552
	Predictive – T	0.0287
	Optimal – T	0.0414
Detection Time Results	Redline – T	10.5
	Predictive – T	10.5
	Optimal – T	10.5

Metric	Method	ELM
Missed Detection Results	Redline – V	0.0619
	Predictive – V	0.0286
	Optimal – V	0.0238
False Alarm Results	Redline – V	0.0032
	Predictive – V	0.0138
	Optimal – V	0.0202
Detection Time Results	Redline – V	20.0
	Predictive – V	10.5
	Optimal – V	10.5

Figure 4.12: Summary of ACCEPT Results from Step 3

Summary from Step 3: after considering models separately by training-base and validation-base, we produce some satisfactory results from LR w/ training and both the ELM models. Hence, these 3 models will keep moving to our last step.

4-4. Step 4: "Cross-Validation" Analysis:

Remember now we only consider 3 models, they are:

1. LR w/ Training-Base Alarm
2. ELM w/ Training-Base Alarm
3. ELM w/ Validation-Base Alarm

Also remember that the example of adverse events are only contained in validation and testing datasets. We are inspired that in order to further improve our results, one efficient way could be trying different validation and testing datasets, and by doing that, we produce 1 new split among the validation and testing sets. Then, what if we run all possible splits? Like the way how cross validation improves the power of the statistical model. Thinking further, we decide to split our given validation and testing datasets into a total of 66 different ways. Remember originally validation dataset contains 10 days while testing dataset contains 2 days. So there are a total of 12C2 different splits.

After running all 66 splits, we find some interesting results:

Splits with 0 DT:

ELM-RT zero detection time rate : 66.6667% (44/66)

ELM-PT zero detection time rate : 43.9394% (29/66)

ELM-OT zero detection time rate : 45.4545% (30/66)

ELMV-RV zero detection time rate : 53.0303% (35/66)

ELMV-PV zero detection time rate : 42.4242% (28/66)

ELMV-OV zero detection time rate : 42.4242% (28/66)

LR-RT zero detection time rate : 96.9697% (64/66)

LR-PT zero detection time rate : 63.6364% (42/66)

LR-OT zero detection time rate : 54.5455% (36/66)

Splits with 100% MDR:

ELM-RT 100% missed detection rate rate : 31.8182% (21/66)

ELM-PT 100% missed detection rate rate : 31.8182% (21/66)

ELM-OT 100% missed detection rate rate : 31.8182% (21/66)

ELM-RV 100% missed detection rate rate : 31.8182% (21/66)

ELM-PV 100% missed detection rate rate : 21.2121% (14/66)

ELM-OV 100% missed detection rate rate : 19.697% (13/66)

LR-RT 100% missed detection rate rate : 22.7273% (15/66)

LR-PT 100% missed detection rate rate : 18.1818% (12/66)

LR-OT 100% missed detection rate rate : 15.1515% (10/66)

Splits with satisfactory results (FAR <= 5% and MDR <=5% and != 0):

ELM-RT satisfactory alarm rate : 4.5455% (3/66)

ELM-PT satisfactory alarm rate : 9.0909% (6/66)

ELM-OT satisfactory alarm rate : 6.0606% (4/66)

ELM-RV satisfactory alarm rate : 7.5758% (5/66)

ELM-PV satisfactory alarm rate : 9.0909% (6/66)

ELM-OV satisfactory alarm rate : 9.0909% (6/66)

LR-RT satisfactory alarm rate : 1.5152% (1/66)

LR-PT satisfactory alarm rate : 12.1212% (8/66)

LR-OT satisfactory alarm rate : 12.1212% (8/66)

Interpretation 4: from the findings above, clearly we can see redline alarm system perform the worst. Also after a series of further investigation, day 16, 19, 23, 26, 61, 73 are the root of causing bad alarm systems, thus whenever any of them are looped into the testing dataset, our results tend to be bad. In addition, day 14 and 15 are the 'best days', whenever they are in the testing dataset, we tend to have good results (recall day 14 and 15 are in the testing dataset as we are given initially).

So, what is the difference between a "good split" and a "bad split"?

