# Machine Learning of Hazard Model Simulations for use in Probabilistic Risk Assessments

by

**Clarence Worrell**

B.S. in Fire Protection Engineering, University of Maryland, 2001

M.S. in Fire Protection Engineering, University of Maryland, 2002

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2017

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented by

Clarence Worrell

It was defended on

November 6, 2017

and approved by

Joel Haight, Ph.D., Associate Professor, Industrial Engineering

Thomas Congedo, Ph.D., Adjunct Professor and Director of Nuclear Engineering Program

Thesis Director: Louis Luangkesorn, Ph.D., Assistant Professor, Industrial Engineering

**Machine Learning of Hazard Model Simulations for use in Probabilistic Risk Assessments**

Clarence Worrell, M.S.

University of Pittsburgh, 2017

This study explored the use of machine learning to generate metamodel approximations of a physics-based fire hazard model called Consolidated Fire and Smoke Transport (CFAST). The motivation to generate accurate and efficient metamodels is to improve modeling realism in probabilistic risk assessments where computational burden has prevented broader application of high fidelity codes. The process involved scenario definition, generating training data by iteratively running the hazard model over a range of input space, exploratory data analysis and feature selection, an initial testing of a broad set of metamodel types, and finally metamodel selection and tuning.

The study identified several factors that should be considered when metamodeling a physics-based computer code. First, the input space should be limited to a manageable scale and number of parameters; otherwise generating sufficient training data becomes infeasible. Second, there is a relationship between the physics being characterized and the metamodel types that will successfully mimic those physics. Finally, metamodel accuracy and efficiency must be balanced against initial development costs. Once developed, trained metamodels are portable and can be applied by many users over a range of modeling conditions.

The Idaho National Laboratory software called RAVEN was used to facilitate the analysis. Twenty five (25) metamodel types were investigated for their potential to mimic CFAST-calculated maximum upper layer temperature and its timing. Linear metamodels struggled to predict with accuracy because the physics of fire are non-linear.

*k*-nearest neighbor (*k*NN) model tuning generated a $k = 4$ model that fit the vast majority of CFAST calculations within ±10% for both maximum upper layer temperature and its timing. This model showed good generalization with use of 10-fold cross validation.

The resulting *k*NN model was compared to algebraic models typically used in fire probabilistic risk assessments. The algebraic models were generally conservative relative to CFAST; whereas the *k*NN model closely mimicked CFAST. This illustrates the potential of metamodels to improve modeling realism over the simpler models often selected for computational feasibility. While the *k*NN metamodel is a simplification of the higher fidelity CFAST code, the error introduced is quantifiable and can be explicitly considered in applications of the metamodel.

## TABLE OF CONTENTS

**PREFACE** ..... **XIII**

**1.0 INTRODUCTION** ..... **1**

**2.0 LITERATURE REVIEW** ..... **5**

**2.1 METAMODELING PROCESS** ..... **5**

**2.2 FIRE HAZARD MODELS** ..... **9**

**2.3 RAVEN SOFTWARE OVERVIEW** ..... **14**

**2.4 PREVIOUS APPLICATIONS OF RAVEN** ..... **15**

**2.5 REDUCED ORDER MODELS AVAILABLE IN RAVEN** ..... **19**

**2.5.1 N-Dimensional Spline** ..... **19**

**2.5.2 Gaussian Polynomial Fitting** ..... **19**

**2.5.3 High Dimensionality Model Representation (HDMR)** ..... **20**

**2.5.4 MSR** ..... **20**

**2.5.5 N-Dimensional Inverse Distance Weighting** ..... **20**

**2.5.6 Linear Models** ..... **21**

**2.5.7 Support Vector Machines (SVM)** ..... **21**

**2.5.8 Multi-Class** ..... **22**

**2.5.9 Naïve Bayes** ..... **22**

2.5.10 Neighbors ..... 22
2.5.11 Tree-Based ..... 23
2.5.12 Gaussian Process ..... 23
2.5.13 Auto-Regressive Moving Average (ARMA) ..... 24
2.6 PREVIOUS APPLICATIONS OF REDUCED ORDER MODELS TO APPROXIMATE NUCLEAR POWER PLANT HAZARDS ..... 24
2.6.1 The Ohio State University Study ..... 25
2.6.2 University of California Los Angeles Study ..... 26
3.0 METHODOLOGY ..... 28
3.1 FIRE MODEL SELECTION ..... 28
3.2 FIRE SCENARIO DEFINITION ..... 33
3.2.1 Characteristics of High Risk Fire Scenarios ..... 34
3.2.2 Validated Range of Fire Model Input Space ..... 39
3.2.3 Fire Scenario Definition for RAVEN Application ..... 41
3.3 RAVEN-CFAST MODEL SETUP ..... 47
3.4 INPUT AND OUTPUT PARAMETERS OF INTEREST ..... 48
3.5 FULL GRID SAMPLING OF THE INPUT SPACE ..... 49
3.6 SIMULATION RESULTS ..... 55
3.7 DATA PREPARATION FOR METAMODELING ..... 63
3.7.1 Consolidating the Data ..... 63
3.7.2 Feature Selection ..... 65
3.7.3 Centering and Scaling ..... 69
3.7.4 Initial Training and Testing: Linear Models ..... 71

**3.7.5 Initial Training and Testing: Tree-Based Models ..... 78**

**3.7.6 Initial Training and Testing: Neighbor-Based Models ..... 79**

**3.7.7 Initial Training and Testing: Support Vector Machine ..... 80**

**3.7.8 Initial Training and Testing: Summary ..... 81**

**3.8 METAMODEL SELECTION AND TUNING ..... 81**

**3.8.1 Decision Tree Regressor ..... 82**

**3.8.2 *k*-Nearest Neighbor (*k*NN) Regression ..... 87**

**3.8.3 Support Vector Machine ..... 91**

**3.9 COMPARISON TO ALGEBRAIC FIRE MODELS ..... 94**

**3.10 ACCURACY-EFFICIENCY TRADEOFF ..... 99**

**3.11 LIMITATIONS ..... 104**

**3.12 FUTURE RESEARCH ..... 107**

**4.0 CONCLUSIONS ..... 110**

**APPENDIX A ..... 114**

**BIBLIOGRAPHY ..... 192**

## LIST OF TABLES

Table 1. Experimental Designs included in RAVEN ..... 8

Table 2. Additional RAVEN-Related Technical References ..... 17

Table 3. Fire Model Validation Results from NUREG-1934 (McGrattan et al., 2012) ..... 31

Table 4. Fire Scenarios Contributing most to Core Damage Frequency for Sample of Plants .... 35

Table 5. Cabinet Fire Peak Heat Release Rate Gamma Distributions per NUREG-2178 (USNRC & EPRI, 2015) ..... 37

Table 6. Summary of Input Space Defined by High Risk Fires at Three Sampled Plants ..... 39

Table 7. Full-Scale Fire Tests used for Fire Model Verification and Validation ..... 40

Table 8. Fire Model Validated Ranges per NUREG-1934 (McGrattan et al., 2012) ..... 41

Table 9. Fire Scenario Input Space over which RAVEN will be Exercised..... 42

Table 10. Summary of Input and Output Parameters of Interest ..... 49

Table 11. Full Grid Sampling Plan to Create Population of Data against which to Test RAVEN ROM Capabilities ..... 50

Table 12. Subdivision of Full Grid into Batches using the Heat Release Rate Parameter ..... 51

Table 13. Description of each Column in the Consolidated .csv File used for Metamodel Training and Testing ..... 64

Table 14. Comparison of Computer Run Time and Accuracy across Several Modeling Options ..... 103

Table 15. Input Parameter Space used for K-Nearest Neighbor Model Training ..... 111

## LIST OF FIGURES

Figure 1. Compartment Fire Behavior ..... 9

Figure 2. Example CFAST Model ..... 12

Figure 3. Example CFD Fire Model ..... 13

Figure 4. Fire Model Bias Factors for Predicted Quantities ..... 32

Figure 5. Range of Fire Heat Release Rate Profiles to be executed by RAVEN ..... 45

Figure 6. Postulated Fire Scenario: Electrical Cabinet Fire ..... 46

Figure 7. CFAST Rendering of Range of Compartment Shapes to be Evaluated by RAVEN ..... 47

Figure 8. Grid Sampled Fire Scenario Input Space ..... 52

Figure 9. Histograms of the Factors upon which the Heat Release Rate Profiles are Based ..... 53

Figure 10. Illustration of the Two Models over which Machine Learning is Exercised ..... 54

Figure 11. Prescribed versus Realized Heat Release Rates ..... 56

Figure 12. Upper Layer Temperature Profiles Calculated by CFAST ..... 57

Figure 13. Lower Layer Temperature Profiles Calculated by CFAST ..... 58

Figure 14. Upper Layer Height Profiles Calculated by CFAST ..... 59

Figure 15. Compartment Pressure Profiles Calculated by CFAST ..... 60

Figure 16. Upper Layer Optical Density Profiles Calculated by CFAST ..... 61

Figure 17. Flame Height Profiles Calculated by CFAST ..... 62

Figure 18. Response Variable Histograms..... 63

Figure 19. Correlation Plot between all Predictor and Response Variables ..... 67

Figure 20. Example Comparison of Raw Parameter Values to their Centered and Scaled Values ..... 70

Figure 21. Initial Fitting of Linear Metamodels (1-4) ..... 72

Figure 22. Initial Fitting of Linear Metamodels (5-8) ..... 73

Figure 23. Initial Fitting of Linear Metamodels (9-12) ..... 74

Figure 24. Initial Fitting of Linear Metamodels (13-16) ..... 75

Figure 25. Initial Fitting of Linear Metamodels (17-20) ..... 76

Figure 26. Initial Fitting of Tree-Based Metamodels ..... 79

Figure 27. Initial Fitting of Neighbor-Based Metamodels..... 80

Figure 28. Support Vector Regression Metamodels ..... 81

Figure 29. Regression Tree Coefficient of Determination ($R^2$) as a Function of Number of Splits ..... 84

Figure 30. Regression Tree for Maximum Upper Layer Temperature ..... 85

Figure 31. Regression Tree Predicted versus CFAST Estimated ..... 86

Figure 32. *k*NN Root Mean Squared Error as a Function of Number of Neighbors ..... 89

Figure 33. K-Nearest Neighbor Predicted versus CFAST Estimated..... 90

Figure 34. Ratio of *k*NN Predicted to CFAST Calculated..... 91

Figure 35. Support Vector Machine Root Mean Squared Error as a Function of Complexity Parameter..... 93

Figure 36. Support Vector Machine Predicted versus CFAST Estimated..... 94

Figure 37. Comparison of CFAST to *k*NN and Algebraic Models..... 98

Figure 38. *k*NN Predicted vs. CFAST Calculated over Range of Training Sample Sizes ..... 101

Figure 39. *k*NN Root Mean Squared Error vs. Training Sample Size ..... 102

# PREFACE

This work is dedicated to my friends at Westinghouse Electric Company.

# 1.0 INTRODUCTION

The United States Department of Energy explains in the Light Water Reactor Sustainability Integrated Program Plan (Idaho National Laboratory, 2017) that:

> *Nuclear power has safely, reliably, and economically contributed approximately 20% of electrical generation in the United States over the past two decades. It remains the single largest contributor (more than 60%) of non-greenhouse-gas-emitting electric power generation in the United States.*
>
> *Domestic demand for electrical energy is expected to grow by about 24% from 2015 to 2040. At the same time, most of the currently operating nuclear power plants will begin reaching the end of their initial 20-year extension to their original 40-year operating license, for a total of 60 years of operation (the oldest commercial plants in the United States reached their 40th anniversary in 2009)…*
>
> *…Operation of the existing fleet of plants to 60 years, extending the operating lifetimes of those plants beyond 60 years and, where practical, making further improvements in their productivity are essential to support the nation's energy needs.*

The program defines sustainability as "…the ability to maintain safe and economic operation of the existing fleet of nuclear power plants for as long as possible and practical" (Idaho National Laboratory, 2017). Four pathways are identified to meet this objective: 1) Materials Aging and

Degradation, 2) Risk-Informed Safety Margin Characterization (RISMC), 3) Advanced Instrumentation, Information, and Control Systems Technologies, and 4) Reactor Safety Technologies.

The RISMC pathway aims to develop a framework of methods and tools to quantitatively and accurately characterize safety margin. Understanding safety margin is particularly important during extended operation, where plant structures, systems, and components may be more susceptible to age-related failures. The RISMC framework is intended to help plant operators and regulators more cost-effectively manage safety margin during extended plant operation by focusing resources on areas with low safety margin, while reducing unnecessary burden in areas where excess margin exists.

In support of RISMC, the Idaho National Laboratory (INL) is developing software called RAVEN (Cristian Rabiti et al., 2017), an acronym for Risk Analysis and Virtual Control ENvironment. RAVEN at its heart is a statistical analysis platform capable of interfacing with complex system codes, for example thermal-hydraulic models of plant response under accident conditions.

In the RAVEN process, the input space is first stochastically defined. Input space refers to the plant design (mitigating systems and components whose reliabilities are characterized statistically), as well as the collection of hazards (fire, flood, seismicity, high winds, random failures internal to the plant, etc.) that can give rise to initiating events. RAVEN is then coupled with thermal-hydraulic plant response models (for example MAAP and RELAP) that are run many times to evaluate a comprehensive sampling of the input space.

One key difference between RAVEN and the current risk assessment framework is that the thermal-hydraulic models are typically run only a few times in the current framework, for

example to characterize plant response to a subset of important sequences using either bounding or best-estimate values. In the RAVEN-based framework, the thermal-hydraulic models are run many times on a probabilistic sampling basis. The great computational expense of running thermal-hydraulic models has, until recent strides in computing power, prevented using these codes on a sampling basis. In addition to leveraging computing power, such as parallel processing on large LINUX clusters, RAVEN also incorporates machine learning methods (for example Gaussian process models an support vector machines) that are trained by initial thermal-hydraulic model runs, and that eventually can replace the thermal-hydraulic model to accelerate the overall simulation.

This thesis explores the following specific RAVEN capabilities:

- Defining a stochastic input space
- Sampling execution of a hazard model over range of uncertain input space
- Machine learning of the resulting hazard model input/output relationships

Fire is the selected hazard for this study because fire has been shown to be a dominant risk contributor to nuclear power plants, and few of the past RAVEN studies have explored fire. Previous and current RAVEN studies have focused instead on accidents initiated by plant equipment failures, seismicity, and external flooding.

One fire scenario is selected for development in RAVEN. The characteristics and boundary conditions for risk significant nuclear power plant fire scenarios are examined to define a meaningful range of input space over which the RAVEN framework is exercised. The input space is defined to be sufficiently broad that it encompasses the most meaningful potential future applications of this work.

The fire modeling software selected for this application is called Consolidated Fire and Smoke Transport (CFAST) Version 6 (R. Peacock, Jones, Reneke, & Forney, 2008), which is publically available and developed by the National Institute of Standards and Technology. CFAST is selected over simpler fire models, which tend to be conservative, as well as more complex models like computational fluid dynamics, which tend to be computationally demanding.

A qualitative review of metamodels available within RAVEN is first performed. Then, a RAVEN model is developed to execute CFAST over a range of input space. An initial fitting of twenty five (25) metamodels to a population of RAVEN-generated fire scenario training data is performed, and the following three metamodel types are selected for further exploration and model tuning:

- Tree-Based Regression
- k-Nearest Neighbor Regression
- Support Vector Machines Regression

The goal of this research is to understand the potential for ROMs to reliably estimate fire-generated conditions. The success of each ROM can be measured in terms of accuracy (for example root mean squared error) and speed (for example CPU-hours required to train and run the ROM). Accurate and efficient ROMs could improve modeling realism in fire probabilistic risk assessments where computational burden has prevented the broader application of fire modeling codes.

# 2.0 LITERATURE REVIEW

Chapter 2 summarizes a literature review of the metamodeling process, fire hazard models, RAVEN software and its previous applications, reduced order modeling methods available within RAVEN, and finally previous applications of metamodeling to approximate hazard modeling within the nuclear power industry.

## 2.1 METAMODELING PROCESS

The term metamodeling refers to the development of models that approximate more complex computer models. The primary motivation of metamodeling is improved computational efficiency over the computer models being approximated. For example, a well-trained and efficient metamodel might be used for uncertainty quantification over many varying parameters; whereas it may be infeasible to quantify the full computer model the hundreds or thousands of times required for uncertainty quantification. Computational fluid dynamics and finite element analysis are example models with often great computational burden where metamodeling may be of value. The metamodeling process can also generate insights into the input/output relationships of the more complex model, and with careful oversight metamodeling can be used for prediction.

(Barton, 2015) provides a tutorial of the metamodeling process, including simplified examples in the field of discrete event simulation. Barton explains the process with the following steps:

1) Define purpose of the proposed metamodeling
2) Identify input and output parameters of interest
3) Choose metamodel type
4) Choose experimental design
5) Fit the metamodel using full model runs specified by the experimental design
6) Validate the metamodel
7) Use metamodel for intended purposes

There are many potential metamodeling purposes. (Barton, 2015) discusses examples with discrete event simulation. (Cohn, Denning, Aldemir, Sezen, & Hur, 2016) exercise metamodels with RAVEN to approximate "stick" computer models of how components and structures respond under seismic excitation. Finally, this thesis explores metamodel approximation of fire hazard models used in probabilistic risk assessments of nuclear power plants.

Selecting the input (predictor) and output (response) parameters of interest defines the metamodeling problem and is generally application-specific. Input selection might be informed by analyst knowledge or suspicions of which phenomena most govern the system. Input parameters that the end-user desires to manipulate in order to observe the resulting system response might also be selected. The selected output (response) parameter is the variable that the metamodel will attempt to predict. Generally a single output is selected, and the metamodel function that relates the selected inputs to the output is referred to as a response surface.

Metamodel complexity and training requirements increase as parameters are added, and this leads to a tradeoff between model fidelity and computational burden.

Many metamodel types and software implementations are available. The INL RAVEN framework (Cristian Rabiti et al., 2017) provides several variations of the following metamodel types, which are geared toward applicability to the nuclear power industry analyses: spline, Gaussian polynomial fitting, high dimensional model representation, MSR, inverse distance weighting, support vector machines, multi-class algorithms, naïve Bayes, tree-based algorithms, Gaussian process models, and auto-regressive moving average. This thesis focuses on metamodeling of fire hazard simulation, which tends to be temporal, and therefore the metamodels available within RAVEN and capable of time series analysis are of most interest. In particular Gaussian process modeling has been used extensively for time series analysis (Williams & Rasmussen, 2006) and has some potential for fire hazard emulation.

The term "experimental design" in this context refers to defining a set of simulation runs whose input and output are used for metamodel training. Given the great computational expense of many simulation codes, the training runs need to be defined strategically to minimize the number of needed runs.

(Sanchez, 2011) and (Barton, 2010) provide overviews of simulation experimental design. Variables are first identified, with primary focus on the independent (input) and dependent (output) variables. Independent variables that are varied in the design of experiment simulations to understand their effects on output are called factors. Intermediate and nuisance variables are also identified. Next each of the selected variables is plotted, and processed through statistical analyses, to understand their ranges and types of relationship (strong, weak, positive, negative, linear, non-linear) with the dependent variable.

There are numerous experimental design approaches, including full factorial, fractional factorial, Latin hypercube, random, sequential screening, and optimization-based methods. The number and type of factors, as well as the response characteristics, influence the selected design. (Sanchez, 2011) provides a chart comparing the strengths and weaknesses of experimental designs relative to the factor and response characteristics. Table 1 identifies the experimental designs available within RAVEN (Cristian Rabiti et al., 2017).

**Table 1.** Experimental Designs included in RAVEN

| **Class** | **Algorithm** |
|---|---|
| Response surface method | Box-Behnken<br>Central composite |
| Factorial | General full factorial<br>2-Level fractional factorial<br>Plackett-Burman |

Next the simulation runs defined by the design of experiments are executed, and the resulting data are used to train the selected metamodel. The model can be validated using error metrics such as mean squared error (MSE) and coefficient of determination ($R^2$). Model performance should be assessed not only on the training data, but also on a test set of data for which the model has not been trained. The acceptable level of model performance is application-specific, and in the case of performing uncertainty analysis for nuclear power plant hazards, it is likely that a relatively high accuracy is required.

## 2.2 FIRE HAZARD MODELS

(Iqbal, Salley, & Weerakkody, 2004) and (McGrattan et al., 2012) provide basic overviews of fire behavior as well as guidance on using publically available fire modeling tools to estimate fire effects such as flame irradiation, plume temperature, ceiling jet temperature, and hot gas layer temperature. These guidance documents are primarily directed towards fire safety applications within the commercial nuclear power industry. Figure 1 illustrates the basic elements of compartment fire behavior.

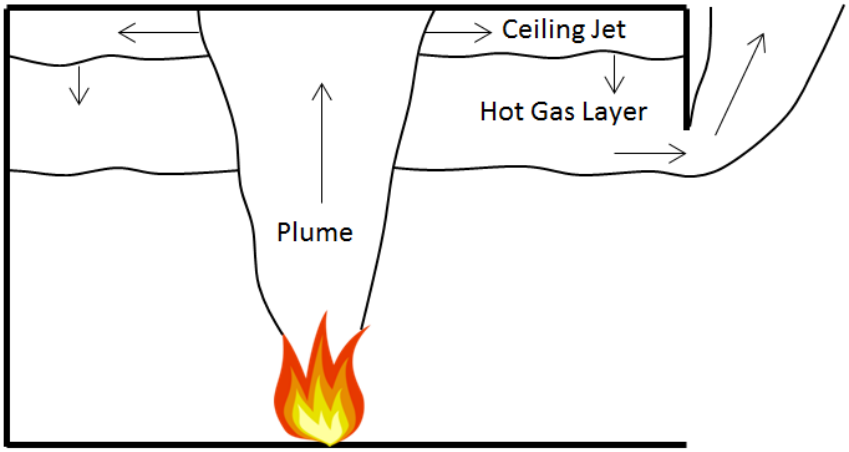


**Figure 1.** Compartment Fire Behavior

Once ignited, a luminous flame forms above the fuel package. This flame radiates energy down to the fuel, causing gasification of the fuel, which flows upward into the flame region. The actual flame occurs at a thin interface where the gaseous fuel meets sufficient oxygen to support

combustion. Within the flame envelope is a region of hot, gasified fuel which has not yet combusted. The flame height is related to the rate of gasification, and how far vertically fuel must travel before it encounters sufficient oxygen to combust. The location of the fuel package against a wall or near a corner (for example, a wall-mounted electrical cabinet) can increase the flame height because the wall reduces on one or two sides the entrainment of oxygen toward the flame, and therefore fuel must travel higher vertically before it combusts.

A plume of hot gases (combustion products) flows vertically above the flame. The temperatures of the flame and plume are equivalent at flame tip. The plume temperature decreases as it travels upward and entrains cooler air. This entrainment causes the plume volume to expand as it travels upward, resulting in an inverted cone shape. The entrainment also causes an axial plume temperature profile that is hottest at its center and coolest at the plume boundary.

Once the plume encounters the ceiling, it is redirected and flows radially outward, away from the plume centerline, underneath the ceiling. This region is called the ceiling jet, and its depth is generally about 10% the height of the room, as a rule of thumb. The ceiling jet cools as it expands radially outward, again as a result of entrainment.

When the ceiling jet encounters the compartment walls, hot gases begin accumulating in the upper portion of the room volume. This region is called the hot gas layer, and it descends as the fire continues to burn and gases accumulate. The hot gas layer descends until it reaches a vent, for example an open door, where gases begin flowing out of the vent. At this point a steady state condition can be reached where cool air flows in through the lower portion of the vent and is drawn into the fire to support combustion, and the resulting combustion products are pumped into the hot gas layer and back out the upper portion of the vent. Some simplified fire models that estimate hot gas layer temperature are based on evaluating this steady state condition.