Good split:

- contain relatively many adverse events (around 100)
- contain relatively steep drop (more than 1 degree)

Bad split:

- contain relatively less adverse events (around 20)
- contain relatively mild drop (less than 1 degree)

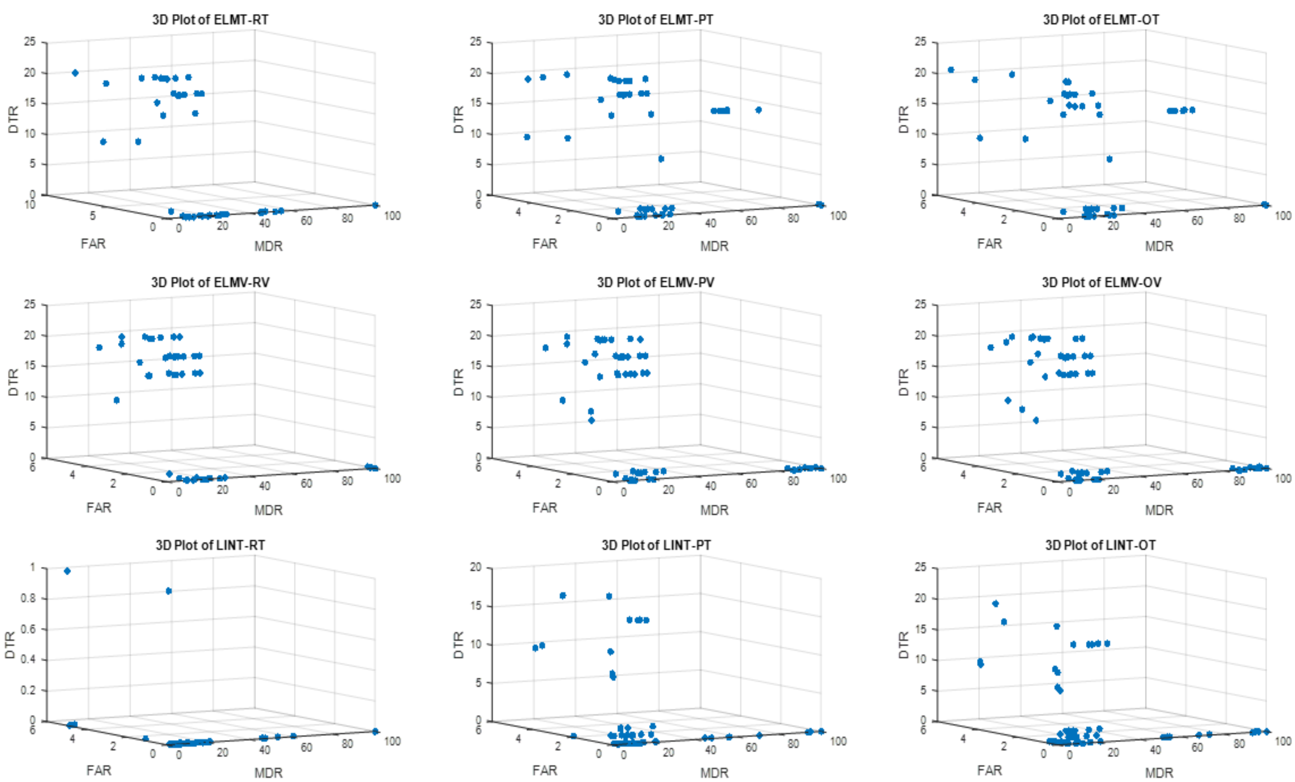


Figure 4.13: Final 3D Plot of FAR, MDR, DT for Each Model

Final Interpretation: generally speaking, LR w/ Training-Base Alarm system, ELM w/ Training-Base or Validation-Base Alarm System both perform well. In more details, after running "cross-validation' splitting analysis, the redline detection method is realized that it does not perform as good as the other two detection methods. Overall, since there is a tradeoff between FAR and MDR, once we increase the MDR constraint, we tend to have results with lower FAR but higher MDR. Specifically, the relatively best models we can explore are:

- LR w/ Predictive-Training Alarm System: 3.5% (FAR), 1.9% (MDR), 10.5 units(DT).
- ELM w/ Predictive-Training Alarm System: 2.87% (FAR), 3.33% (MDR), 10.5 units (DT).
- ELM w/ Predictive-Validation Alarm System: 1.38% (FAR), 2.86% (MDR), 10.5 units (DT).