There are three general classes of fire modeling tools:

- Algebraic Models
- Two-Zone Models
- Computational Fluid Dynamics Models

(Iqbal et al., 2004) provides an overview of algebraic models used to estimate fire generated conditions, such as flame height, plume temperature, ceiling jet temperature, hot gas layer temperature, and thermal radiation. (McGrattan et al., 2012) provides guidance for the application of fire modeling tools to nuclear power plant analyses.

The algebraic models are straightforward to implement, but they generally yield conservative results. These models are also limited in their ability to estimate the evolution of fire-generated conditions as a function of time.

Two-zone models divide the analysis space into two zones, one representing a hot upper layer where energy released by the fire accumulates, and the other representing a relatively cool lower layer. The model solves the conservation of energy and mass equations across these two layers as a function of time. Two-zone models generally provide more realistic results relative to algebraic models, and they also estimate conditions as a function of time; however, these improvements come at some computational expense. CFAST (R. Peacock et al., 2008) is an example of a publically available two-zone model, which was developed by the National Institute of Standards and Technology and is widely used by the fire protection community. Typical CFAST models have a runtime of a few minutes, and some effort is required to set up the model. Figure 2 depicts an example CFAST model.

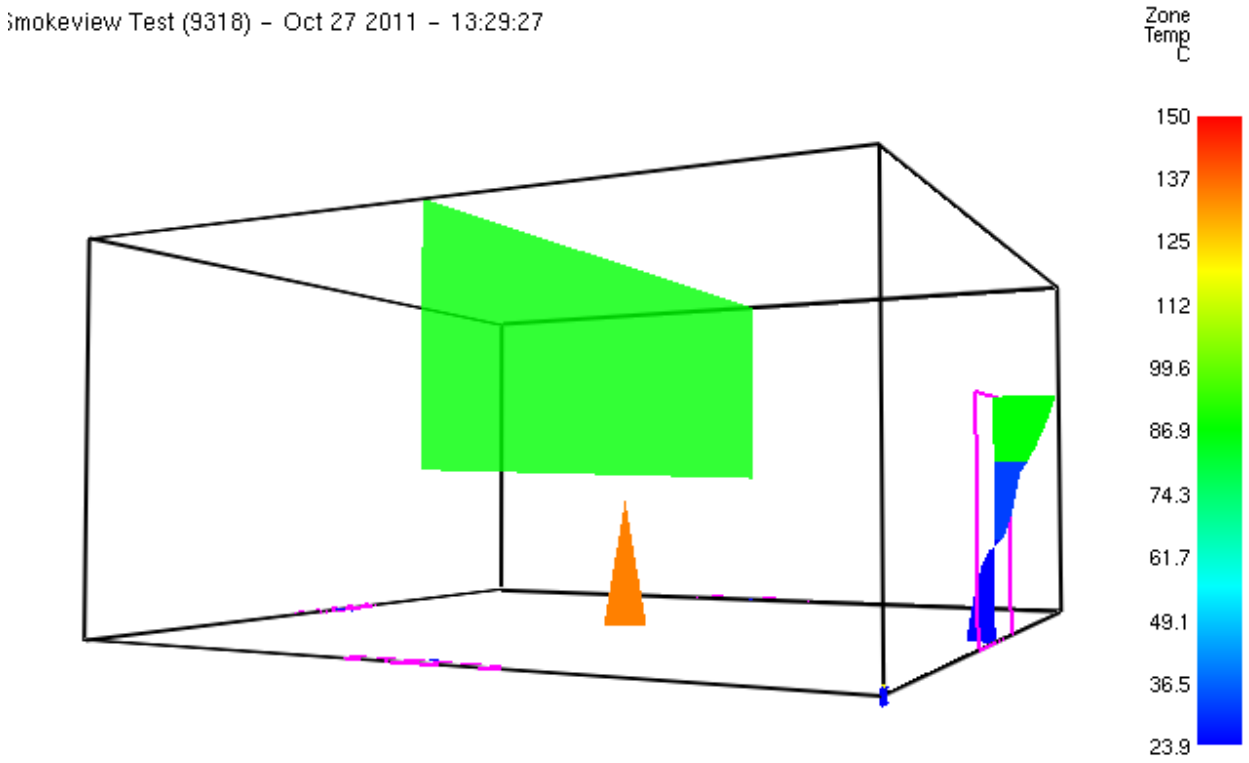


**Figure 2.** Example CFAST Model

Finally, computational fluid dynamics models subdivide the analysis space into a mesh, commonly involving tens or hundreds of thousands of cells. The analysis timeline is also discretized into time steps. The conservation of energy, mass, and momentum equations are solved across each cell surface for each time step throughout the simulation. These models can produce highly resolute and generally accurate estimations of fire-generated conditions (temperature, heat flux, flow, visibility, etc.) throughout the scenario; however, these models come at great computational expense. Relatively basic fire scenarios can require many hours of computer runtime, and several days of runtime is not uncommon for complex scenarios. Figure 3 depicts an example CFD fire model.

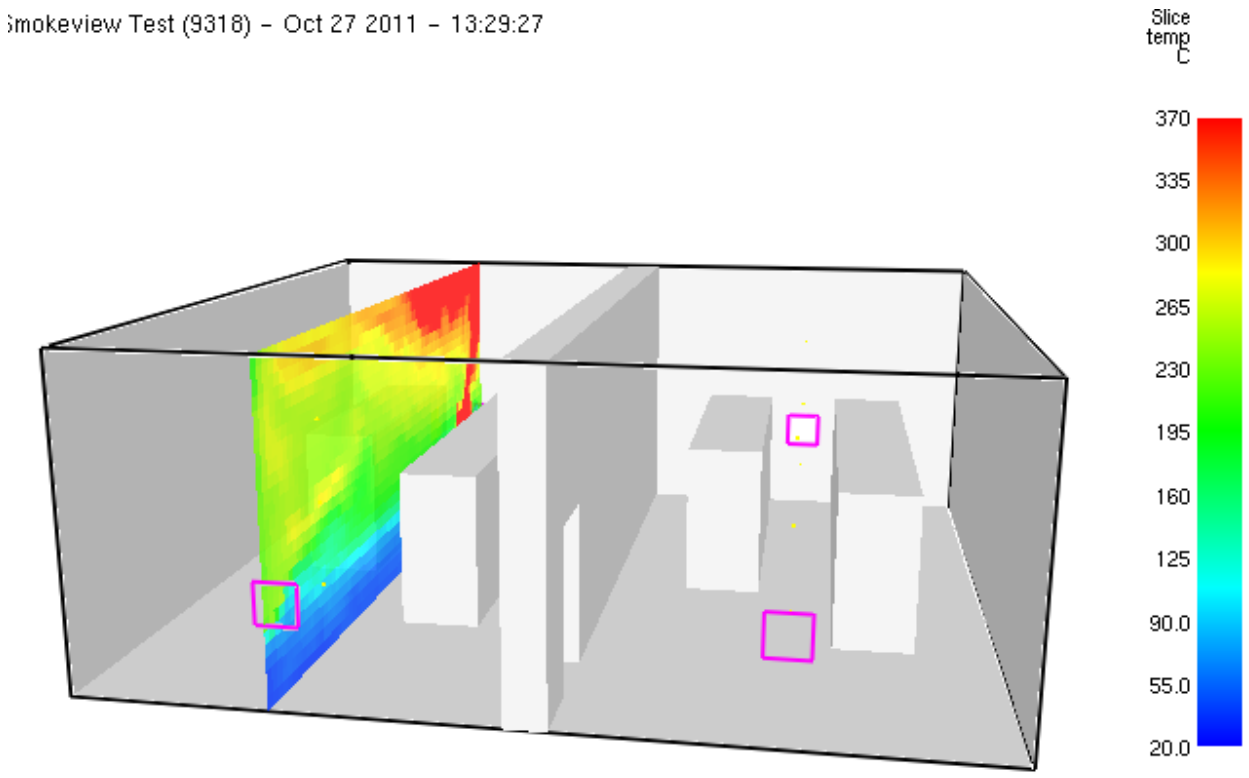


**Figure 3.** Example CFD Fire Model

NUREG-1824 (Hill et al., 2007) provides a systematic verification and validation of the algebraic, two-zone, and computational fluid dynamics fire modeling tools described in (Iqbal et al., 2004) and NUREG-1934 (McGrattan et al., 2012). This study compared fire modeling results against a series of full-scale fire test experiments for parameters such as flame height, plume temperature, ceiling jet temperature, and hot gas layer temperature. The conclusion included a qualitative ranking of each model for each of the parameters to be estimated, which at a high level indicated that the computational fluid dynamics model was generally most accurate, the simplified algebraic models tended to be conservative, and the two-zone models tended to be somewhere in between.

NUREG-1824 (Hill et al., 2007) also noted that it is important for the models to be used within their capabilities and ranges of applicability. The subsequent NUREG-1934 (McGrattan et al., 2012) provides criteria for determining whether any particular scenario is within the range of applicability of various fire modeling tools. Guidance is also provided to characterize the uncertainty associated with fire model output.

## 2.3 RAVEN SOFTWARE OVERVIEW

RAVEN is currently under development by the Idaho National Laboratory. An open-source and periodically updated version of the software is available at their 'github' (Idaho National Laboratory, n.d.). An extensive user manual (Cristian Rabiti et al., 2017) and a theory (Andrea Alfonsi et al., 2017) describe the technical methods implemented by RAVEN and how to install, set up, and run the software.

RAVEN can be installed on Linux, OSX, and Microsoft Windows computing platforms. The code uses a mix of the XML, C++, C, and Python programming languages, and the installation is relatively large, including a variety of dependencies. The central input file is written in XML and can be created and modified with any text file editor. The XML input file has blocks where the inputs, models, outputs, and overall calculation flow are defined.

Models are available in couple forms. Some existing models are available directly within RAVEN and have associated XML input file commands. User-defined models can be written in a separate Python (.py) file and called from the XML input file at the desired point in the calculation flow. RAVEN also includes a process for executing external models, such as the

CFAST fire model to be used in this analysis, or the MAAP and RELAP thermal-hydraulic plant response models.

A variety of post-processing techniques are available within RAVEN. The MatPlotLib library (Hunter, 2007) is available for constructing high quality plots and visualizations. Basic summary statistics and limit surfaces can be calculated, and a data mining tool called knowledge discovery in databases (KDD) is available.

## 2.4 PREVIOUS APPLICATIONS OF RAVEN

One of the first major analyses using RAVEN was performed by the developers at Idaho National Laboratory as a demonstration case and is summarized in (Smith et al., 2014) and (Diego Mandelli, Prescott, et al., 2015). The case study was highly relevant in the wake of the Fukushima Dai-ichi nuclear accident (Miller et al., 2011) and examined response of a pressurized water reactor design to station blackout, seismic-induced station blackout, and tsunami. The study also examined the effect of power uprate on plant response to the progression and timing of the postulated accidents. The model was used to assess the potential risk benefit of physical plant modifications, such as installation of wave protection walls and moving or otherwise bunkering the emergency diesel generators, and revisions to emergency operating procedures such as crediting flexible coping strategies to restore emergency power. A predecessor study (D Mandelli et al., 2014) similarly examined response of a boiling water reactor design to station blackout.

(Szilard et al., 2015) document an early demonstration of using the RISMC framework to manage safety margin under a proposed federal rulemaking that would revise the acceptance criteria for plant response to postulated large break loss of coolant accidents. These reports consider using RAVEN to integrate and manage the overall analysis from sampling the uncertain input parameters, executing the thermal-hydraulic plant response model (RELAP5-3D in this case), and post-processing the results by for example using limit surface searching algorithms to bisect regions of success and failure.

Table 2 identifies national laboratory reports and conference papers that summarize software development progress at various milestones and include a significant amount of technical background. The RAVEN user guide (Cristian Rabiti et al., 2017) and a theory manual (Andrea Alfonsi et al., 2017) summarize the final product, and the below references can be consulted for historical background or as supplemental technical content.

**Table 2.** Additional RAVEN-Related Technical References

| Reference | Topic |
|---|---|
| (Cristian Rabiti, Alfonsi, Cogliati, Mandelli, & Kinoshita, 2012)<br>(Cristian Rabiti, Alfonsi, Mandelli, Cogliati, & Kinoshita, 2014)<br>(A Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita, 2013)<br>(Andrea Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita, 2013a)<br>(Diego Mandelli et al., 2013)<br>(A Alfonsi et al., 2013)<br>(Andrea Alfonsi, Cristian, et al., 2014)<br>(C Rabiti, Alfonsi, Mandelli, Cogliati, & Martineau, 2012)<br>(Rabitia, Alfonsi, Cogliati, Mandelli, & Kinoshita, 2014) | General / overview |
| (Christian Rabiti et al., 2013)<br>(Diego Mandelli, Smith, Ma, et al., 2014)<br>(Diego Mandelli, Smith, Alfonsi, & Rabiti, 2014) | Demonstration cases |
| (Andrea Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita, 2013b)<br>(Andrea Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita, 2014b)<br>(A. Alfonsi et al., 2014)<br>(A Alfonsi et al., 2013)<br>(Andrea Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita, 2014a) | Dynamic Event Tree |
| (Cristian Rabiti et al., 2015)<br>(Manselli et al., 2013)<br>(Diego Mandelli, Rabiti, & Alfonsi, 2012) | Reduced order models |

**Table 2 (Continued).**

| Reference | Topic |
|---|---|
| (Sen, Maljovec, Alfonsi, & Rabiti, 2015) | Data mining |
| (Cristian Rabiti, Talbot, Alfonsi, Mandelli, & Cogliati, 2013)<br>(Swiler, Laura, Mandelli, Diego, Rabiti, Crisitan, Alfonsi, 2013)<br>(Diego Mandelli, Smith, Alfonsi, Rabiti, & Cogliati, 2015)<br>(A. Alfonsi et al., 2015) | Math and algorithms |
| (Guler et al., 2014) | Modeling aging |

## 2.5 REDUCED ORDER MODELS AVAILABLE IN RAVEN

RAVEN includes many regression and classification ROM types, including the library of models associated with the Python SciKitLearn library. This section briefly reviews each of the ROM types currently available in RAVEN according to its user guide (Cristian Rabiti et al., 2017). (Kuhn & Johnson, 2016) provides a practical overview of many of the machine-learning methods available within RAVEN, and (Hastie, Tibshirani, & Friedman, 2016) provides a more comprehensive examination including mathematical foundations.

### 2.5.1 N-Dimensional Spline

Spline methods are a type of interpolation where the data space is discretized into intervals, and a function (or "spline") is defined to fit the data within each interval. The final fitted model is comprised of the collection of splines over each interval. This is a piecewise fitting procedure.

### 2.5.2 Gaussian Polynomial Fitting

This reduced order model is also referred to as generalized polynomial chaos expansion and is used for regression. (Cristian Rabiti et al., 2013) describes implementation of this approach in RAVEN, including an example evaluation of a pressurized water reactor under station blackout conditions.

### 2.5.3 High Dimensionality Model Representation (HDMR)

(Li, Rosenthal, & Rabitz, 2001) provides an overview of the HDMR approach and a few example applications from the chemical industry. At a high level, HDMR attempts to represent high dimensionality systems (those with a large number of input parameters) using a relatively small number of input parameters. In addition to reduced order modeling, HDMR is useful for identifying important input-output relationships in high dimensionality systems.

### 2.5.4 MSR

According to the RAVEN user manual (Cristian Rabiti et al., 2017), MSR decomposes the data into monotonic regions and performs fitting within those regions. It appears to be similar to spline-based approaches and can be used for both regression and classification.

### 2.5.5 N-Dimensional Inverse Distance Weighting

Inverse distance weighting is a form of interpolation where the interpolated value is an average of surrounding data points weighted by the Euclidian distance to those points. The inverse weighting assigns less weight to more distant points and conversely more weight to nearby points.

### 2.5.6 Linear Models

RAVEN includes numerous linear models such as ordinary least squares regression, logistic regression, ridge regression, lasso, perceptron, and elastic net. Linear models take the following general form:

$$\boldsymbol{y} = \beta_0 + \boldsymbol{\beta X} + \boldsymbol{e}$$

In this form $\boldsymbol{y}$ is the vector of response variable values, $\beta_0$ is the intercept, $\boldsymbol{\beta}$ is the vector of coefficients in the linear model, $\boldsymbol{X}$ is the matrix of predictor variable values (with dimensions of the number of variables by the number of observations), and $\boldsymbol{e}$ is the vector of errors. Each of the linear model methods attempts to find coefficient values, $\boldsymbol{\beta}$, that minimize the sum of the squared errors associated with the model. Each of the methods differ in the biases and variances they produce in the resulting model, and therefore the selected method should be application-specific and based on user-preferred attributes of the final model.

A linear model is likely not a good surrogate for representing fire hazard model output since the underlying physics of fire behavior are non-linear.

### 2.5.7 Support Vector Machines (SVM)

(Steinwart & Christmann, 2008) provides a comprehensive examination of SVM methods for classification and regression problems, both linear and non-linear. For classification problems where no clear linear separation exists between the two classes to be predicted, SVM formulates an optimization that creates a curved hyperplane separating the two classes to the greatest extent possible. The SVM classification approach is not probabilistic, and data that fall on either side of

the hyperplane are classified accordingly. SVM can be used for both binary and multi-class problems, as well as for regression.

### 2.5.8 Multi-Class

Multi-class algorithms are used for classification problems with more than two potential outcomes. One example is an image recognition problem where the model attempts to classify images as either containing a dog, cat, or fish. Multi-class algorithms will likely not be relevant to the emulation of fire hazard model output, which tends to be numeric and continuous.

### 2.5.9 Naïve Bayes

(Kuhn & Johnson, 2016) provides a practical overview of naïve Bayes as a non-linear classification model. Naïve Bayes estimates the probability that an observation belongs to a particular class given (conditional upon) observed data. The naïve portion of this model assumes that all predictors are independent of each other, which, although not realistic for many applications, simplifies the computation.

### 2.5.10 Neighbors

Neighbor-based approaches include both supervised and unsupervised algorithms and can be used for classification or regression. In the *k*-nearest neighbor approach, the user defines the number of clusters, *k,* and the algorithm recursively processes the data to assign each point to a cluster. In the first pass, *k* cluster centroids are arbitrarily defined, and each data point is assigned

to a cluster based on its Euclidean distance to each centroid. In the second iteration, the centroids are re-calculated based on the initial assignment, and each point is then re-assigned to a cluster based on their Euclidean distance to the new centroids. The process repeats itself until the cluster definitions converge. This approach is an example of an unsupervised neighbor-based classification model. Other neighbor-based approaches exist for regression problems as well.

### 2.5.11 Tree-Based

Tree-based algorithms sequentially partition the data, forming a tree of user-specified depth where each branch represents a data partition. Tree models can be visualized and are easy to interpret. They can handle mixed numerical and categorical data. Tree structure can be sensitive to the training data, and slight variations in the training data can create differing tree structures by changing the partitioning criteria. To mitigate this sensitivity, ensemble methods such as random forest create many trees, and the resulting classifications are based on a voting scheme. There are tree-based approaches for both classification and regression problems.

### 2.5.12 Gaussian Process

(Williams & Rasmussen, 2006) present a thorough framework, example applications, and software implementation of Gaussian process modeling in the context of machine learning. Gaussian process modeling, also known as kriging, is a type of supervised learning suitable for both regression and classification problems. The approach is Bayesian and starts with a Gaussian prior distribution of functions (i.e., the random variable of the distribution is a function), where

each possible function is a representation of how the system response varies with the predictor variables. The prior distribution is selected by the user, based on knowledge of the system, for example whether the response is monotonic, increasing, decreasing, or cyclical. The Gaussian posterior distribution is then calculated with Bayes' rule, where the prior is updated with the training data. Gaussian process models have been used extensively for time series analysis and are therefore a potentially natural fit for reduced order approximation of fire hazard models, which estimate environmental conditions as a function of time.

### 2.5.13 Auto-Regressive Moving Average (ARMA)

ARMA is a forecasting technique used with time-series data. The model is defined by a user-specified weighted combination of auto-regression ($p$) and moving average ($q$) models. The auto-regression portion regresses previous data points to predict the next data point, while the moving average portion uses an average of previous data point values to predict the next data point.

## 2.6 PREVIOUS APPLICATIONS OF REDUCED ORDER MODELS TO APPROXIMATE NUCLEAR POWER PLANT HAZARDS

A literature review identified two studies where ROMs have been explored for their potential to emulate hazard models used in nuclear power plant probabilistic risk analyses, which typically include fire, flooding, seismicity, and high winds. The first study, referred to as The Ohio State University study, examined surrogate model approximation of a seismic hazard, and the second

study by the University of California, Los Angeles explores response surface approximation of a fire hazard model.

### 2.6.1 The Ohio State University Study

(Cohn et al., 2016) explores the use of surrogate models to approximate the response of "stick models", which estimate the acceleration at various locations throughout a structure that is exposed to ground-level motion during an earthquake. The authors tested the following classification and regression models for estimating component failure probability, which is a function of seismic-induced acceleration: *k*-nearest neighbor regressor, *k*-nearest neighbor classifier, inverse distance weighting, linear support vector classifier, and C-support vector classifier.

Distributions were assigned to the uncertain input space, which included floor mass and stiffness in their study. The study used a Latin hypercube sampler to ensure the full input space was assessed. Two training sets were tested, one low-fidelity set using 500 runs of the stick model, and one high-fidelity set using 20,000 runs of the stick model.

Surrogate model errors ranged from about 2% to 65%. In this study, the *k*-nearest neighbor regressor performed well, while support vector machine performed poorly. The authors attributed the poor support vector machine performance to its attempting to subdivide the input space into clear regions of success and failure, which in this application did not exist due to high non-linearity in the underling physics. That is, the relationship between the floor mass and stiffness input parameters and the acceleration estimated by the stick model is highly non-linear.

The study also found that, in its particular application, the prediction error between models trained on 500 runs and 20,000 runs was similar (within about 5%). Finally, the authors observed that the input parameters with the greatest natural ranges, in this case floor stiffness, most influenced the surrogate models. It is possible that this could be alleviated by training the surrogate models on centered and scaled versions of the input data.

### 2.6.2 University of California Los Angeles Study

(Brandyberry & Apostolakis, 1990) explored response surface approximation of a computer model called COMPBRN, which was used at that time to estimate fire-generated conditions in support of probabilistic risk assessments. COMPBRN (Ho, Siu, Apostolakis, & Flanagan, 1986) is a predecessor to the more current CFAST code. The goal of the study was to generate a response surface that could not only be used for uncertainty quantification, but also as a general analytical tool that could potentially be used in place of COMPBRN under certain conditions. The motivation was not to mitigate computer run time, as the authors noted COMPBRN executes efficiently in a mainframe environment; instead their motivation was to mitigate the tedious model setup required for each run (for example creating a Fortran NAMELIST with more than 60 variables that is not carried forward from run to run).

The UCLA study focused on estimation of cable temperature, as a function of time, when exposed to the fire environments of postulated nuclear power plant fire scenarios. Fifteen predictor variables characterizing thermophysical properties, combustion properties, heat transfer characteristics, and room geometry were examined. The analysis used a central composite experimental design, which is an extension of factorial design.

One of the study challenges was that the cable temperature response variable was temporal, evolving with the fire and the compartment heat transfer characteristics throughout the fire duration. To incorporate the time dimension, the authors examined the general shape of the time-temperature profiles estimated by COMPBRN and fit a non-linear regression to that shape. The equation had four constant terms, and a response surface relating each term to the predictor variables was developed. The final model was therefore a combination of the non-linear regression and four fitted response surfaces.