5. Conclusion & Recommendations

In this project, we used ACCEPT in Matlab to analyze the Cold Complain dataset gathered from the Sustainability Base smart building. Under ACCEPT, we analyzed different regression and detection algorithms to predict the adverse events. In the main steps, we estimated the hyperparameters, set alarm constraints and observed changes in performance, compared and found the best performance method, used training and validation alarm to refine the alarm design, and ran all validation-training splits and observe alarm detection in these splits. After comparing different system, we selected the best models as LR w/Predictive-Training Alarm System, ELM w/ Predictive-Training Alarm System, and ELM w/Predictive-Validation Alarm System.

We learned many useful techniques in this projects and also have some recommendations for the further research on Cold Complaints data and ACCEPT quality improvement.

For the Cold Complaint data, ACCEPT can include 3 lagged response variables. We use random data-set optimal parameters as 0.04 for Ridge Regression and 164 Neurons for Extreme Learning Machine.

For the software, ACCEPT framework can be improved in the following ways:. The Data files can be renamed with same number of digits, for example using 02 for the second data file and 12 for the twelfth file so that they all two digits, then these files will be in correct order when read in ACCEPT. Also we modified the coding of new ground truth function for ACCEPT. The grid search parameters can be smaller, using 10 instead of 100. With this modification, it can enable the users to run code much faster and still get reasonable result.

For the ACCEPT future projects, we also have some recommendations to help improve the quality of ACCEPT. ACCEPT can add more combined functionality with EXCEL and other data file. For example, it can include modules that perform variable reduction from CSV files and add excel output that show individual missed or false detections. Matlab is the main platform that supports ACCEPT. ACCEPT can be updated to fit and run charts in current Matlab version, R2016. ACCEPT includes many powerful algorithms, but the tradeoff is that it is time consuming to run the toolbox. It can take hours to run some models. Further project can investigate and improve the runtime for algorithms. The result of ACCEPT are displayed in separate graphs and tables. ACCEPT could put these result graphs and tables in a single document which would make ACCEPT more convenient for users to compare different outputs.

We will transfer information we have to the next ACCEPT team from Southern Denmark University Master Program. We had a Webex presentation on May 3rd that showed ACCEPT demonstration and debugging points. We will also send them some main materials includes the updated ACCEPT code with all fixed and cross-validation capability, algorithms for conversion back and forth from ACCEPT format to CSV, and cleaned and configured datasets.

References:

Bickford, R. 2000. MSET Signal Validation System Final Report. Technical Report, NASA Contract NAS8-98027.

James, G.; Witten, D.; Hastie, T.; and Tibshirani, R. 2013. An Introduction to Statistical Learning. Springer.

Martin, R.; Das, S.; Janakiraman, V.; and Hosein, S. 2015a. ACCEPT: Introduction of the adverse condition and critical event toolbox. Techical Report NASA/TM-2015-218927. National Aeronautics and Space Administration NASA.

Martin, R.; Mengshoel, O.; Hosein, S.; Jayakumaran, J.; Morga, M.; Basak, A.; and Aghav, I. 2015b. Identifying Contributing Factors of Occupant Thermal Discomfort in a Smart Building. Association for the Advancement of Artificial Intelligence (AAAI) Conference 2015.

Martin, R.; O., M.; Basa, A.; and Hosein, S. 2015c. Scalable Causal Learning for Predicting Adverse Events in Smart Buildings. Association for the Advancement of Artificial Intelligence (AAAI) Conference 2015. NASA Sustainability Base Website. 2015. Nasa sustainability base.

Energy Dieting. (2015, August 26). Retrieved from http://www.nasa.gov/ames/facilities/sustainabilitybase/energydieting

Hull, D. (2012, April 19). NASA's greenest building unveiled at Moffett Field. From http://www.mercurynews.com/business/ci_20435325/nasas-sustainability-base-at-moffett-field-is-nations

Sustainability Base. (2012, April 18). Retrieved from

http://www.nasa.gov/centers/ames/pdf/640590main_SB_FactSheet.pdf