The study then examined implications of the resulting model, which was statistically fitted, and found many of its features comported intuitively with fire behavior. For example, the coefficient values and exponential order of each term indicated that cable tray temperature is controlled primarily by the thermal environment and not its composition. Other model parameters were not easily interpreted, especially those added solely for tuning the regression. This exercise in part was a validation of the fitted model, but it also yielded insights as to the COMPBRN input/output relationships, as well as general insights into fire behavior that may not self-reveal when simply running the computer code.

Finally, the study exercised the surrogate model to predict cable tray time-temperature profiles measured during a series of fire tests performed for Sandia National Laboratories. The surrogate model matched the experimental data reasonably well (which is primarily a validation of COMPBRN), and it matched the COMPBRN estimations very well (which is a validation of the surrogate model).

# 3.0 METHODOLOGY

## 3.1 FIRE MODEL SELECTION

The fire modeling software selected for this application is called Consolidated Fire and Smoke Transport (CFAST) Version 6 (R. Peacock et al., 2008), which is publically available and developed by the National Institute of Standards and Technology. CFAST is selected over simpler fire models, which tend to be conservative, as well as more complex models like CFD, which tend to be computationally demanding. The following paragraphs discuss how this selection was made.

(Sargent, 2008) summarizes the process of systematically verifying and validating simulation models. Verification ensures a given model is translated correctly into the computer program, and validation ensures the model accurately represents the phenomena of interest.

For example, the verification of a finite element analysis model representing heat conduction through a metal plate might focus on how the governing heat transfer equations are implemented via coding (what programing language is used, how input data are imported and declared, how the routines are structured, and how the output is processed, etc.). Meanwhile, validation might focus on whether the selected heat transfer equations represent the conditions over which the model is intended to be applied (whether the correct temporal forms of the equations are used when timing is important, whether empirical constants are appropriate for the

materials under consideration, and whether the model dimensionality is consistent with the intended applications, etc.).

Verification of the fire models under consideration has been performed by the developers. (R. D. Peacock, Forney, & Reneke, 2017) includes a systematic verification of the CFAST model, using a standardized set of test cases designed to exercise the model implementation of the governing energy balance, mass balance, and heat transfer equations. (McGrattan et al., 2017) provides an extensive verification of the CFD fire model called Fire Dynamics Simulator (FDS), and verification of the simpler algebraic fire models can be found throughout the fire protection literature.

Regarding validation, (Sargent, 2008) discusses that validation cost is usually significant, increasing exponentially with the level of model confidence required. The fire models considered herein are used to assess and manage fire risk at nuclear power plants, and high model confidence is therefore required due to the potential consequences of a nuclear accident. In that context the U.S. Nuclear Regulatory Commission funded a significant fire model verification and validation effort involving the model developers (National Institute of Standards and Technology), fire behavior and modeling experts from academia, regulators, and end-users. This program is documented in NUREG-1824 (Hill et al., 2007), and it applied the ASTM E 1355 *Standard Guide for Evaluating the Predictive Capability of Deterministic Fire Models* (American Society for Testing Materials International, 2012). The subsequent NUREG-1934 (McGrattan et al., 2012) provides guidance on how to apply the verified and validated fire models to nuclear power plant fire scenarios.

(Sargent, 2008) identifies a number of validation techniques, several of which were implemented in NUREG-1824 (Hill et al., 2007), including predictive validation, comparison to

other models, and even animation. The predictive validation was performed by comparing fire model predictions to the results of several full-scale fire tests representative of nuclear power plant scenarios. Several models (FDT, CFAST, and FDS) were assessed, and predictions were not only compared to the experimental data, but also to the predictions from each of the other models. Finally, and while this was not a focus of the study, two of the models (CFAST and FDS) provide animated representations of the fire model output. FDS in particular provides a very detailed three-dimensional visualization of the fire and smoke flow. These animations highlight well-established fire behaviors, such as a "V-shaped" fire plume and the accumulation of hot gases near the ceiling, which provides some qualitative validation of the models, even for those without modeling expertise.

One outcome of this project is exercising the selected model over a large range of input space. Some of the model runs will likely represent extreme cases, in particular at the input distribution tails, and examining model performance for these cases may add to the overall validation.

NUREG-1934 (McGrattan et al., 2012) Table 4-1 summarizes a validation comparison of full-scale fire test experiments against model performance, selected to represent typical nuclear power plant scenarios, and this information is reproduced below as Table 3. Note that the FDT is a set of algebraic fire models, CFAST is a two-zone fire model, and FDS is a computational fluid dynamics fire model. The term $\delta$ is a calculated bias factor representing the degree to which the model over-predicted or under-predicted experimental data, the term $\tilde{\sigma}_M$ is a measure of model uncertainty, and the term $\tilde{\sigma}_E$ is a measure of experimental uncertainty. $\delta$ >0 means the model over-predicted the observations, and $\tilde{\sigma}_M$ < $\tilde{\sigma}_E$ means that the model uncertainty is within

experimental uncertainty. Refer to the source document (McGrattan et al., 2012) for additional information.

**Table 3.** Fire Model Validation Results from NUREG-1934 (McGrattan et al., 2012)

| | FDT | | CFAST | | FDS | | Experiment |
|---|---|---|---|---|---|---|---|
| **Model Output** | $\delta$ | $\tilde{\sigma}_M$ | $\delta$ | $\tilde{\sigma}_M$ | $\delta$ | $\tilde{\sigma}_M$ | $\tilde{\sigma}_E$ |
| HGL Temp. | 1.44 | 0.25 | 1.06 | 0.12 | 1.03 | 0.07 | 0.07 |
| HGL Depth | N/A | | 1.04 | 0.14 | 0.99 | 0.07 | 0.07 |
| Ceiling Jet Temp. | N/A | | 1.15 | 0.24 | 1.04 | 0.08 | 0.08 |
| Plume Temp. | 0.73 | 0.24 | 1.25 | 0.28 | 1.15 | 0.11 | 0.07 |
| Oxygen Conc. | N/A | | 0.91 | 0.15 | 1.08 | 0.14 | 0.05 |
| Smoke Conc. | N/A | | 2.65 | 0.63 | 2.70 | 0.55 | 0.17 |
| Room Pressure | N/A | | 1.13 | 0.37 | 0.95 | 0.51 | 0.20 |
| Target Temp. | N/A | | 1.00 | 0.27 | 1.02 | 0.13 | 0.07 |
| Radiant Heat Flux | 2.02 | 0.59 | 1.32 | 0.54 | 1.10 | 0.17 | 0.10 |
| Total Heat Flux | N/A | | 0.81 | 0.47 | 0.85 | 0.22 | 0.10 |
| Wall Temp. | N/A | | 1.25 | 0.48 | 1.13 | 0.20 | 0.07 |
| Wall Heat Flux | N/A | | 1.05 | 0.43 | 1.04 | 0.21 | 0.10 |

The FDT algebraic models are eliminated from consideration due to their validated outputs being limited to hot gas layer temperature, plume temperature, and radiant heat flux. While these are important quantities, hot gas layer depth, ceiling jet temperature, radiant heat flux, and in particular target temperature are useful for probabilistic fire risk assessments.

Figure 4 plots the CFAST and FDS bias factors for each predicted quantity. The CFAST and FDS bias factors are within 10% of each other for all quantities, except radiant heat flux, room pressure, and oxygen concentration. Room pressure is not of interest here because it does not impact target failure probability. While the CFAST bias factor for radiant heat flux is about 20% higher than the FDS bias factor, both models conservatively over-predicted the experimental data ($\delta$ = 1.32 for CFAST and $\delta$ = 1.1 for FDS). For oxygen concentration,

CFAST under-predicted ($\delta$ = 0.91) the experimental data, while FDS over-predicted the data ($\delta$ = 1.08). In this context under-predicting oxygen concentration would be non-conservative if flame extinction is modeled due to inadequate oxygen.

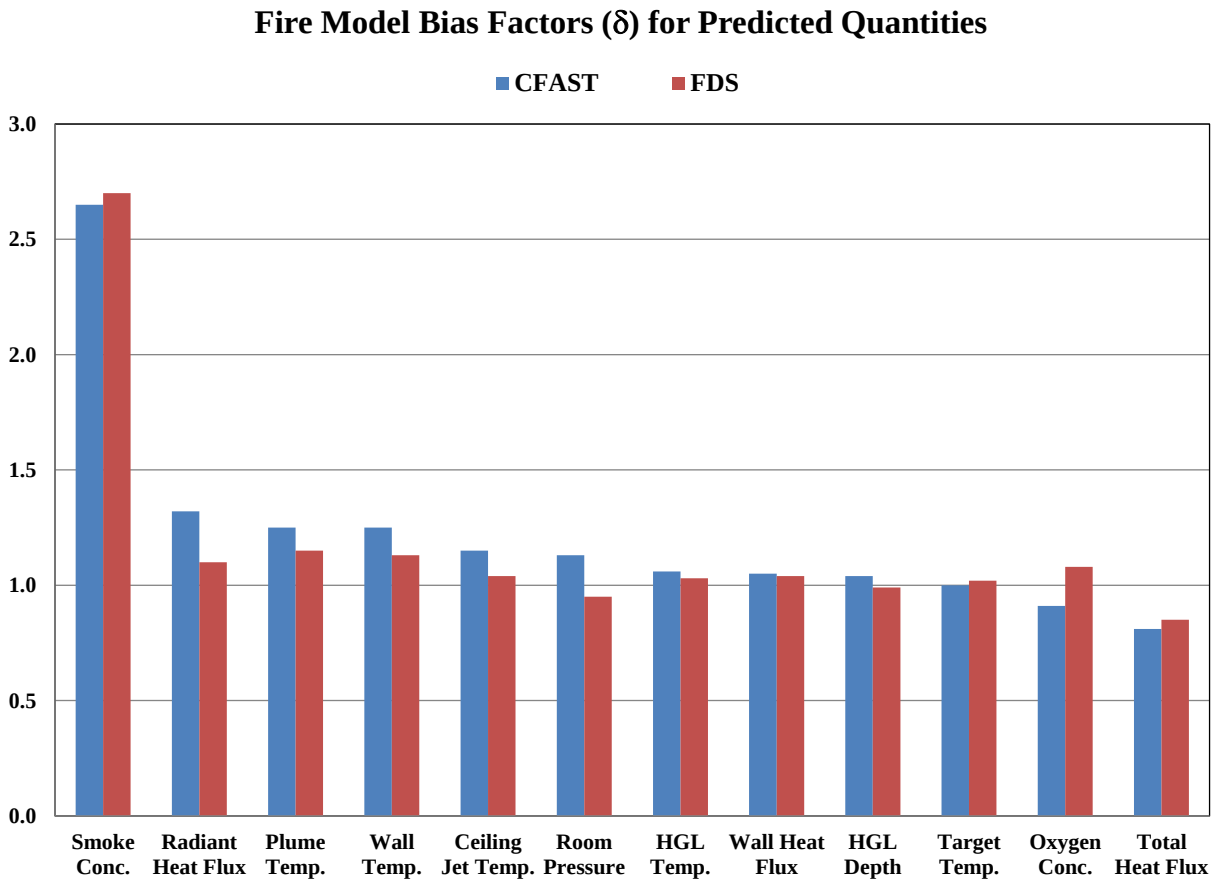


**Figure 4.** Fire Model Bias Factors for Predicted Quantities

Figure 4 also indicates that CFAST was conservatively biased, with respect to the experimental data examined, for all parameters except oxygen concentration and total heat flux. FDS also tended to be conservative with the exception of under-predicting total heat flux. Both CFAST and FDS significantly over-predicted smoke concentration. Finally, while both models

were generally conservative, FDS tended to be slightly more realistic than CFAST with biases closer to 1.0.

In this application, the improved realism of FDS does not outweigh its significantly higher computational expense relative to CFAST. FDS models can require many hours, or days, to run, while similar CFAST models run in a few minutes at most. Given the number and range of uncertain input parameters, developing an accurate metamodel will likely require hundreds or thousands of runs, in which case it would be infeasible to apply FDS.

CFAST is therefore selected for this analysis. Attention is required when using CFAST to predict oxygen concentration (for example if flame extinction is modeled) and heat flux (for example as a target failure mechanism) due to non-conservative model bias. Similarly, care is required when predicting smoke concentration, for example as a visibility impact to plant operators, because CFAST is very conservatively biased for this parameter.

## 3.2 FIRE SCENARIO DEFINITION

For the purposes of metamodel development and uncertainty quantification, the "fire scenario" represents a range of potential conditions characterized by probability density functions. For example, a scenario might have a lognormal occurrence frequency with a 1E-04 /yr mean value, the peak heat release rate might be gamma-distributed with a mean value of 200 kW, and even typically fixed parameters such as room dimensions and ventilation rates may vary. The benefit of varying fixed parameters is to maximize the potential applicability of the trained metamodel.

The following two approaches will be used to define the fire scenario under consideration:

1) Identify the characteristics of high risk fire scenarios per nuclear power plant probabilistic risk assessments.
2) Identify the range of input space over which the CFAST fire model has been validated.

The first approach ensures the metamodel training data encompass relevant scenarios, and the second approach ensures the metamodel applicability is as broad as the model being emulated.

### 3.2.1 Characteristics of High Risk Fire Scenarios

Many U.S. nuclear power plants have converted their fire protection programs to the risk-informed methodology outlined in NFPA 805 (National Fire Protection Association, 2001). This methodology is a risk-informed alternative to the generally prescriptive, rule-based requirements to which plants were initially licensed. Transitioning to NFPA 805 involves submitting a comprehensive license amendment request to the regulator, part of which summarizes the dominant fire risk locations and contributors for the plant. Table 4 summarizes a review of publically available NFPA 805 license amendment requests to identify the highest risk fire scenarios.

**Table 4.** Fire Scenarios Contributing most to Core Damage Frequency for Sample of Plants

| Plant | Design | Top 5 Fire Scenarios |
|---|---|---|
| Cooper (Nebraska Public Power District, 2012) | General Electric Type 4 boiling water reactor with wet containment | ▪ Diesel generator fire<br>▪ Transient fire in turbine building corridor<br>▪ Bus duct fault in switchgear room<br>▪ Station battery charger fire<br>▪ Station battery fire |
| Arkansas Nuclear One (Entergy Operations, 2014) | Babcock & Wilcox pressurized water reactor with large dry containment | ▪ Switchgear room fire<br>▪ Main control room abandonment<br>▪ Radwaste processing area fire<br>▪ Pipe chase fire<br>▪ Containment fire |
| McGuire (Duke Energy, 2013) | Westinghouse four-loop pressurized water reactor with ice condenser containment | ▪ Auxiliary relay rack fire<br>▪ Main control board fire<br>▪ Switchgear 1ETB fire<br>▪ Switchgear 1ETA fire<br>▪ Switchgear 1TC fire |

The following paragraphs characterize each of the above fire scenario types, with the purpose of defining the input space over which the metamodel will be trained. Note that nuclear power plant layout and architectural drawings are typically not available publically due to security concerns. General compartment characteristics, for example range of room dimensions, are therefore assembled based on judgment and discussion with industry experts. The goal is not to represent any specific plant or scenario, but instead for the input space to encompass a broad range of relevant scenarios. Metamodel training on a broad input space will maximize potential applications of the resulting metamodel.

Diesel generator fires can be severe due to a large volume of diesel fuel and lubricating oil. For this reason, each diesel generator is typically located in its own dedicated compartment.

The compartments are rectangular to accommodate the shape of the diesel generator, with estimated dimensions on the order of ~15-20 meters long, ~5-10 meters wide, and ~5-8 meters tall. A large volume diesel fuel oil or lubricating oil fire is so severe that all targets in the compartment would likely be damaged, and therefore fire modeling generally provides little benefit. This scenario is therefore excluded from consideration in defining the scenario input space.

Transient fires are those which initiate on temporary combustible packages and can occur at almost any location. An example might be a plastic cart containing tools and parts required for equipment maintenance. NUREG/CR-6850 (Electric Power Research Institute & U.S. Nuclear Regulatory Commission, 2005) characterizes their potential heat release rate as gamma-distributed with $\alpha = 1.8$, $\beta = 57.4$, and at $98^{th}$ percentile value of 317 kW. Corridor configurations fall outside the range of applicability for CFAST, due to its model assumption that heat accumulates homogeneously and instantaneously across the ceiling. This assumption does not apply to corridors, where there can be a non-negligible delay time for smoke transport from one end of the corridor to the other. Corridor configurations are therefore excluded from the scenario input space; however, the range of transient fire heat release rates is included in the input space.

All three sampled plants identified switchgear room fires as significant. The primary fire sources in switchgear rooms are electrical cabinets. These electrical cabinets can include medium voltage switchgear, motor control centers, low voltage panels, and battery chargers. Note that the bus duct fault identified by the Cooper plant is an explosive electrical event that is not modeled by CFAST, or any other traditional fire model. NUREG-2178 (USNRC & EPRI, 2015) evaluated a series of full scale electrical cabinet fire tests and developed heat release rate probability

density functions based on the type of cabinet, fuel load, and ventilation configuration. The resulting distributions are reproduced here as Table 5, and they have a 98$^{th}$ percentile peak heat release rate range of 45-1,000 kW. Switchgear rooms are generally square or rectangular, with an estimated floor area on the order of ~500-1,000 m$^2$ and a height on the order of ~5-8 meters. NUREG-1934 (McGrattan et al., 2012) includes an example switchgear room with dimensions 26.5 meters long, 18.5 meters wide, and 6.1 meters tall. The example scenario also has a mechanical ventilation system with three 0.5 by 0.6 meter supply ducts, each supplying 0.47 m$^3$/s ventilation, and three 0.5 by 0.6 meter return ducts.

**Table 5.** Cabinet Fire Peak Heat Release Rate Gamma Distributions per NUREG-2178 (USNRC & EPRI, 2015)

| **Cabinet Type** | **Ventilation** | **Fuel Type** | **Default Fuel Load** | | **Low Fuel Load** | | **Very Low Fuel Load** | |
|---|---|---|---|---|---|---|---|---|
| | | | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ |
| Switchgear and Load Centers | Closed | TS, QTP, SIS | 0.32 | 79 | N/A | | | |
| | Closed | TP | 0.99 | 44 | | | | |
| Motor Control Centers and Battery Chargers | Closed | TS, QTP, SIS | 0.36 | 57 | | | | |
| | Closed | TP | 1.21 | 30 | | | | |
| Power Inverters | Closed | TS, QTP, SIS | 0.23 | 111 | | | | |
| | Closed | TP | 0.52 | 73 | | | | |
| Large Enclosures | Closed | TS, QTP, SIS | 0.23 | 223 | 0.23 | 111 | 0.38 | 32 |
| | Closed | TP | 0.52 | 145 | 0.52 | 73 | 0.88 | 21 |
| | Open | TS, QTP, SIS | 0.26 | 365 | 0.26 | 182 | 0.38 | 32 |
| | Open | TP | 0.38 | 428 | 0.38 | 214 | 0.88 | 21 |
| Medium Enclosures | Closed | TS, QTP, SIS | 0.23 | 111 | 0.27 | 51 | 0.88 | 12 |
| | Closed | TP | 0.52 | 73 | 0.52 | 36 | 0.88 | 12 |
| | Open | TS, QTP, SIS | 0.23 | 182 | 0.19 | 92 | 0.88 | 12 |
| | Open | TP | 0.51 | 119 | 0.30 | 72 | 0.88 | 12 |
| Small Enclosures | N/A | All | 0.88 | 12 | N/A | | | |

The station batteries are usually located in dedicated compartments due to their personnel safety hazard as well as the risk of hydrogen accumulation during charging. NUREG/CR-6850 (Electric Power Research Institute & U.S. Nuclear Regulatory Commission, 2005) describes the peak heat release rate of station battery fires as gamma-distributed with $\alpha = 2.0$ and $\beta = 11.7$. Battery room dimensions are estimated to be generally square with a floor area on the order of ~50 $m^2$.

Main control room fires can be particularly challenging for two reasons. First, they can force operators to abandon the control room due to lost tenability. Visibility (smoke obscuration) and heat flux are therefore important CFAST output parameters for main control room fire modeling. It was noted in the validation that CFAST tends to significantly over-predict smoke concentration and under-predict heat flux. Second, fire in the control room can damage control and instrumentation for important plant equipment required for accident mitigation. The primary ignition sources include electrical cabinets and transient fires, and their heat release rates are characterized in the preceding paragraphs. Control rooms can vary greatly in size, depending on whether they support one or two reactor units. They are estimated to be generally square ranging from ~15-40 meters wide/long and ~5-8 meters tall. NUREG-1934 (McGrattan et al., 2012) includes an example control room with dimensions 24.6 meters long, 16.2 meters wide, and 5.2 meters tall. The example scenario also has a mechanical ventilation system capable of 25 air changes per hour with supply equally distributed over six vents and return over two vents.

The radwaste building and pipe chase fires identified for Arkansas Nuclear One are excluded from further consideration due to lack of information. These locations are likely significant due to a very plant-specific configuration. The radwaste building typically does not

contain many safety-related components or cables, and pipe chases usually do not contain many ignition sources.

In summary, the range of input space characterized by examining the most risk-significant fire scenarios at three U.S. nuclear power plants of differing design is provided in Table 6.

**Table 6.** Summary of Input Space Defined by High Risk Fires at Three Sampled Plants

| Parameter | Range |
|---|---|
| Peak Fire Heat Release Rate | 0 - 1,000 kW |
| Compartment Length | 15 - 40 meters |
| Compartment Width | 5 - 40 meters |
| Compartment Height | 5 - 8 meters |
| Ventilation Rate | 0.6 - 1.4 $m^3/s$ |

### 3.2.2 Validated Range of Fire Model Input Space

The NUREG-1824 (Hill et al., 2007) fire model validation was performed against a specific set of full-scale fire tests, which are summarized in Table 7.

**Table 7.** Full-Scale Fire Tests used for Fire Model Verification and Validation

| Fire Test Series | Description | Reference |
|---|---|---|
| Sandia National Laboratory | Test series intended to simulate fire in the main control room. Enclosure was a single room of dimensions 18.3 x 12.2 x 6.1 meters with forced mechanical ventilation. The fire source was propylene gas-fired for the tests used by the validation effort. | NUREG/CR-4681 (USNRC & Sandia National Laboratories, 1987) NUREG/CR-5384 (USNRC & Sandia National Laboratories, 1989) |
| National Bureau of Standards | Setup consisted of two relatively small rooms connected by a corridor. Various configurations of doors open and closed were tested. Fire source was a gas burner located in one of the rooms, with fire sizes of 100, 300, and 500 kW. | NBSIR 88-3752 (NIST, 1988) |
| International Collaborative Fire Model Project (ICFMP) Benchmark Exercise | One series used a relatively large 27 x 14 x 19 meter enclosure. Fire source in each test was a heptane pool fire ranging from 2,000 to 4,000 kW.<br><br>Second series was in a 21.7 x 7.15 x 3.7 meter room. The room was mechanically ventilated, and ventilation conditions were varied between the tests. Fire sizes included 350 kW, 1,000 kW, and 2,000 kW.<br><br>Third series involved a relatively large fire in a relatively small concrete room.<br><br>Fourth series involved the same relatively small concrete enclosure but also contained cable trays. | See NUREG-1824 (Hill et al., 2007) and supporting references |

NUREG-1934 (McGrattan et al., 2012) defines validated ranges of applicability using non-dimensional parameters that characterize important aspects of the fire scenario, such as fire size, compartment size and aspect ratio, ventilation conditions, and target location relative to fire location. The non-dimensional parameters were calculated for each of the fire tests used in the

validation, and they provide one mechanism to assess whether a particular scenario of interest is sufficiently similar to the tested configurations. The validated ranges are reproduced here in Table 8, and the variable definitions are provided in NUREG-1934.

**Table 8.** Fire Model Validated Ranges per NUREG-1934 (McGrattan et al., 2012)

| Non-Dimensional Parameter | Definition | Description | Validated Range |
|---|---|---|---|
| Froude Number | $\dot{Q}^* = \frac{\dot{Q}}{\rho_\infty c_p T_\infty D^2 \sqrt{gD}}$ | Measure of the buoyant strength of the fire plume | 0.4-2.4 |
| Flame Length Ratio | $\frac{H_f + L_f}{H_c}$<br>$\frac{L_f}{D} = 3.7\dot{Q}^{*2/5} - 1.02$ | Measure of the flame height relative to the ceiling height | 0.2-1.0 |
| Ceiling Jet Distance Ratio | $\frac{r_{cj}}{H_c - H_f}$ | Characterizes the location of interest within the ceiling jet relative to the plume height | 1.2-1.7 |
| Equivalence Ratio | $\varphi = \frac{\dot{Q}}{\Delta H_{O_2} \dot{m}_{O_2}}$<br>$\dot{m}_{O_2} = \frac{0.23 A_0 \sqrt{H_0}}{2}$ (Natural)<br>$\dot{m}_{O_2} = 0.23\rho\dot{V}$ (Forced) | Measure of the fuel pyrolysis rate relative to the oxygen supply (ventilation) rate | 0.04-0.6 |
| Compartment Aspect Ratio | $\frac{L}{H_c} or \frac{W}{H_c}$ | Characterizes the extent to which the compartment deviates from a cube | 0.6-5.7 |
| Radial Distance Ratio | $\frac{r_{rad}}{D}$ | Characterizes the target radial proximity to the fire relative to the fire diameter, where flame radiation is the damage mechanism of concern. | 2.2-5.7 |

### 3.2.3 Fire Scenario Definition for RAVEN Application

This section defines the fire scenario parameter space over which RAVEN will be exercised. The input space surrounds fire occurring in a switchgear room of a nuclear power facility. The

switchgear room is selected because of its risk significance to currently operating plants, which rely heavily on electric power for accident mitigation.

Table 9 summarizes the resulting fire scenario definition. Note that while the input space is defined surrounding a switchgear room, the analysis can be considered generic: it is applicable to any fire scenario that falls within the defined input space. For example, this analysis might be applicable to a couch fire whose heat release rate profile and room dimensions are within the defined input space. Or, the analysis might be applicable to an electrical cabinet fire originating in a non-nuclear facility.

**Table 9.** Fire Scenario Input Space over which RAVEN will be Exercised

(Selected to Encompass Typical Switchgear Room Fire Scenarios)

| Parameter | Description | Definition | Notes |
|---|---|---|---|
| *Compartment Characteristics* | | | |
| $L$ | Compartment length (m) | $U(a=10, b=35)$ | N/A |
| $W$ | Compartment width (m) | $U(a=10, b=35)$ | N/A |
| $H_c$ | Ceiling height (m) | $U(a=5, b=10)$ | N/A |
| $\dot{V}$ | Ventilation rate ($m^3/s$ per cubic meter of room volume) | $U(a=0.00047, b=0.00189)$ | Review of typical switchgear forced ventilation rates per room volume identifies a range of 1-4 $cfm/m^3$ (or 1.7-6.8 room changes per hour), which translates to 0.0047-0.00189 $m^3/s$ per cubic meter of room volume. This parameter is scaled by room volume because the total heat load of the operating electrical equipment is estimated to roughly scale by room volume. |

**Table 9 (Continued).**

| Parameter | Description | Definition | Notes |
|---|---|---|---|
| $A_0$ | Ventilation opening area ($m^2$) | $4 \times 0.05 = 0.2$ | Small natural leakage area specified to represent leakage underneath doors and other small leaks that may be present along the compartment boundaries. This leakage area is specified at floor level and divided evenly to each of the four walls. Note that specifying at least some leakage is important for numerical stability of the CFAST calculation. |
| $H_0$ | Distance from floor to center of ventilation opening (m) | 0.0125 | |
| $T_\infty$ | Ambient temperature (K) | *U*(*a*=297, *b*=311) | Estimated temperature range of 297 K (75 °F) to 303 K (85 °F) for a switchgear room. |
| $\rho_\infty$ | Ambient air density ($kg/m^3$) | $\rho_\infty(T_\infty)$ | Air density is a function of sampled ambient temperature. This value is calculated internally by CFAST. |
| $c_p$ | Ambient air specific heat (kJ/kg-K) | $c_p(T_\infty)$ | Air specific heat is a function of sampled ambient temperature. This value is calculated internally by CFAST. |
| $k_w$ | Thermal conductivity of wall material (W/m-K) | 1.75 | CFAST default value for normal weight concrete. Consistent with (SFPE & NFPA, 2002). |
| $\rho_w$ | Density of wall material ($kg/m^3$) | 2,200 | CFAST default value for normal weight concrete. Consistent with (SFPE & NFPA, 2002). |
| $c_w$ | Specific heat of wall material (kJ/kg-K) | 1.0 | CFAST default value for normal weight concrete. Consistent with (SFPE & NFPA, 2002). |
| $t_w$ | Wall thickness (m) | 0.15 | Six inch thick wall |

**Table 9 (Continued).**

| Parameter | Description | Definition | Notes |
|---|---|---|---|
| *Fire Characteristics* | | | |
| $H_f$ | Height of fire above floor (m) | $U(a = 0, b = 0.9) \times Compartment\ Height$ | Height of fire above floor level ranging uniformly between 0% and 90% of ceiling height |
| $\dot{Q}_p$ | Peak fire heat release rate (kW) | Uniformly distributed over range depicted in Figure 5 | Encompasses range of electrical cabinet fire heat release rate profiles, including contribution from secondary burning of overhead cable trays. |
| $D_f$ | Fire diameter (m) | $D = \left(\frac{\dot{Q}}{\dot{Q}^* \rho_\infty c_p T_\infty \sqrt{g}}\right)^{2/5}$ | Fire diameter estimated based on mid-point of validated Froude number range ($\dot{Q}^* = 1.0$), using the scenario peak heat release rate, and with the nominal values: $\rho_\infty = 1.2\ kg/m^3$ $c_p = 1.05\ kJ/(kg \cdot K)$ $T_\infty = 304\ K$ $g = 9.81\ m/s^2$ This diameter term is used to calculate the fire surface area in CFAST, which varies as a function of time and heat release rate. Note that 0.01 $m^2$ is used as a lower bound, since CFAST does not allow fire objects with no surface area, even when their heat release rate is temporarily zero (0) kW. |
| *Constants* | | | |
| $g$ | Acceleration of gravity ($m/s^2$) | 9.81 | N/A |
| $\Delta H_{O_2}$ | Energy generated per oxygen consumed (kJ/kg) | 13,100 | Point estimate per (SFPE & NFPA, 2002) |

Figure 5 depicts the range of postulated electrical cabinet fire heat release rate profiles, including contribution from secondary burning of overhead cable trays. These profiles were developed to be consistent with scenarios that might be found in typical switchgear room fire probabilistic risk assessments.

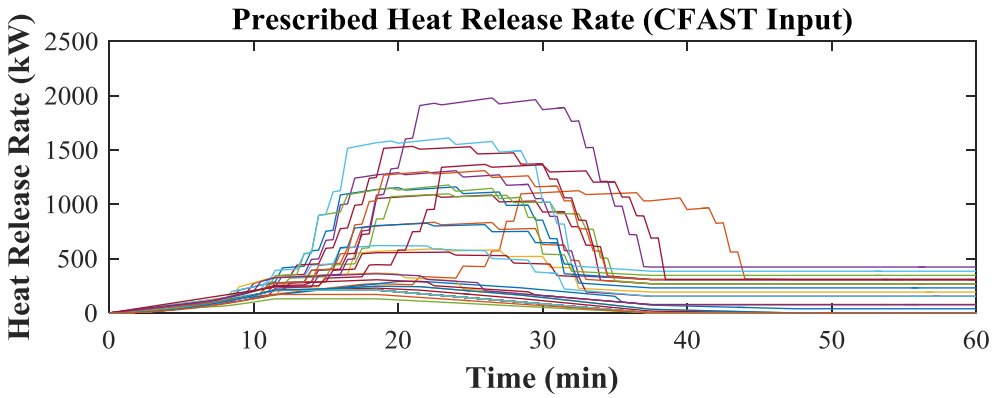


**Figure 5.** Range of Fire Heat Release Rate Profiles to be executed by RAVEN

Note that the heat release rate profile for a given fire scenario is not a fundamental parameter; it instead is a function of the cabinet characteristics, the number and configuration of overhead cable trays to which the fire may propagate, and the number of adjacent cabinets to which the fire may propagate. Figure 6 depicts the postulated electrical cabinet fire scenario with overhead cable trays and adjacent cabinets.

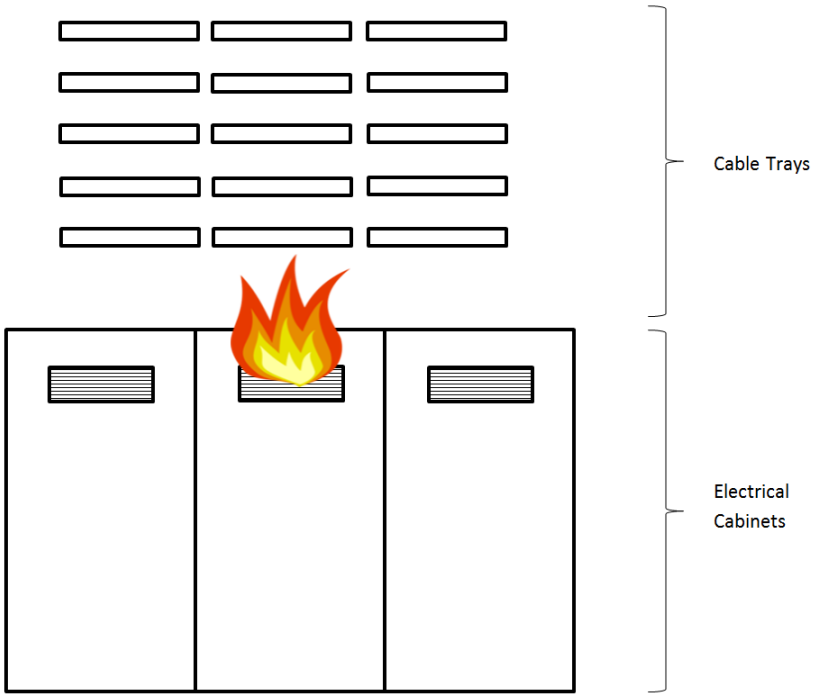


**Figure 6.** Postulated Fire Scenario: Electrical Cabinet Fire

Propagating to Overhead Cable Trays and Adjacent Electrical Cabinets

Figure 7 is a CFAST rendering of the range of compartment sizes that will be exercised by RAVEN. Per Table 9, compartment length will range between 10 and 35 meters, compartment width will range between 10 and 35 meters, and compartment height will range between 5 and 10 meters. The compartment dimensions will randomly vary within these ranges for each sample used to train the metamodel. Other parameters, such as fire heat release rate, will also be varied over their credible ranges in accordance with Table 9.

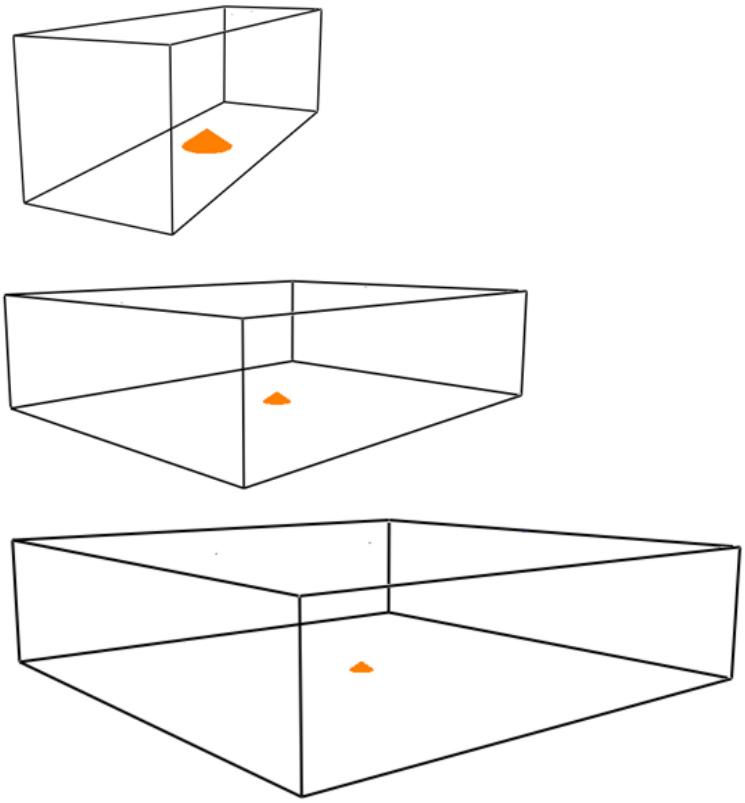

**Figure 7.** CFAST Rendering of Range of Compartment Shapes to be Evaluated by RAVEN

## 3.3 RAVEN-CFAST MODEL SETUP

The RAVEN analysis is defined and coordinated by a base XML file. This file defines the input distributions, samples those distributions, calls a Python interface to execute CFAST for each sample, and assembles the input/output results for each sample. The XML file also coordinates post-processing functions such as summary statistics, visualizations, and ROM development. The Python interface between RAVEN and CFAST generates a text-based CFAST input file using the sampled parameters, and it returns a command line that runs CFAST with the generated

input file. Appendix A provides an annotated version of the XML and Python code comprising the RAVEN-CFAST model

## 3.4 INPUT AND OUTPUT PARAMETERS OF INTEREST

The input (predictor) parameters of interest include all variable inputs whose values are sampled during the RAVEN calculation. These include, for example, fire heat release rate and compartment dimensions. Note that the heat release rate input parameter varies with time throughout the simulation, and it is a function of several more fundamental parameters including the cabinet type, number and configuration of overhead cable trays, and the number of immediately adjacent cabinets.

The output (response) parameters of interest could include any calculated outputs of the CFAST model, for example upper layer temperature, upper layer height, smoke density, or the heat flux to particular target locations. Note that these parameters all vary with time. The maximum upper layer temperature and the time at which the maximum upper layer temperature is achieved are the primary output parameters of interest. Table 10 summarizes the input and output parameters of interest for this analysis.

**Table 10.** Summary of Input and Output Parameters of Interest

| Input (Predictor) Parameters | Output (Response) Parameter |
|---|---|
| Fire Heat Release Rate (varies with time and is a function of the overhead cable tray configuration and number of adjacent cabinets) | Maximum Upper Layer Temperature<br><br>Time at which Maximum Upper Layer Temperature is Achieved |
| Compartment Length | |
| Compartment Width | |
| Compartment Height | |
| Ambient Temperature | |
| Ventilation Rate | |
| Height of Fire above Floor | |

## 3.5 FULL GRID SAMPLING OF THE INPUT SPACE

Table 11 summarizes an arbitrarily large full grid sampling plan to generate a population of data against which the reduced order models available in RAVEN can be tested in the fire modeling context. A full grid sampler is selected, as opposed to random samplers such as Monte Carlo and Latin Hypercube, to ensure the entire input space including its boundaries is considered.

Note that a full grid is quite inefficient and not what this thesis ultimately recommends for metamodel development, especially for applications where computational expense of generating the data is high. In such cases a more intelligent sampling, such as adaptive, would be recommended. In addition, it is later discussed that sufficient metamodel accuracy was achieved with 50,000 - 100,000 samples, as opposed to 675,000 samples generated by the full grid. The

full grid was simply used as an initial step to generate a large population of data to support experimenting with the various reduced order model types available in RAVEN.

**Table 11.** Full Grid Sampling Plan to Create Population of Data against which to Test RAVEN ROM Capabilities

| Parameter | Range | Discretization |
|---|---|---|
| Length | 10-35 meters | 10 points, 9 breakpoints, increments of 2.5 meters |
| Width | 10-35 meters | 10 points, 9 breakpoints, increments of 2.5 meters |
| Height | 5-10 meters | 5 points, 4 breakpoints, increments of 1 meter |
| Fire Heat Release Rate Profile | 30 unique profiles | 30 points, 29 breakpoints, all heat release rate profiles used |
| Ambient Temperature | 297-311 Kelvin | 3 points, 2 breakpoints, increments of 4.7 Kelvin |
| Fire Height | 0-90% of room height | 5 points, 4 breakpoints, increments of 0.18 meters |
| Ventilation Rate per Room Volume | 0.00047-0.00189 ($m^3$/s per $m^3$ of room volume) | 3 points, 2 breakpoints, increments of 0.000473 $m^3$/s per $m^3$ of room volume |

The sample plan was divided into 15 batches of 45,000 CFAST runs such that the results could be monitored and any problems identified as the calculation progressed. The heat release rate parameter was selected to facilitate this sample plan subdivision. A total of 30 unique heat release rate profiles were included in the full sample plan. Each batch therefore evaluated two heat release rate profiles and the full grid of all other parameters. Note that in the first batch, three profiles were attempted, but two were ultimately selected for each due to computational (disk space and run time) limitations.

**Table 12.** Subdivision of Full Grid into Batches using the Heat Release Rate Parameter

| Run Batch | Covered HRR Indices | RAVEN Syntax for HRR Grid Definition |
|---|---|---|
| 1 | 1, 2, 3 | <grid construction="equal" steps="2" type="CDF">0.000 0.067</grid> |
| 2 | 4, 5 | <grid construction="equal" steps="1" type="CDF">0.101 0.165</grid> |
| 3 | 6, 7 | <grid construction="equal" steps="1" type="CDF">0.170 0.230</grid> |
| 4 | 8, 9 | <grid construction="equal" steps="1" type="CDF">0.25 0.29</grid> |
| 5 | 10, 11 | <grid construction="equal" steps="1" type="CDF">0.32 0.35</grid> |
| 6 | 12, 13 | <grid construction="equal" steps="1" type="CDF">0.38 0.42</grid> |
| 7 | 14, 15 | <grid construction="equal" steps="1" type="CDF">0.45 0.48</grid> |
| 8 | 16, 17 | <grid construction="equal" steps="1" type="CDF">0.52 0.55</grid> |
| 9 | 18, 19 | <grid construction="equal" steps="1" type="CDF">0.58 0.62</grid> |
| 10 | 20, 21 | <grid construction="equal" steps="1" type="CDF">0.65 0.68</grid> |
| 11 | 22, 23 | <grid construction="equal" steps="1" type="CDF">0.72 0.75</grid> |
| 12 | 24, 25 | <grid construction="equal" steps="1" type="CDF">0.78 0.82</grid> |
| 13 | 26, 27 | <grid construction="equal" steps="1" type="CDF">0.85 0.88</grid> |
| 14 | 28, 29 | <grid construction="equal" steps="1" type="CDF">0.92 0.95</grid> |
| 15 | 30 | <grid construction="equal" steps="1" type="CDF">0.98 0.99</grid> |

The RAVEN analysis was run on a Hewlett Packard Z640 engineering workstation with the following specifications:

- Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz 2.20 GHz (2 processors)
- 64 GB Memory
- 64-bit Windows 7 Enterprise operating system

The full grid sampling generated 675,000 unique CFAST runs across the input space. These runs were divided into 15 batches, which in total took approximately 15 days of computer run time (using two processors) and generated nine (9) terabytes of CFAST input/output files. RAVEN consolidated the data of interest into 15 HDF5 databases totaling 30 gigabytes. Each batch of 45,000 runs reliably took 24 hours (using two processors) of computation time. The

input/output files generated, including the results data consolidation into HDF5 databases, were consistent in terms of disk space consumption (usually to the kilobyte).

Figure 8 illustrates full grid sampled fire scenario input space, consistent with the sample design in Table 11.

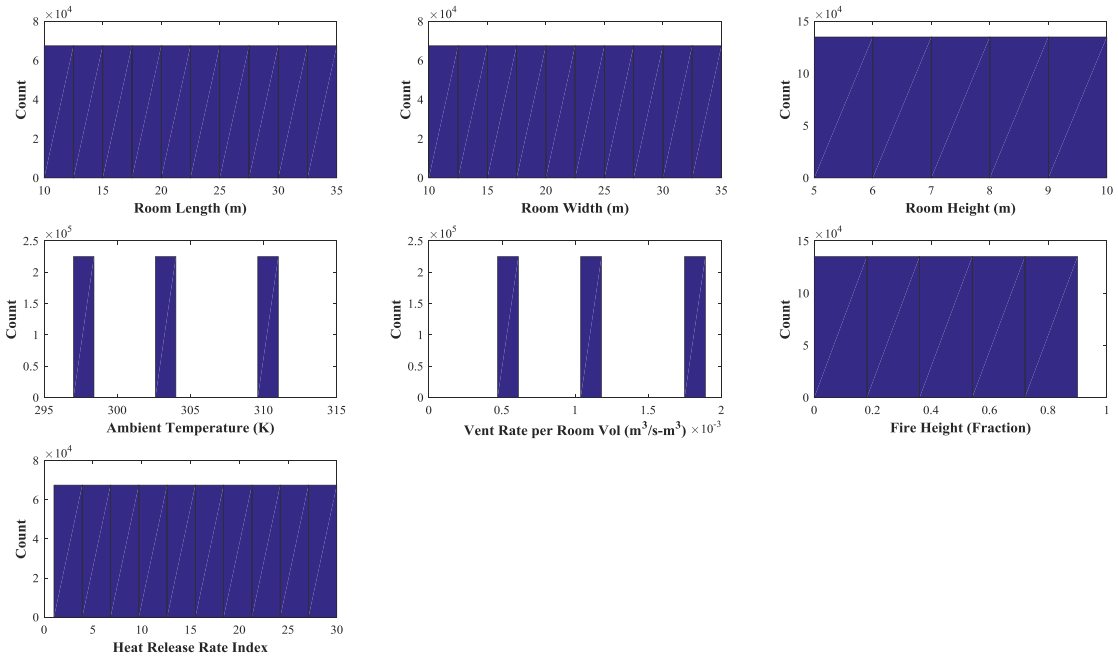


**Figure 8.** Grid Sampled Fire Scenario Input Space

Note that the heat release rate indices each refer to a unique heat release rate profile that varies with time (see Figure 5 for the heat release rate profiles). Each of these profiles is a function of the number of cabinets adjacent to the originating cabinet, as well as the number and configuration of overhead cable trays, to which fire can propagate. Figure 9 provides histograms of these factors that influence the heat release rate profiles.

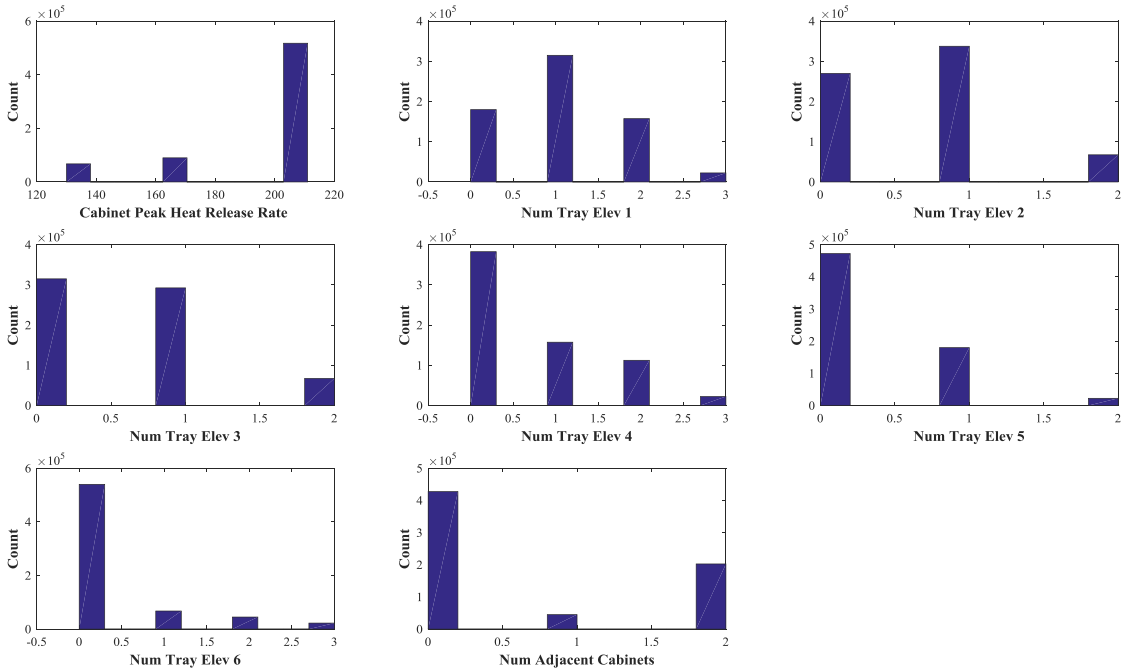


**Figure 9.** Histograms of the Factors upon which the Heat Release Rate Profiles are Based

Figure 9 illustrates that while the heat release rate profiles were uniformly sampled, the physical and geometric factors that comprise the heat release rate profiles are not uniformly distributed. The heat release rate profiles were selected to be representative of actual plant configurations, and the non-uniformity of the underlying physical and geometric factors is reflective of typical plant design and layout.

The benefit of including these parameters is that they are scalar, whereas the heat release rate profiles are time series. A reduced order model based on scalar quantities is easier to implement than one relying upon time series.

This also highlights a benefit of machine learning in that it can emulate relationships quantified across multiple individual models. Figure 10 illustrates that this application of

machine learning involves emulating two fire modes: 1) a spreadsheet-based model that estimates a heat release rate profile given an electrical cabinet fire, with overhead cable trays and adjacent cabinets to which the fire can propagate, and 2) a CFAST fire model that estimates upper layer temperature based on the heat release rate profile and compartment characteristics.

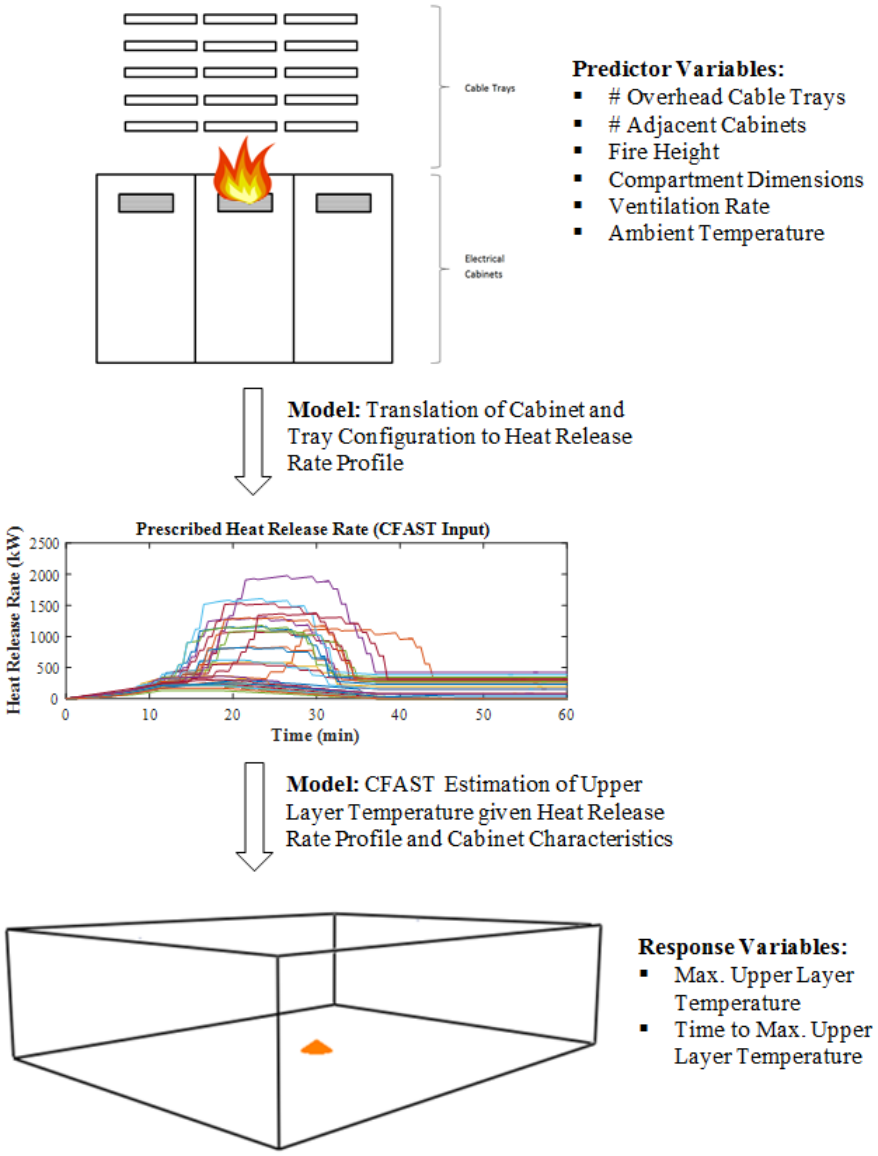


**Figure 10.** Illustration of the Two Models over which Machine Learning is Exercised

## 3.6 SIMULATION RESULTS

The following figures summarize various aspects of the CFAST simulation results. These figures were generated using 500 random samples (representing 500 unique CFAST runs) of the 675,000 case population generated in Section 3.5.

Figure 11 compares the prescribed (CFAST input) against the realized (CFAST output) heat release rate profiles. Discrepancies between the prescribed and realized profiles represent sampled configurations where the room volume and ventilation conditions provide insufficient oxygen to support the full heat release rate. In these cases, the heat release rate typically grows, reaches its peak value, and burns at a steady state until oxygen required to support combustion is consumed or the upper layer descends below the fire elevation, at which point the heat release rate drops below its prescribed profile. The fire does not fully extinguish, since the ventilation provided by leakage (for example under the doors) is sufficient to support some level of burning.

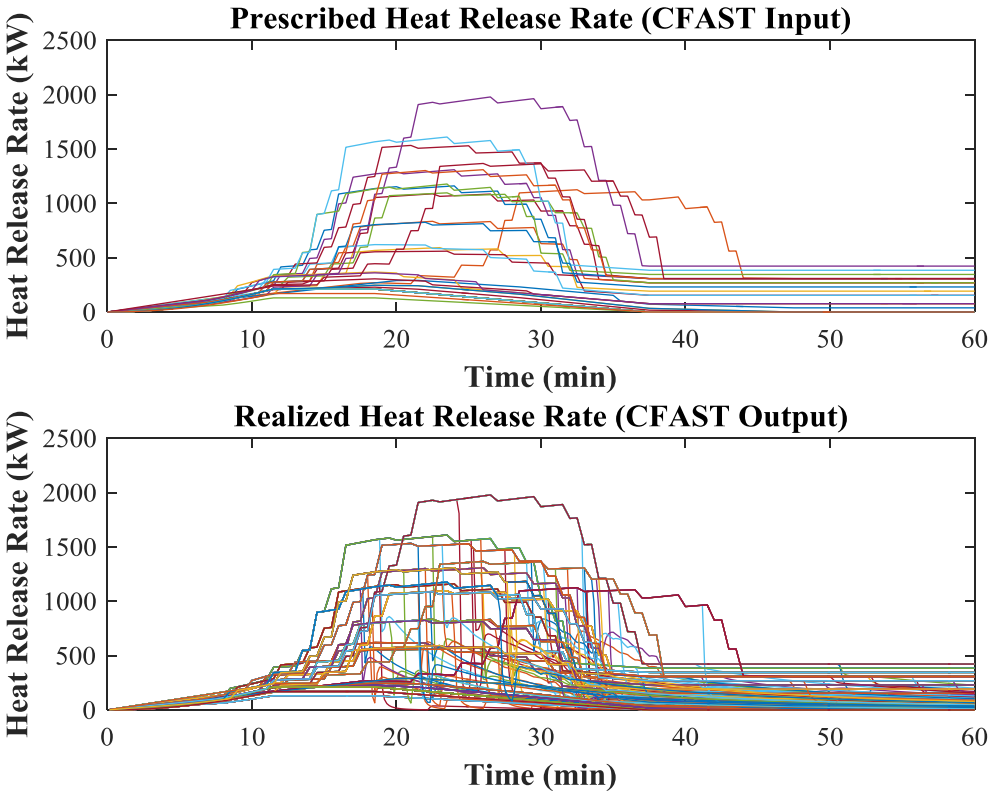


**Figure 11.** Prescribed versus Realized Heat Release Rates

Figure 12 and Figure 13 depict the upper and lower layer, respectively, temperature profiles calculated by CFAST for each RAVEN-generated sample of the input space. The maximum upper layer temperature achieved in this 500 run simulation is about 200 °C, and the maximum lower layer temperature is about 100 °C. Both the upper and lower layers follow similar temperature profiles, characterized by an initial growth period (generally about 20 minutes consistent with the heat release rate profiles), followed by steady state burning, and generally a gradual decay back toward ambient temperature. Sometimes the temperature decay is abrupt, and this is due to either oxygen consumption or the upper layer enveloping the flame such that its full heat release rate is no longer supported.

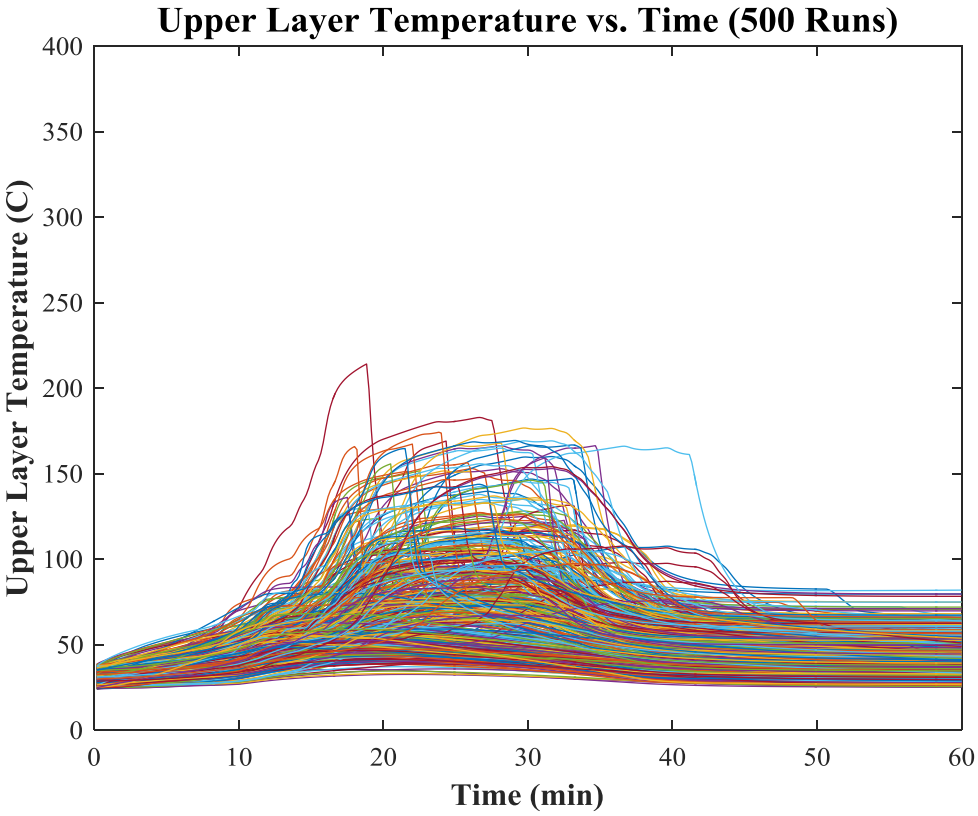


**Figure 12.** Upper Layer Temperature Profiles Calculated by CFAST

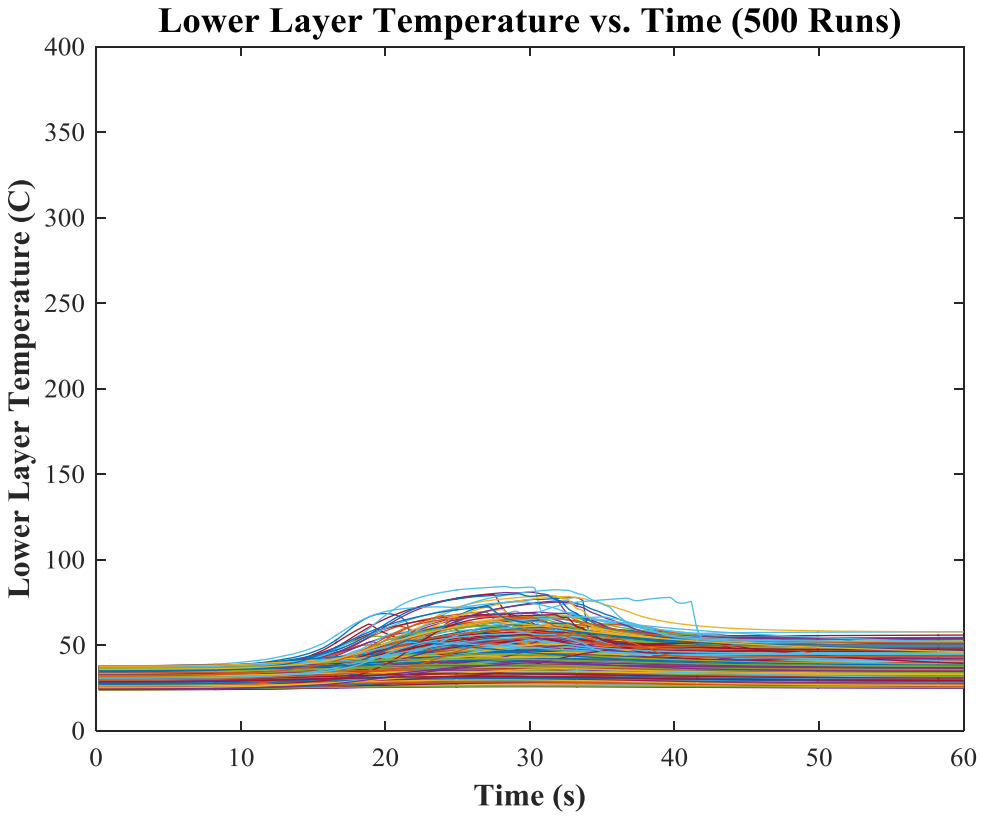


**Figure 13.** Lower Layer Temperature Profiles Calculated by CFAST

Figure 14 shows the calculated upper layer heights as a function of time. All upper layer heights initially start at the ceiling, and the variation in starting heights shown in the figure is due to variation in sampled compartment heights. The upper layer height then descends for the first approximately 20 minutes of burning, consistent with when the prescribed heat release rate profiles reach their peaks. At that point the upper layer height tends to stabilize through the remainder of the simulation.

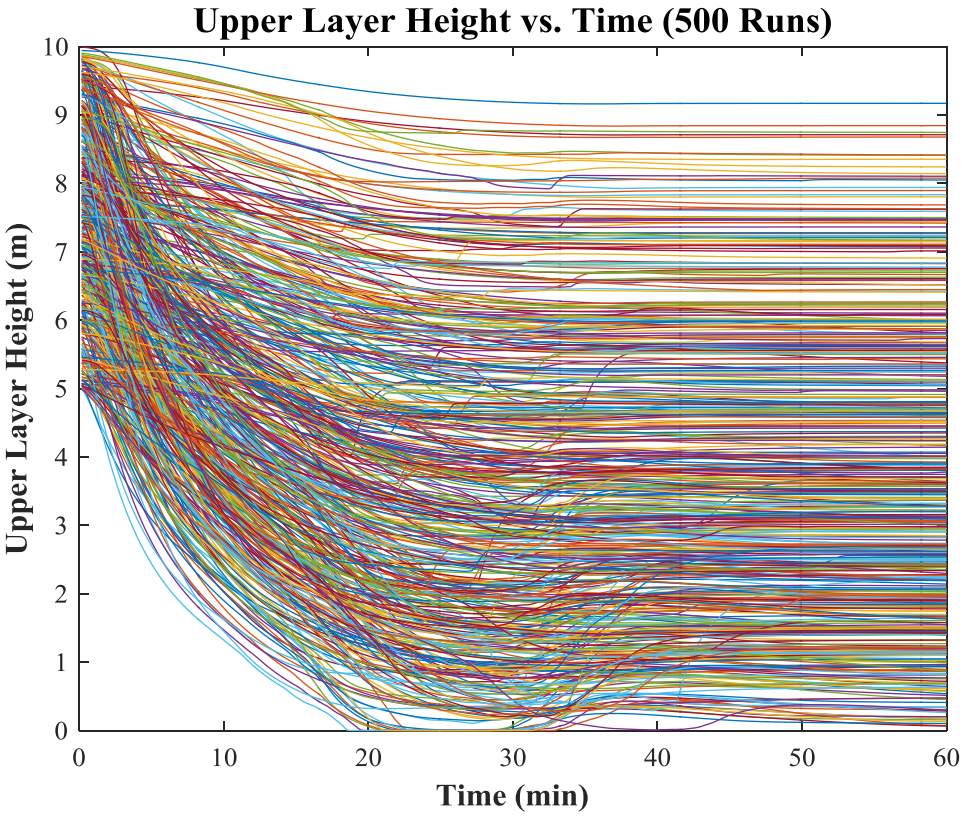


**Figure 14.** Upper Layer Height Profiles Calculated by CFAST

Figure 15 illustrates the compartment pressure profiles for 500 randomly sampled CFAST runs. During the fire growth period, generally from zero (0) to 20 minutes, compartment pressure increases as the fire deposits energy into the compartment. Compartment pressure becomes negative as the fire decays and room temperature decreases. The abrupt negative pressure spikes occur when the fire nearly extinguishes due to oxygen consumption or the upper layer descending below the fire elevation.

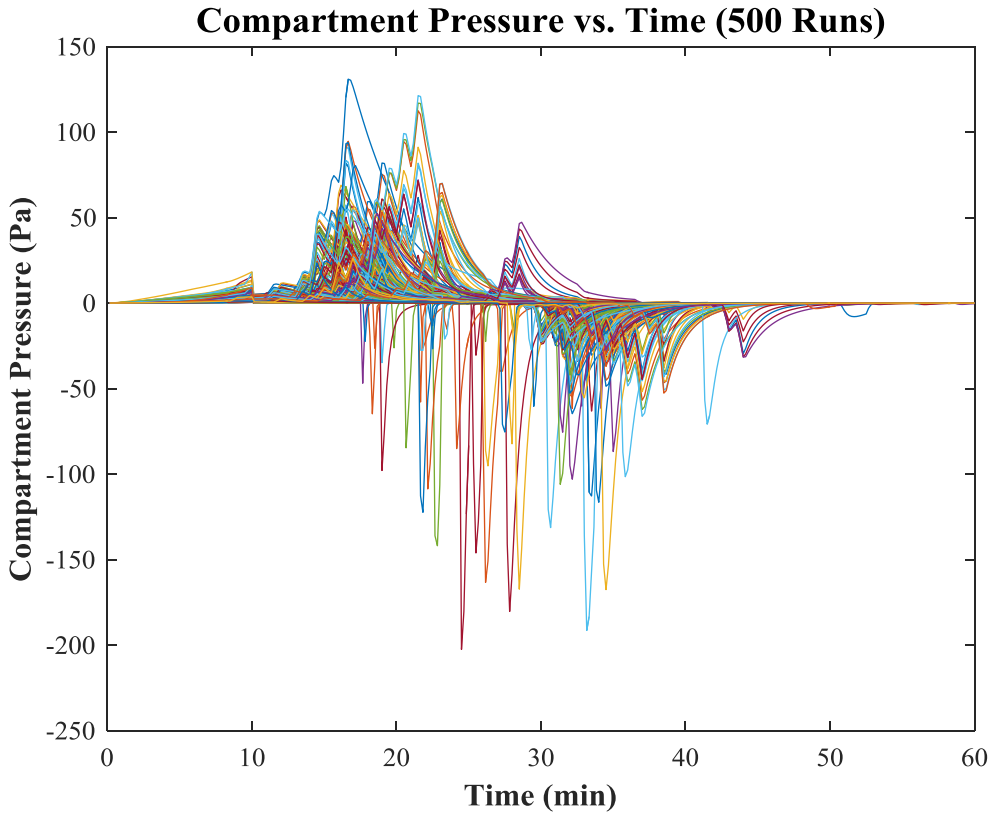


**Figure 15.** Compartment Pressure Profiles Calculated by CFAST

Figure 16 illustrates the calculated optical density profiles. Optical density is a measure of light obscuration due to smoke. The figure shows that optical density tends to increase along with the heat release rate profile until a peak is reached, generally at about 20 minutes, at which point it stabilizes through the remainder of the simulation.

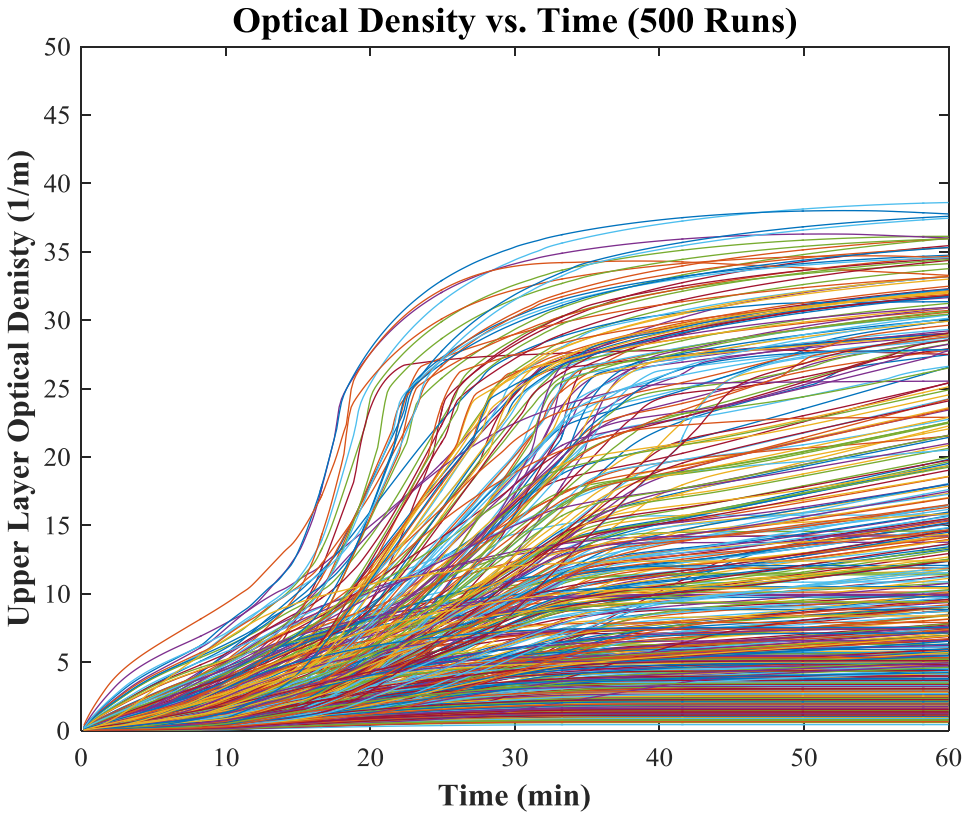


**Figure 16.** Upper Layer Optical Density Profiles Calculated by CFAST

Figure 17 shows the flame height profile for each of the 500 sampled cases. Consistent with other calculated parameters, flame height closely follows the heat release rate profile. Flame height is proportional, although non-linearly, to fire heat release rate.

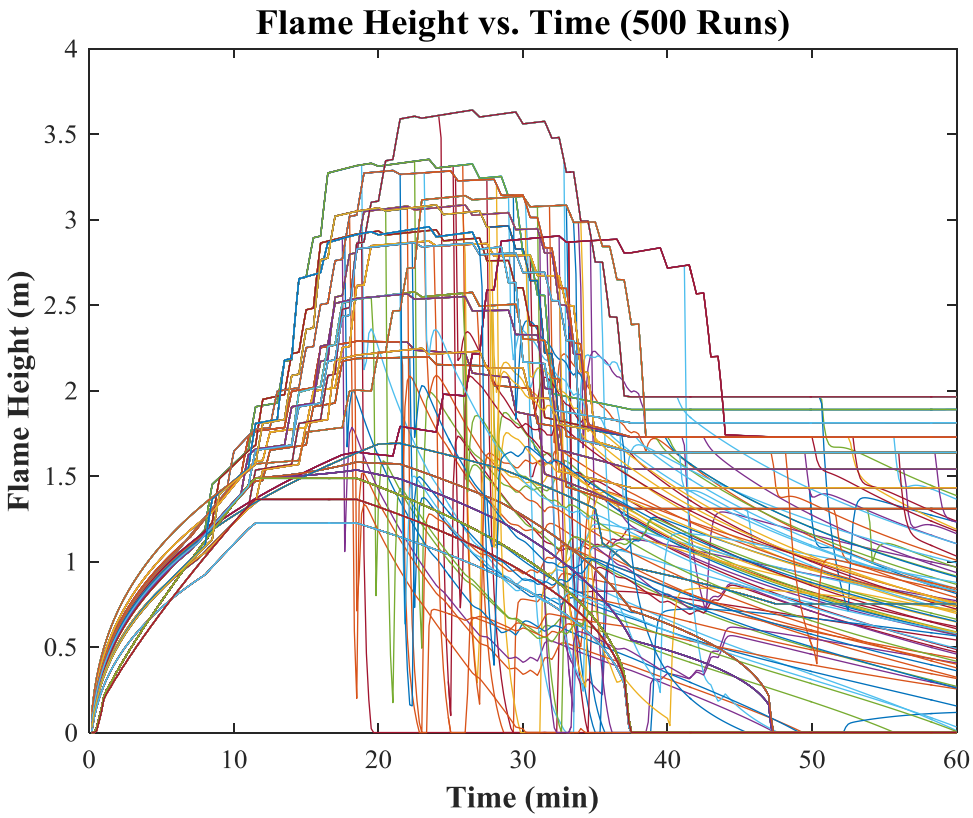


**Figure 17.** Flame Height Profiles Calculated by CFAST

Figure 18 provides histograms of the two primary response variables: maximum upper layer temperature and time to maximum upper layer temperature. These histograms include all 675,000 samples from the population generated in Section 3.5. It is curious that the time to maximum upper layer temperature does not follow the same distribution shape as the maximum upper layer temperature itself. The time to maximum upper layer temperature histogram also includes some apparent discontinuities. This timing is strongly related to the heat release rate timing, and because the selected 30 unique heat release rate profiles are not uniformly distributed (they represent actual plant design and layout), it is reasonable that the time to reach maximum upper layer temperature histogram has some texture. The histogram for maximum upper layer

temperature is “smoother” because there is less variation in the HRR profile maximum values compared to the HRR profile timing.

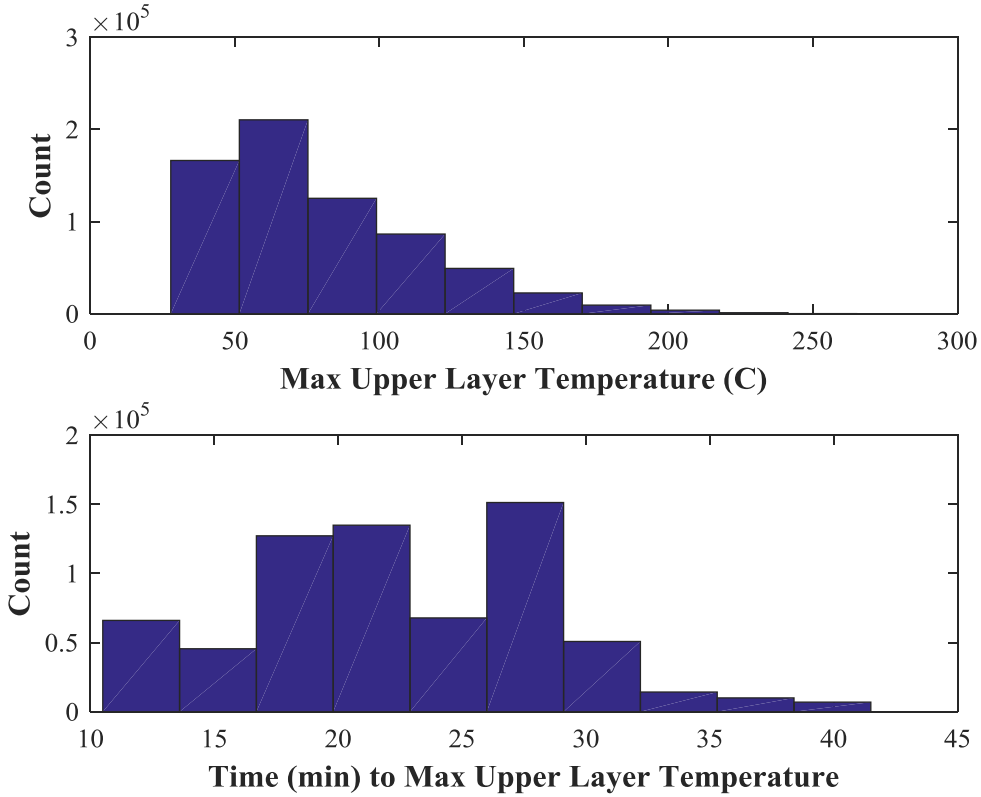


**Figure 18.** Response Variable Histograms

## 3.7 DATA PREPARATION FOR METAMODELING

### 3.7.1 Consolidating the Data

The full grid sampling described in Section 3.5 executed 675,000 runs of the CFAST fire model, with each run modeling a unique sample of the uncertain input space. The input and output data

for all 675,000 runs were consolidated into a single comma separated variable (.csv) file, where each row represents one run, and the columns represent corresponding predictor and response variable values. Table 13 describes each column in the consolidated .csv. Refer to Section 3.5 for parameter ranges and histograms of predictor and response variables.

**Table 13.** Description of each Column in the Consolidated .csv File used for Metamodel Training and Testing

| Column Name | Predictor or Response? | Description |
|---|---|---|
| hrr | predictor | Value sampled from uniform distribution ranging between 0.5 and 30.5. |
| hrrIndex | predictor | 'hrr' sampled value rounded to nearest integer, resulting in a range of 1 to 30. Each 'hrrindex' represents a unique heat release rate profile. |
| hrrPDF | predictor | 'hrrPDF' takes on one of three values: 130, 170, and 211. Each value represents the peak heat release rate to which the originating cabinet fire grows. |
| hrrAdj | predictor | 'hrrAdj' takes one of three values: 0, 1, 2. Each value represents the number of adjacent cabinets |
| hrrTray1 | predictor | 'hrrTray1' includes the following range of values: 0, 1, 2, 3. It represents the number of cable trays at the first elevation in the overhead tray matrix (see Figure 6). |
| hrrTray2 | predictor | 'hrrTray2' includes the following range of values: 0, 1, 2. It represents the number of cable trays at the second elevation in the overhead tray matrix (see Figure 6). |
| hrrTray3 | predictor | 'hrrTray3' includes the following range of values: 0, 1, 2. It represents the number of cable trays at the third elevation in the overhead tray matrix (see Figure 6). |
| hrrTray4 | predictor | 'hrrTray4' includes the following range of values: 0, 1, 2, 3, 4. It represents the number of cable trays at the fourth elevation in the overhead tray matrix (see Figure 6). |
| hrrTray5 | predictor | 'hrrTray5' includes the following range of values: 0, 1, 2. It represents the number of cable trays at the fifth elevation in the overhead tray matrix (see Figure 6). |
| hrrTray6 | predictor | 'hrrTray6' includes the following range of values: 0, 1, 2, 3. It represents the number of cable trays at the sixth elevation in the overhead tray matrix (see Figure 6). |
| length | predictor | Compartment length taking one of ten evenly distributed values between 10 and 35 meters. |

**Table 13 (Continued).**

| Column Name | Predictor or Response? | Description |
|---|---|---|
| width | predictor | Compartment width taking one of ten evenly distributed values between 10 and 35 meters. |
| height | predictor | Compartment width taking one of five evenly distributed values between 5 and 10 meters. |
| floorArea | predictor | Compartment floor area calculated as product of length and width. |
| fireHeightFract | predictor | Fraction of compartment height at which the fire is located, taking one of five evenly distributed values between 0.0 and 0.9. |
| fireHeight | predictor | Elevation of the base of the fire, calculated as the 'fireHeightFract' multiplied by compartment 'height'. |
| ventPerVol | predictor | Compartment ventilation rate per room volume ($m^3$/s per $m^3$ of room volume) taking one of three values: 0.000473, 0.00118, 0.00189. The ventilation rate used by CFAST is calculated as the product of the 'ventPerVol', compartment 'length', compartment 'width', and compartment 'height' terms. |
| tempAmb | predictor | Compartment ambient temperature (Kelvin) at the beginning of the simulation, taking on one of three values: 297, 304, 311. |
| maxULT | response | CFAST-estimated maximum upper layer temperature (Celsius). |
| timeToMaxULT | response | CFAST-estimated time to achieve maximum upper layer temperature (Celsius). |

### 3.7.2 Feature Selection

Several predictor variables are eliminated from the metamodeling exercise because they have significant overlap with other, more fundamental parameters. The primary benefit of eliminating redundant parameters is improved efficiency in the training, testing, and ultimately application of the resulting metamodel.

‘hrr’ and ‘hrrIndex’ are index parameters that identify which of the 30 unique heat release rate profiles to include in a given CFAST run. The top panel in Figure 11 depicts each heat release rate profile available for the simulation. However, each profile is actually the result of a spreadsheet-based model that combines the heat release rates of the originating cabinet, propagation to any immediately adjacent cabinets, and propagation through the overhead cable tray configuration. This is the step depicted between the first and second panels in Figure 10. The ‘hrr’ and ‘hrrIndex’ are therefore eliminated as they are redundant with the following more fundamental predictor variables: ‘hrrPDF’, ‘hrrAdj’, ‘hrrTray1’, ‘hrrTray2’, ‘hrrTray3’, ‘hrrTray4’, ‘hrrTray5’, and ‘hrrTray6’.

‘floorArea’ is calculated as the product of compartment ‘length’ and ‘width’. The ‘floor area’ parameter is not used directly by the RAVEN-CFAST simulation. However, because CFAST models the upper layer as homogeneous, the estimated upper layer temperature is a function of the floor area and is independent of the compartment aspect ratio. The ‘length’ and ‘width’ parameters are therefore eliminated, and the ‘floorArea’ parameter is retained.

‘fireHeightFract’ is a random variable taking values between 0.0 and 0.9, and it represents the fraction of compartment height at which the fire is located. The ‘fireHeight’ parameter is the product of ‘fireHeightFract’ and the compartment ‘height’. Note that ‘fireHeightFract’ was included in the RAVEN model only as a sampling convenience, as an intermediate variable to generate sampled fire heights. In this exercise the ‘fireHeightFract’ parameter is eliminated, and ‘fireHeight’ is retained as it may be a more intuitive parameter for future applications of metamodels resulting from this work.

Figure 19 depicts the correlation between all selected predictor and response variables. Note that the two response variables are ‘maxULT’ and ‘timeToMaxULT’, and the remainder of

the plotted parameters are predictor variables that have been retained for the metamodeling analysis.

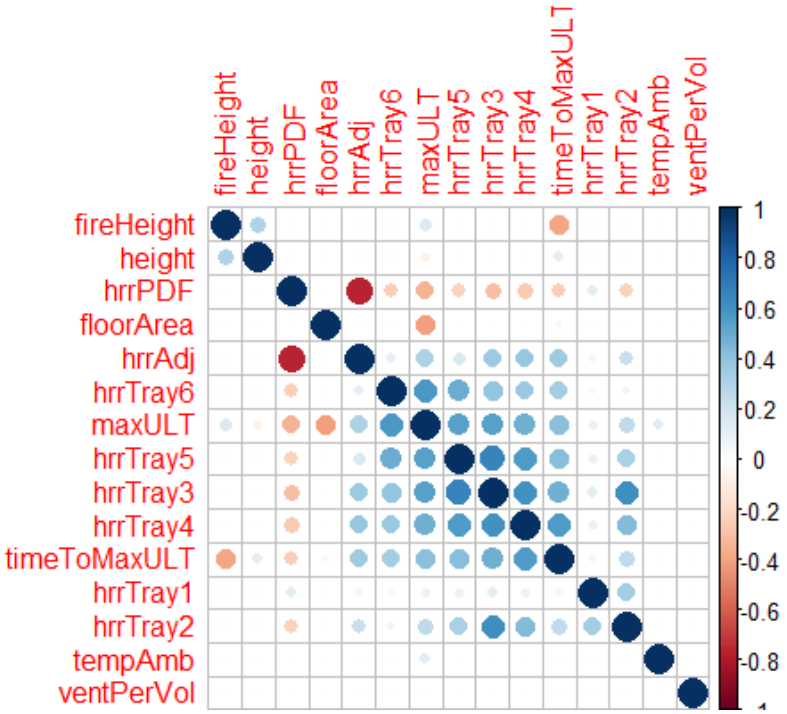


**Figure 19.** Correlation Plot between all Predictor and Response Variables

There is strong negative correlation between the 'hrrAdj' and 'hrrPDF' parameters. The 'hrrAdj' parameter represents the number of cabinets immediately adjacent to the originating cabinet, taking discrete values of zero (0), one (1), and two (2). The 'hrrPDF' parameter represents the peak fire heat release rate that the originating cabinet achieves, taking discrete values of 130, 170, and 211 kW. Both factors are functions of cabinet type and design. Their negative correlation simply indicates that larger cabinets tend to be located in isolation, whereas smaller cabinets tend to be grouped together. This strong correlation suggests one of the two parameters could be removed from the model without losing predictive accuracy; however, both

are retained because this correlation could be unique to the particular plant to which the models are trained, and it is possible that the correlation may not be as strong for other plant designs or vintages. Retaining both parameters therefore broadens applicability of the resulting model.

There is moderate negative correlation between ‘floorArea’ and the response parameter ‘maxULT’. The ‘floorArea’ is directly proportional to the upper layer volume available to dissipate fire-generated heat. Fixing all other parameter values, larger values of ‘floorArea’ lead to larger upper layer volumes, and lower average upper layer temperatures. The ‘floorArea’ term is not perfectly correlated because upper layer volume is also affected by sampled compartment ‘height’.

There is moderate negative correlation between ‘fireHeight’ and the response parameter ‘timeToMaxULT’. The CFAST model suppresses the fire heat release rate if the upper layer descends to and encompasses the fire. Since the upper layer is composed largely of products of combustion, its reduced oxygen concentration may not support the prescribed fire heat release rate. The negative correlation indicates that fires located higher within the compartment are often affected by upper layer descent, resulting in suppression of the heat release rate, at which point the upper layer has likely reached its maximum temperature for the scenario (unless the layer ascends, for example if a door were to be opened, which is not postulated in this analysis). Because the time scale of upper layer descent is shorter than that of the heat release rate profile, the time to maximum upper layer temperature tends to be shorter for cases where the upper layer envelops the fire.

There are moderate positive correlations between parameters characterizing the overhead cable tray configuration (‘hrrTray1’, ‘hrrTray2’, ‘hrrTray3’, ‘hrrTray4’, ‘hrrTray5’, and ‘hrrTray6’). These parameters represent the number of cable trays at each elevation above the

ignition source (see Figure 6). Their correlation is sensible and relates to how cable trays tend to be designed and routed in symmetric stacks.

Finally, there is moderate positive correlation between the overhead cable tray configuration and the response parameters ‘maxULT’ and ‘timeToMaxULT’. This correlation is sensible and indicates that larger numbers of overhead cable trays available for fire propagation lead to larger upper layer temperatures, and also to longer times to reach maximum temperature because the fire growth profile is extended by the fire propagation.

### 3.7.3 Centering and Scaling

The input parameter histograms in Figure 8 and Figure 9 illustrate a range of magnitudes between parameters. For example, the number of cable tray parameters take integer values between 0 and 3, the room dimensions range between 10 and 35, and the ventilation rates range between 0.00047 and 0.00189.

Many metamodels are sensitive to scale variance between predictors, and RAVEN therefore first centers and scales all predictor and response parameters to a mean zero and unit standard deviation prior to metamodel training, using the following formula:

$$x_{i,cs} = \frac{x_i - \bar{x}}{s_x}$$

Where,

$x_{i,cs}$ is the centered and scaled $i^{th}$ value of the parameter $x$

$x_i$ is the $i^{th}$ raw value of the parameter $x$

$\bar{x}$ is the parameter $x$ sample mean

$s_x$ is the parameter standard deviation

Figure 20 is an example comparison using the 'length' parameter between the raw and centered/scaled values.

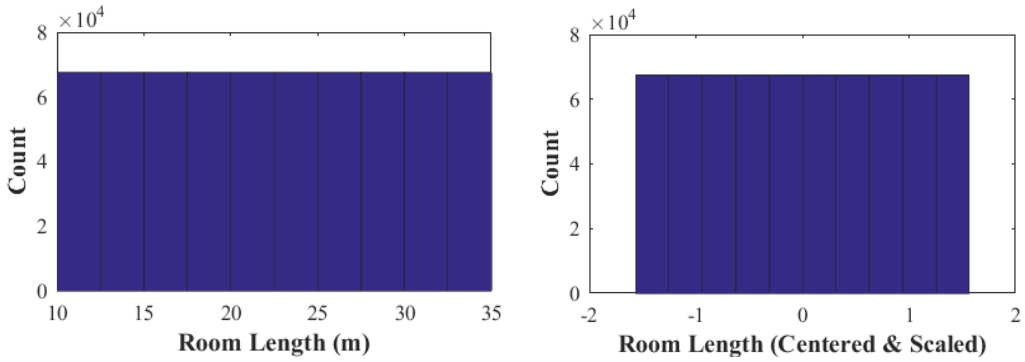


**Figure 20.** Example Comparison of Raw Parameter Values to their Centered and Scaled Values

When the trained model is used for predicting, RAVEN centers and scales the input data as required, runs the metamodel, and transforms the prediction back to the original center and scale. The metamodel prediction is therefore in the correct units (degrees Celsius and minutes) in this application.

### 3.7.4 Initial Training and Testing: Linear Models

Twenty five (25) reduced order models are tested. The initial metamodel training data consist of 50,000 random samples drawn from the 675,000 sample population. One thousand (1,000) random samples, independent from the training data, are reserved for testing the trained metamodels. It was not computationally feasible to use the entire population of data due to the computer run time required for training and testing 25 reduced order models.

The linear models tested were generally unable to predict maximum ULT, or time to maximum ULT, with reasonable accuracy. Figure 21 through Figure 25 plot for each of the two response parameters the “observed” (CFAST calculated) versus predicted values using the trained metamodels.

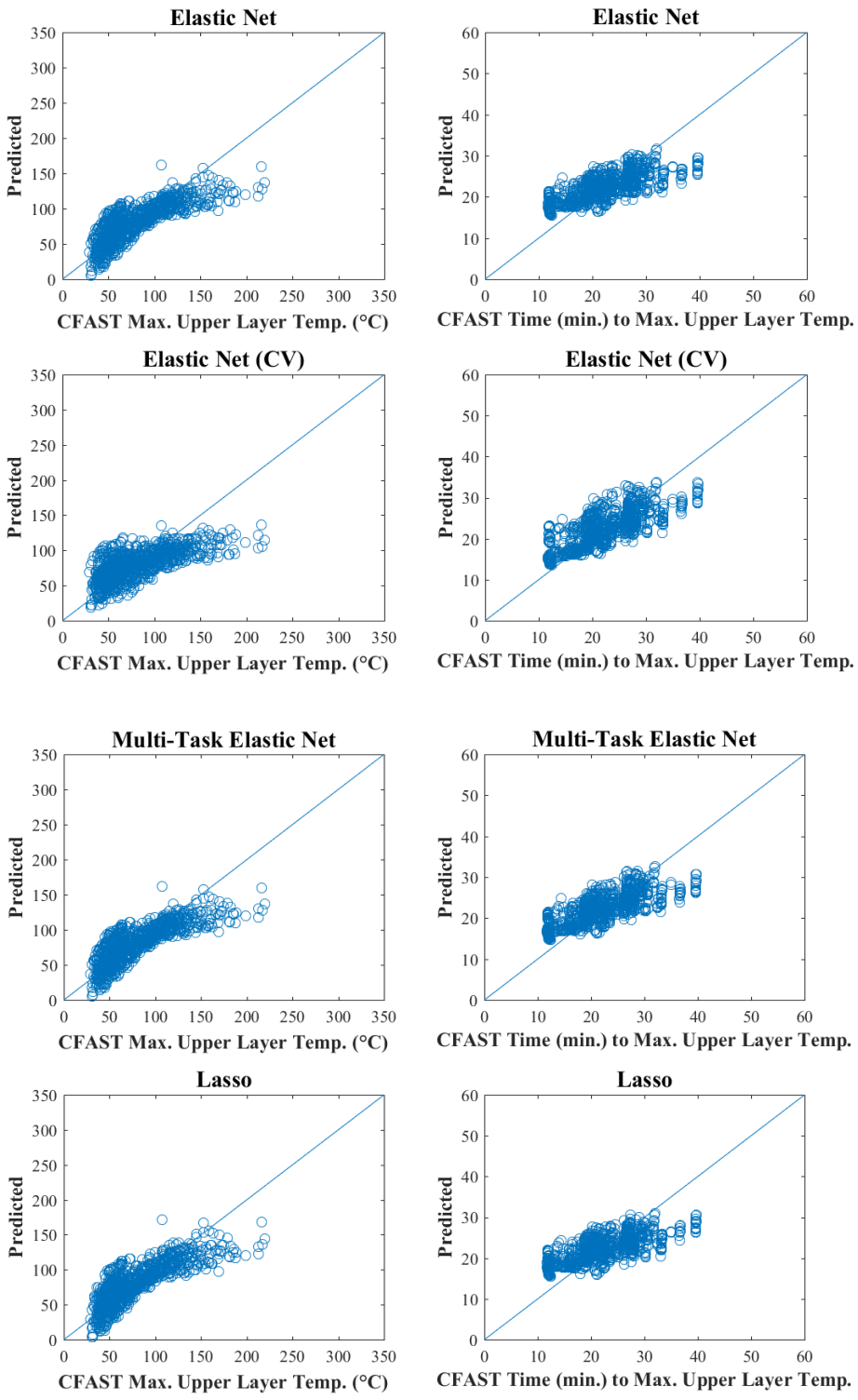


**Figure 21.** Initial Fitting of Linear Metamodels (1-4)

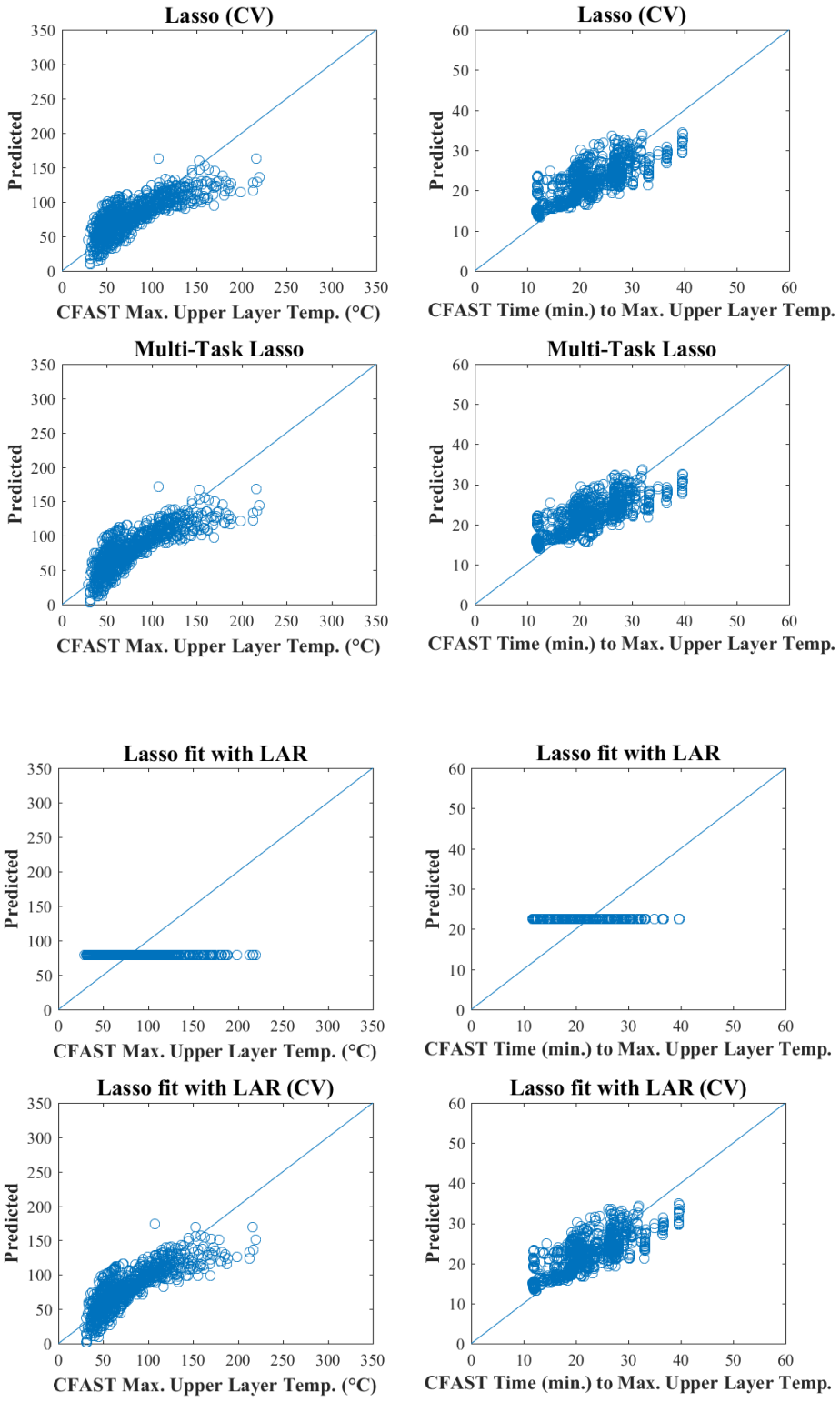


**Figure 22.** Initial Fitting of Linear Metamodels (5-8)

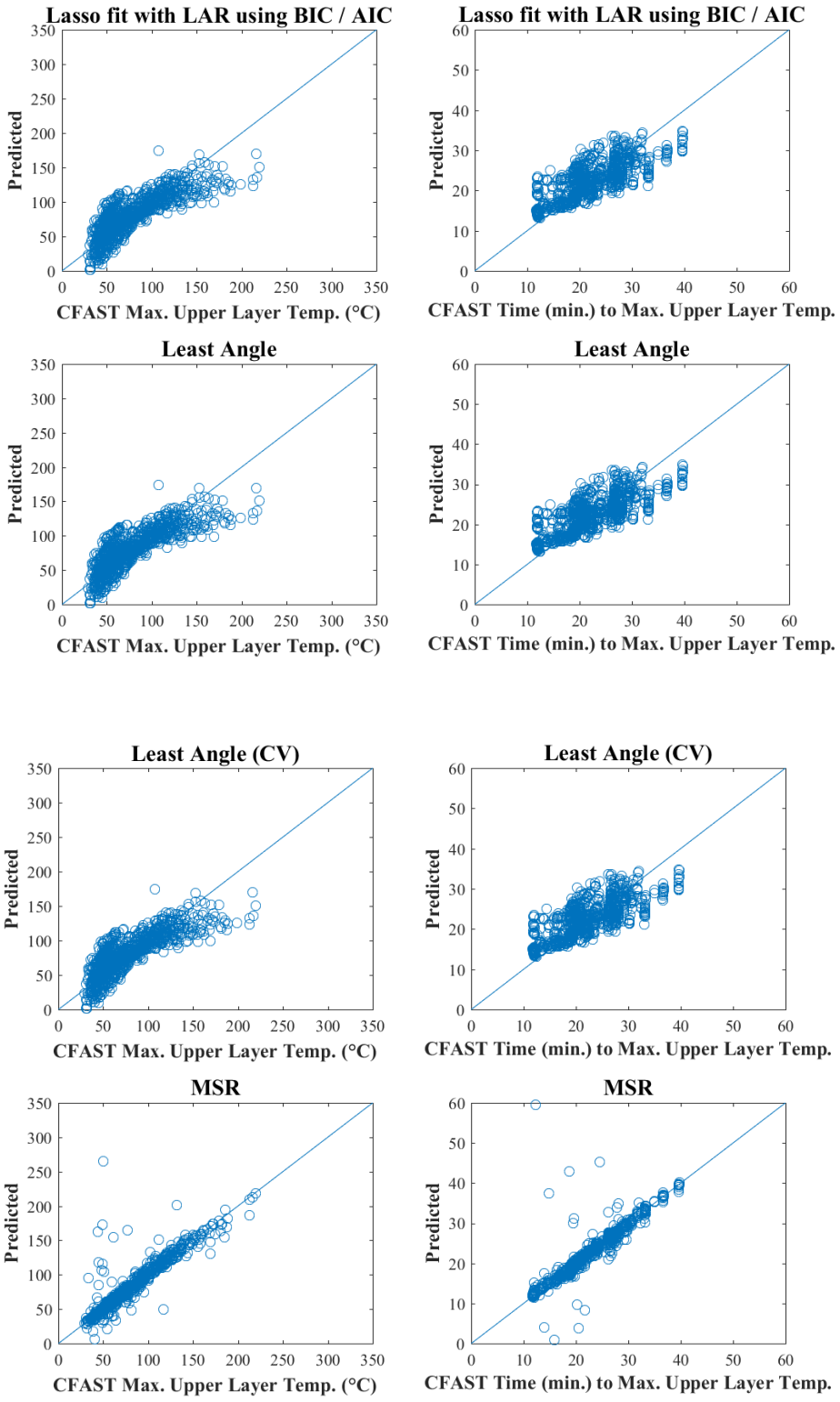


**Figure 23.** Initial Fitting of Linear Metamodels (9-12)

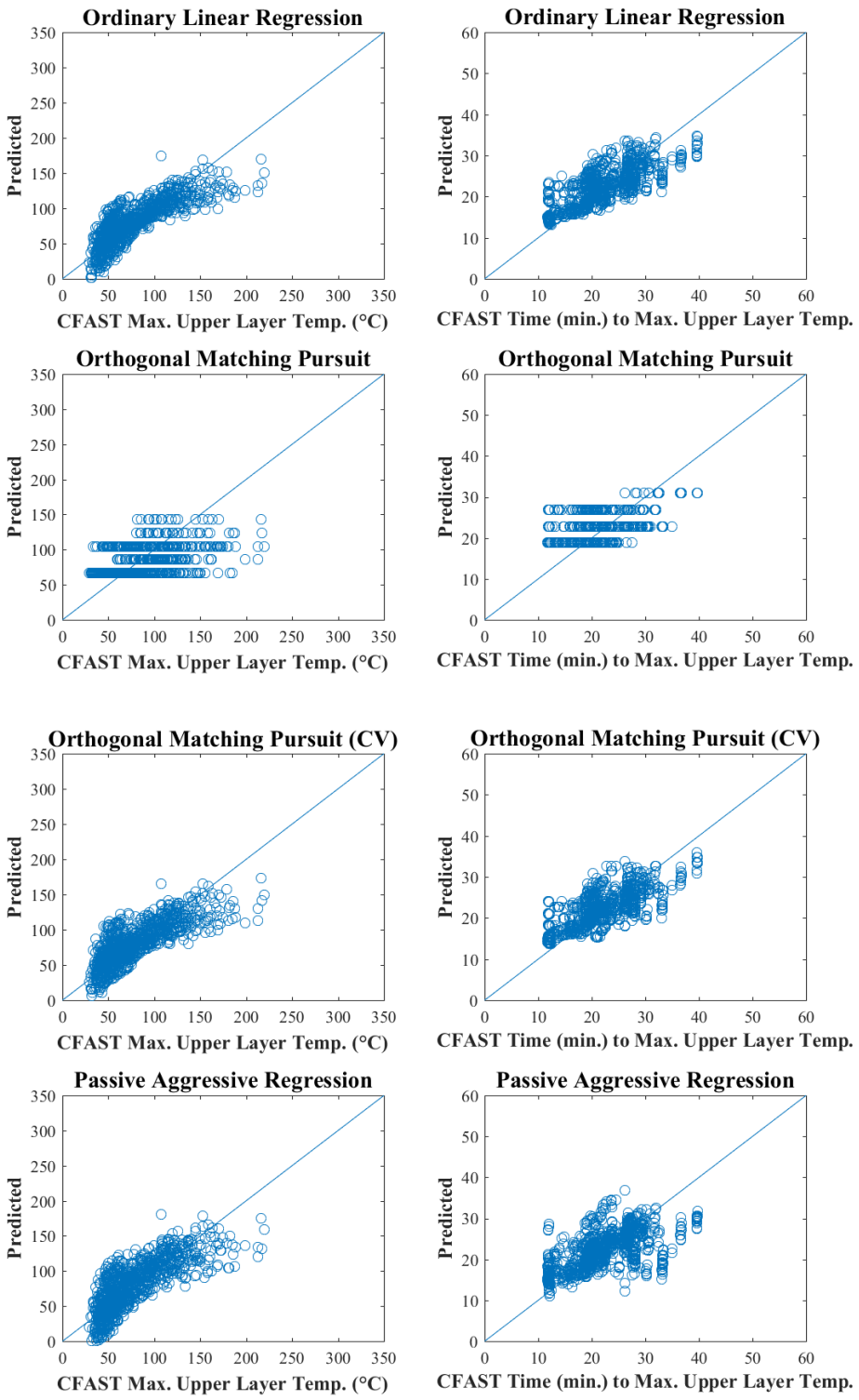


**Figure 24.** Initial Fitting of Linear Metamodels (13-16)

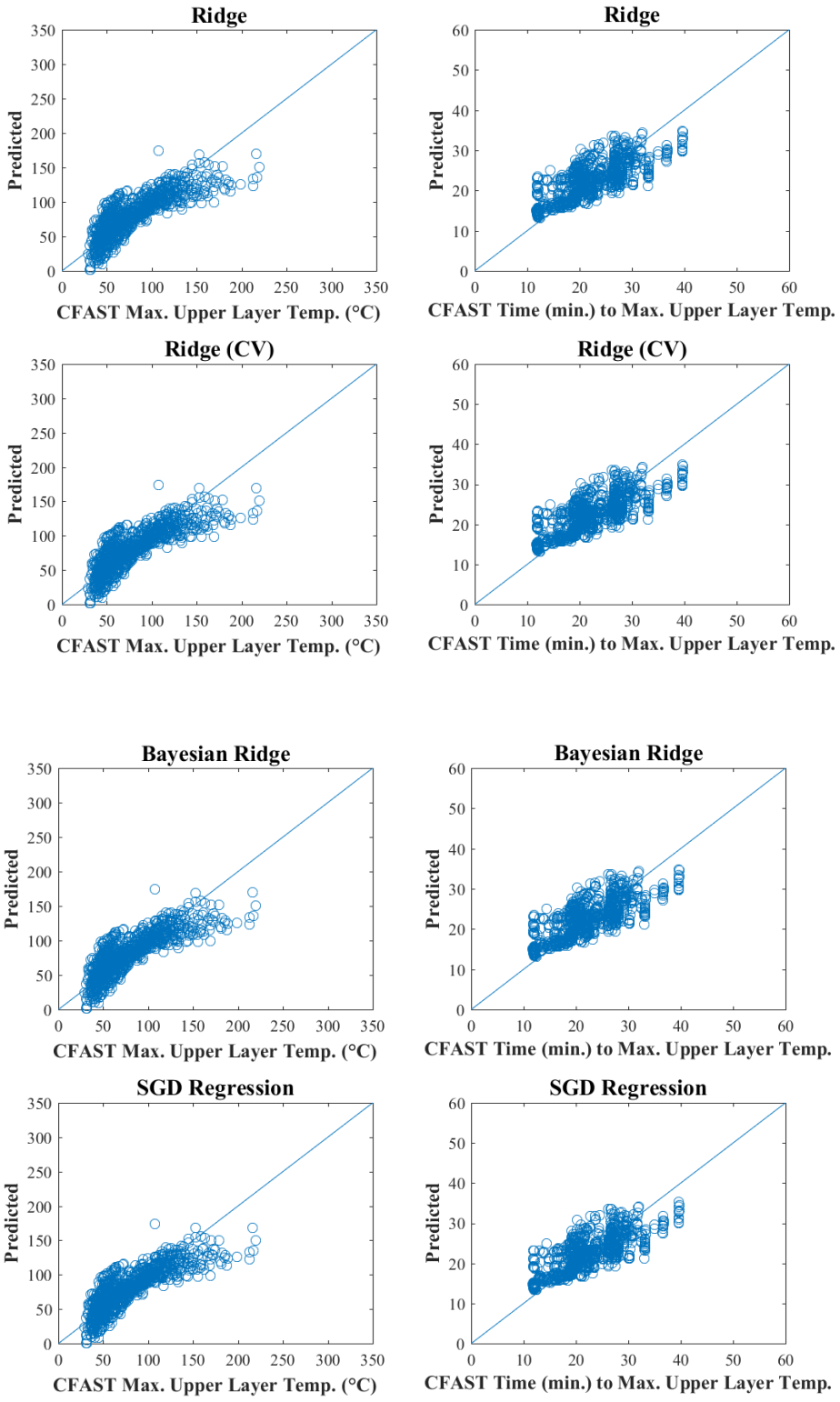


**Figure 25.** Initial Fitting of Linear Metamodels (17-20)

The poor fit of linear models is fundamentally because the physics of fire growth and compartment energy balance are non-linear.

Fire growth follows an exponential profile from ignition until reaching peak heat release rate, as illustrated in Figure 5. This exponential profile has been found to be so reliably representative that it is commonly prescribed for fire protection design applications.

In addition, the conservation of energy equations solved by CFAST are highly non-linear. To illustrate this non-linearity, the following equation is a simplified form of the conservation of energy equation for compartment fires used commonly by the fire protection industry for quick estimations (Karlssson & Quintiere, 1999):

$$\Delta T_{HGL} = 6.85 \left( \frac{\dot{Q}^2}{A_O \sqrt{H_O} h_k A_T} \right)^{1/3}$$

Where,

$\Delta T_{HGL}$ = Hot gas layer temperature rise (°C)

$\dot{Q}$ = Fire heat release rate (kW)

$A_O$ = Total ventilation opening area ($m^2$)

$H_O$ = Average height of ventilation opening (m)

$h_k$ = Convective heat transfer coefficient ($kW/m^2$-K)

$A_T$ = Total internal surface area of compartment boundaries excluding ventilation openings ($m^2$)

As previously discussed, the fire heat release rate profile itself is non-linear, and it enters into the energy balance with a 2/3 power. The temperature rise scales inversely with the total internal surface area of the compartment, which is a non-linear function of the compartment dimensions. Finally, the temperature rise also scales non-linearly with opening area and its location height. Numerical solution to the energy balance differential equations, which is what CFAST performs, is similarly non-linear.

In conclusion, the linear-based reduced order models available within RAVEN are excluded from further consideration for this application.

### 3.7.5 Initial Training and Testing: Tree-Based Models

The tree-based regression methods (decision tree and extra tree) showed good fits for both response variables during their initial testing. Extra tree differs from decision tree in that, when determining how to split the samples of a given node into two groups, extra tree selects the best of several randomly drawn possibilities. However, as shown in Figure 26, the additional complexity of the extra tree model does not significantly improve its accuracy, at least in the CFAST fire modeling context.

The conventional decision tree is selected for additional analysis, and the extra tree is eliminated from further consideration. Note that these initial tree-based models are likely highly over-fit due to not specifying a maximum depth or group size. These factors will be explored in subsequent model tuning.

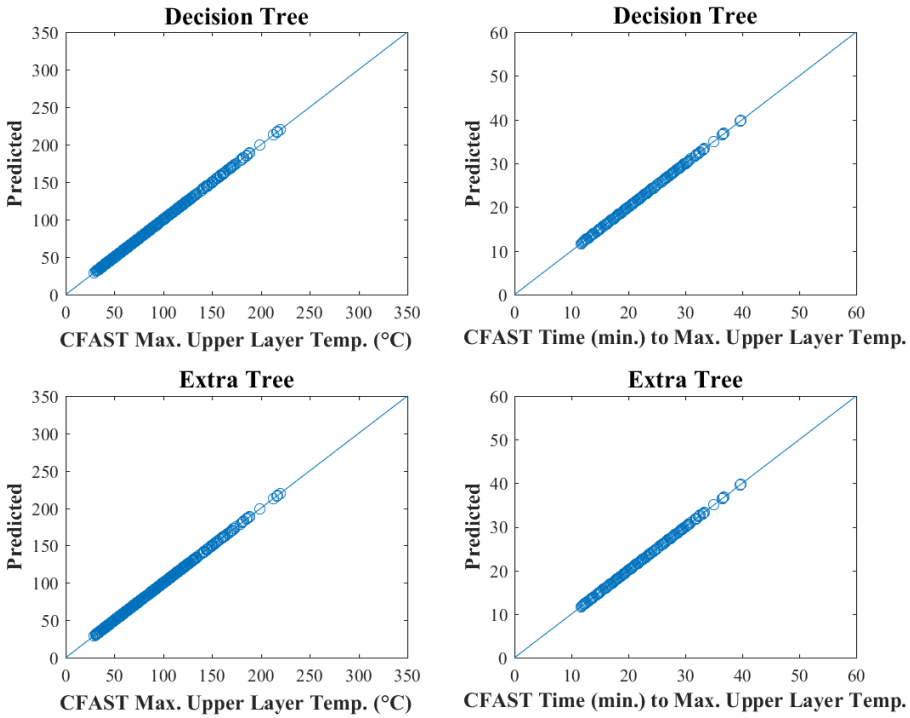


**Figure 26.** Initial Fitting of Tree-Based Metamodels

### 3.7.6 Initial Training and Testing: Neighbor-Based Models

Figure 27 shows that the neighbor-based models (*k*-nearest neighbor and radius-based neighbor) are able to resolve more of the response than the linear models, but there is still some scatter. These models appear to hold some promise, and *k*-nearest neighbor is retained for further consideration.

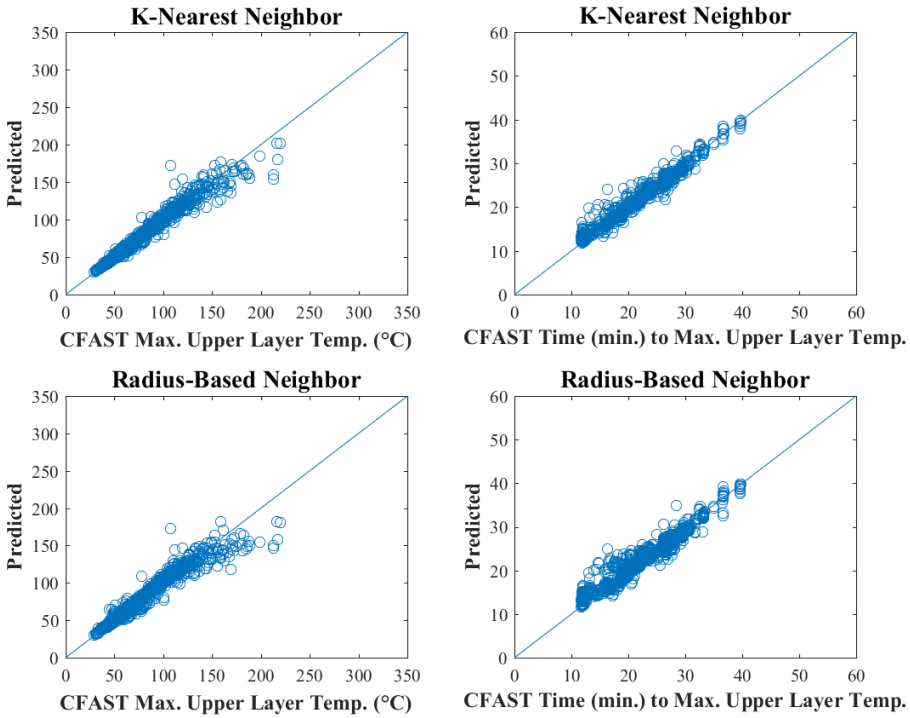


**Figure 27.** Initial Fitting of Neighbor-Based Metamodels

### 3.7.7 Initial Training and Testing: Support Vector Machine

Similar to the neighbor-based models, Figure 28 shows that support vector machine is able to resolve more of the response than the linear models. The fit for time to maximum upper layer temperature appears slightly better than the fit for maximum upper layer temperature. Support vector machine is retained for further study.

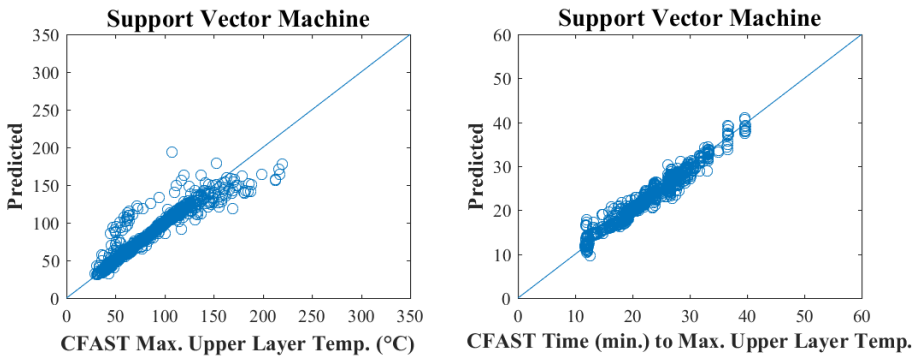


**Figure 28.** Support Vector Regression Metamodels

### 3.7.8 Initial Training and Testing: Summary

In summary, the following metamodel types are retained for further exploration:

- Decision Tree
- K-Nearest Neighbor
- Support Vector Machine

These metamodel types appear to have the most potential of those tested to efficiently mimic CFAST estimations of maximum upper layer temperature, as well as time to maximum upper layer temperature.

## 3.8 METAMODEL SELECTION AND TUNING

The reduced order model tuning and optimization features of RAVEN were under development at the time of this thesis. After using RAVEN for the data generation and initial screening of

various reduced order model types, the selected models were tuned and validated with the `R` software (R Core Team, 2016). Using an alternate platform for the final model tuning provides some measure of validation and comparison against RAVEN and its underlying Python-based algorithms.

The full dataset consists of 675,000 CFAST runs, with 80% (540,000 runs) used for final model training, and 20% (135,000 runs) reserved for testing. The training and testing runs were selected randomly.

### 3.8.1 Decision Tree Regressor

The `caret` (Kuhn, 2017) and `partykit` (Hothorn & Zeileis, 2015) packages of the `R` software were used for final tuning of the decision tree reduced order model. The `rpart.control` function enables control of several tuning parameters, of which the following are exercised for this application:

- `minsplit`: minimum number of observations at a node for the algorithm to attempt splitting.
- `minbucket`: minimum number of observations at any terminal (leaf) node.
- `maxdepth`: maximum depth of any node in the final tree. Note maximum value is 30 for the `rpart.control` function.
- `cp`: complexity parameter. The algorithm stops when the next step would not improve fit by a factor of the complexity parameter.

To first attempt a rather deep, and potentially over-fit tree, these parameters were set to `minsplit`=1, `minbucket`=1, `maxdepth`=30, and `cp`=0.001), yielding the following output:

```
Regression tree:
rpart(formula = maxULT ~ ., data = trainMaxTempData, method = "anova",
    control = treeControl)

Variables actually used in tree construction:
 [1] fireHeight floorArea  height     hrrAdj     hrrPDF     hrrTray3
hrrTray4   hrrTray5   hrrTray6   tempAmb

Root node error: 681467353/540000 = 1262

n= 540000

           CP nsplit rel error  xerror       xstd
1  0.3864657      0   1.00000 1.00000 0.00238375
2  0.1424653      1   0.61353 0.61354 0.00167808
3  0.0309442      2   0.47107 0.47108 0.00128597
4  0.0268769      3   0.44012 0.44014 0.00115187
5  0.0198330      4   0.41325 0.41326 0.00110675
6  0.0187160      6   0.37358 0.37360 0.00098649
7  0.0152718      7   0.35487 0.35489 0.00095466
8  0.0146581      8   0.33959 0.33965 0.00092796
9  0.0119471      9   0.32494 0.32499 0.00091234
10 0.0115245     10   0.31299 0.31452 0.00081545
11 0.0114741     11   0.30146 0.29951 0.00079119
12 0.0114024     12   0.28999 0.28700 0.00077498
13 0.0113709     13   0.27859 0.27357 0.00075073
14 0.0095782     16   0.24448 0.24459 0.00068694
15 0.0071252     19   0.21574 0.21586 0.00065793
16 0.0070957     20   0.20862 0.20991 0.00063890
17 0.0068717     21   0.20152 0.20164 0.00060157
18 0.0056783     24   0.18090 0.18102 0.00045518
19 0.0048261     25   0.17523 0.17527 0.00044112
20 0.0045721     26   0.17040 0.17049 0.00044211
21 0.0044519     27   0.16583 0.16638 0.00043956
22 0.0041076     28   0.16138 0.16193 0.00043100
23 0.0034041     29   0.15727 0.15748 0.00042400
24 0.0028606     30   0.15386 0.15408 0.00041947
25 0.0023971     31   0.15100 0.15119 0.00041689
26 0.0023934     32   0.14861 0.14917 0.00041328
27 0.0023859     33   0.14621 0.14775 0.00041130
28 0.0021540     34   0.14383 0.14454 0.00040670
29 0.0019941     35   0.14167 0.14104 0.00040168
30 0.0019153     36   0.13968 0.13984 0.00040064
31 0.0016980     37   0.13776 0.13793 0.00038725
32 0.0016593     38   0.13607 0.13595 0.00038627
33 0.0016252     39   0.13441 0.13479 0.00038213
34 0.0015952     40   0.13278 0.13361 0.00037982
35 0.0014524     41   0.13119 0.13137 0.00037718
36 0.0014121     42   0.12973 0.12966 0.00037512
37 0.0012624     43   0.12832 0.12793 0.00037045
38 0.0012520     44   0.12706 0.12674 0.00036885
39 0.0012384     45   0.12581 0.12579 0.00036679
40 0.0012121     46   0.12457 0.12489 0.00036519
41 0.0011227     47   0.12336 0.12339 0.00036339
42 0.0010858     48   0.12223 0.12217 0.00036148
43 0.0010387     49   0.12115 0.12118 0.00036135
44 0.0010000     50   0.12011 0.12020 0.00036081
```

Fourty four (44) trees were generated, with the most complex tree having 50 splits and a relative error of 12%. This error is relative to the least complex tree with only one split. Figure 29 depicts the coefficient of determiation ($R^2$) versus number of tree splits, and the accuracy appears to start leveling off near 35 splits.

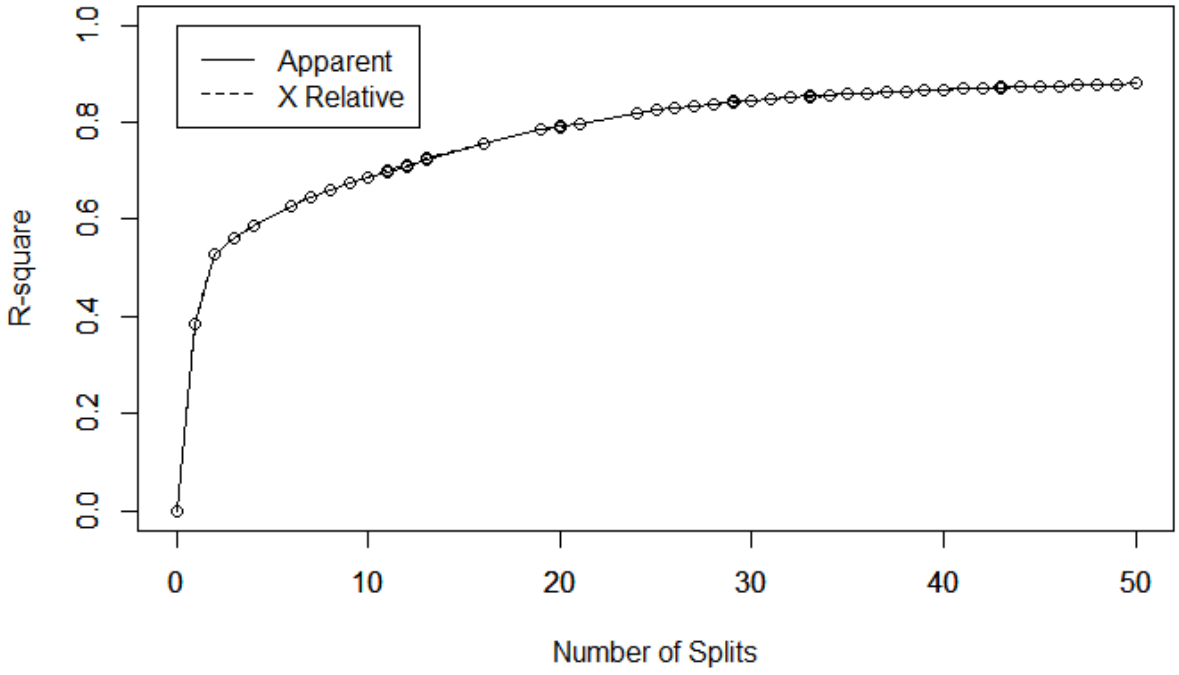


**Figure 29.** Regression Tree Coefficient of Determination ($R^2$) as a Function of Number of Splits

Figure 30 illustrates the regression tree for maximum upper layer temperature with 35 splits. Note that several parameters have been dropped from this pruned tree, most notably the ventilation rate and room height do not appear important to the final model. Conversely, the model retains floor area and factors affecting fire propagation (number of adjacent cabinets and

overhead cable trays) as these parameters are most important to the maximum achieved upper layer temperature.

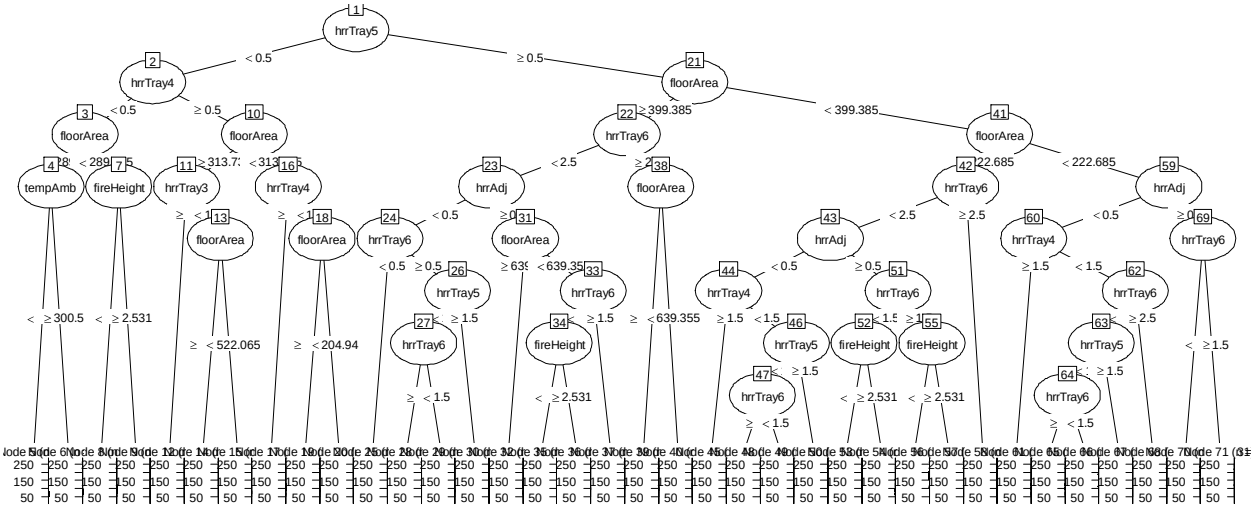


**Figure 30.** Regression Tree for Maximum Upper Layer Temperature

This same process was applied to the response variable time to maximum upper layer temperature, and Figure 31 illustrates the resulting regression tree predicted versus CFAST estimated response parameters.

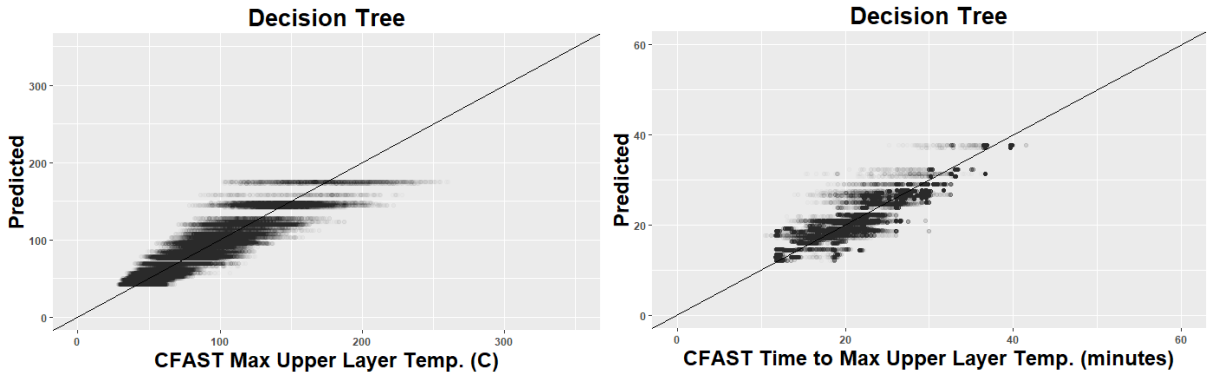


**Figure 31.** Regression Tree Predicted versus CFAST Estimated

The `rpart` pruned decision tree for maximum upper temperature appears well-centered around the CFAST calculated results, however there is considerable variance. The decision tree for time to maximum upper layer temperature is similarly centered around the CFAST calculated results, however there is considerable variance, and that variance does not appear symmetric or consistent.

This is in contrast to the RAVEN/Python-developed decision tree shown in Figure 26, which appears highly accurate. Upon further investigation, while RAVEN allows specifying some control parameters such as maximum tree depth, these parameters were not exercised in the initial screening. In addition, the RAVEN implementation of decision tree regression does not include a cross-validation option. This resulted in the initial model being very deep and over-fit. Tuning the model with `rpart` to a reasonable depth and complexity, and using cross-validation, resulted in the decision tree modestly fitting the CFAST parameters of interest.

### 3.8.2 *k*-Nearest Neighbor (*k*NN) Regression

The `caret` package (Kuhn, 2017) of the `R` software (R Core Team, 2016) was used for final tuning of the *k*NN regression model. First, a tuning grid over a range of *k*-neighbors, using 10-fold cross-validation, was applied to a 10,000 sample random subset of the 540,000 samples reserved for training. Computational burden prevented running this tuning grid over more than 10,000 samples. The data were centered and scaled prior to training, as is required by neighbor-based approaches. This initial fitting yielded the following output:

```
k-Nearest Neighbors

10000 samples
   13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9000, 9000, 9000, 9000, 9000, 9000, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   1  13.37537  0.8593849  7.963614
   2  12.07356  0.8824451  7.180628
   3  11.86646  0.8866827  7.046577
   4  11.81016  0.8884675  7.021705
   5  11.86146  0.8881417  7.096210
   6  11.91607  0.8876787  7.144894
   7  11.91869  0.8879799  7.179175
   8  11.95182  0.8877548  7.211671
   9  11.95644  0.8882051  7.221465
  10  12.01002  0.8875607  7.273300
  11  12.04641  0.8871765  7.321889
  12  12.08740  0.8867153  7.356246
  13  12.15766  0.8856441  7.375611
  14  12.24918  0.8840457  7.430219
  15  12.32137  0.8828615  7.482220
  16  12.36077  0.8824768  7.506818
  17  12.42883  0.8814411  7.550267
  18  12.50406  0.8801791  7.588733
  19  12.60102  0.8783818  7.653825
  20  12.68058  0.8770752  7.720678
  21  12.77538  0.8752380  7.784058
  22  12.85360  0.8739216  7.828273
  23  12.94020  0.8722173  7.886573
  24  13.02942  0.8705440  7.957689
  25  13.11021  0.8690089  8.017898
  26  13.18634  0.8675714  8.077936
  27  13.25899  0.8660713  8.140844
  28  13.33462  0.8646183  8.200422
  29  13.42653  0.8627696  8.264695
  30  13.50092  0.8613037  8.318028

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 4.
```

The output confirms that 10,000 samples of 13 centered and scaled predictor variables were used to fit 30 models, ranging from $k \in [1,30]$. 10-fold cross validation was used with 9,000 training sample batches. Figure 32 plots the *k*NN error (RMSE) as a function of the number of

neighbors used by the algorithm. According to this plot and the R output, four (4) neighborhoods appears optimal.

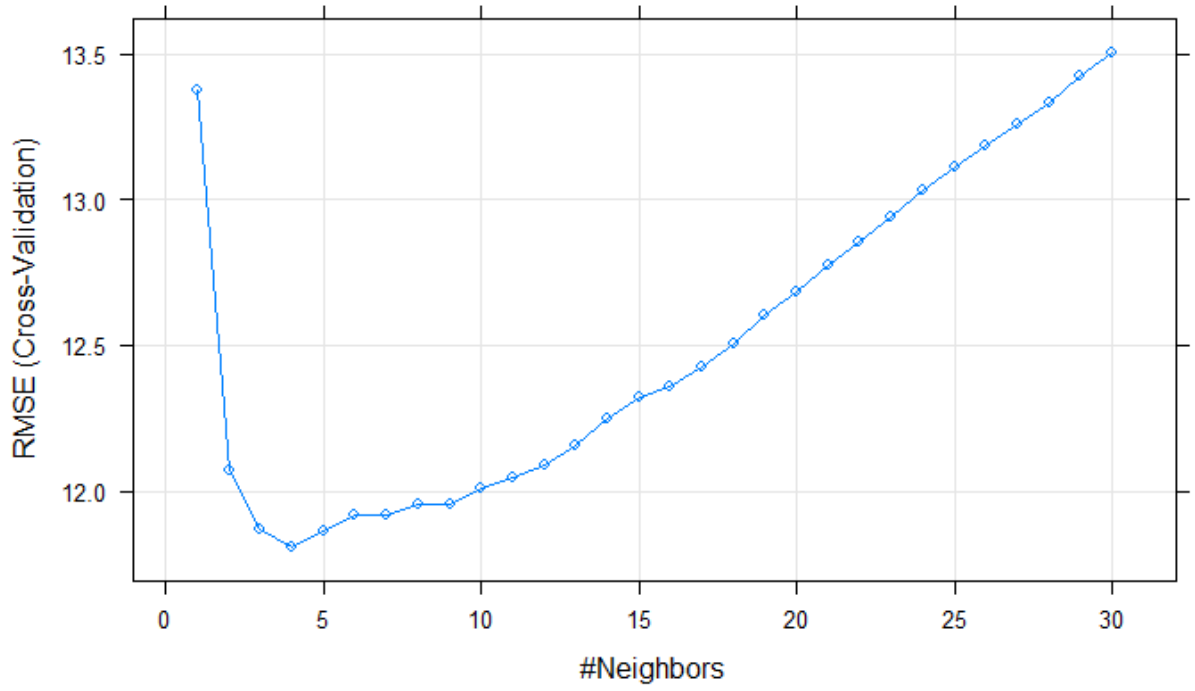


**Figure 32.** *k*NN Root Mean Squared Error as a Function of Number of Neighbors

The convex shape of the Figure 32 error plot is caused by the use of cross-validation to prevent over-fitting. When more neighbors are used, the accuracy improves relative to the training set, but the generalized accuracy decreases when tested against the reserved datasets.

Next, a *k*NN model using four (4) neighbors was trained using 100,000 samples. This same process was applied to the response variable time to maximum upper layer temperature, which resulted in a model optimized at five (5) neighbors. Figure 33 illustrates the model predicted versus CFAST estimated response parameters.

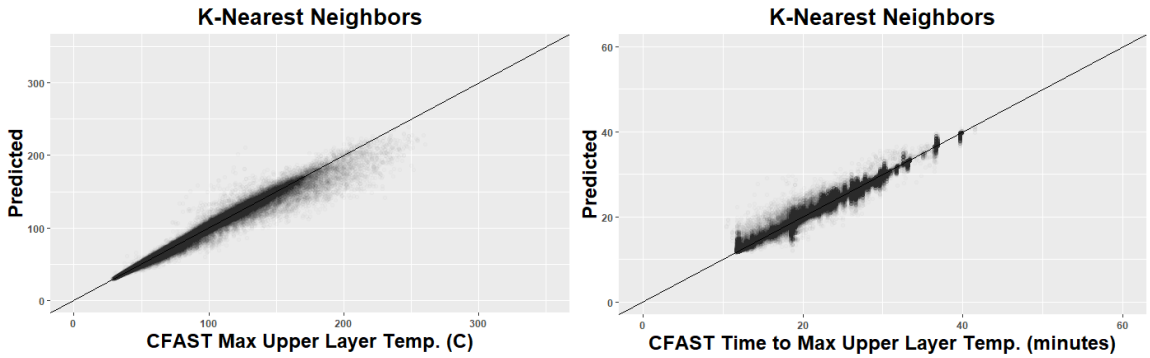


**Figure 33.** K-Nearest Neighbor Predicted versus CFAST Estimated

The scatter plots for both maximum upper layer temperature, and time to maximum upper layer temperature, show good fits by *k*-nearest neighbors. The models show good generalization, and they are not over-fit, due to the use of 10-fold cross validation, and testing on an independent dataset that was not used for training.

Note that data density can be overstated in Figure 33 because it attempts to display all 135,000 test data points. Figure 34 depicts the ratio of predicted to observed for all 135,000 data points. Values greater than 1.0 indicate over-prediction, and values less than 1.0 indicate under-prediction. The histogram shows the vast majority of *k*NN model predictions are within ±10% of the CFAST calculated values.

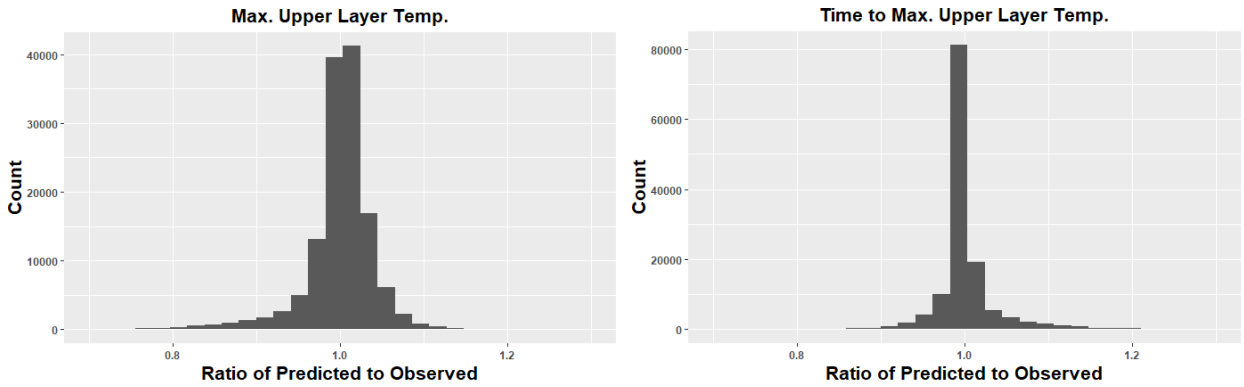


**Figure 34.** Ratio of *k*NN Predicted to CFAST Calculated

### 3.8.3 Support Vector Machine

The `caret` package (Kuhn, 2017) of the `R` software (R Core Team, 2016) was used for final tuning of the support vector machine regression model. First, a tuning range of 14 complexity parameter values, using 10-fold cross-validation, was applied to a 5,000 sample random subset of the 540,000 samples reserved for training. Computational burden prevented running this tuning grid over more than 5,000 samples. The data were centered and scaled prior to training, and this initial fitting yielded the following output:

```
Support Vector Machines with Radial Basis Function Kernel

5000 samples
  13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4499, 4500, 4500, 4500, 4500, 4500, ...
Resampling results across tuning parameters:

  C        RMSE      Rsquared   MAE
     0.25  24.59553  0.7038072  15.51496
     0.50  20.24993  0.7774093  12.50497
     1.00  17.40660  0.8189565  11.06145
     2.00  16.30502  0.8293096  10.68497
     4.00  16.00320  0.8324349  10.63159
     8.00  15.97615  0.8331374  10.62822
    16.00  15.97184  0.8333873  10.62777
    32.00  15.97184  0.8333873  10.62777
    64.00  15.97184  0.8333873  10.62777
   128.00  15.97184  0.8333873  10.62777
   256.00  15.97184  0.8333873  10.62777
   512.00  15.97184  0.8333873  10.62777
  1024.00  15.97184  0.8333873  10.62777
  2048.00  15.97184  0.8333873  10.62777

Tuning parameter 'sigma' was held constant at a value of 0.05070748
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.05070748 and C = 16.
```

The output confirms that 5,000 samples of 13 centered and scaled predictor variables were used to fit 14 models, ranging in complexity parameter. 10-fold cross validation was used with ~4,500 training sample batches. Figure 35 plots the support vector machine error (RMSE) as a function of complexity parameter. According to this plot and the R output, an SVM model with a complexity parameter of 16 appears optimal. A second model for the time to reach maximum upper layer temperature response variable was similarly developed.

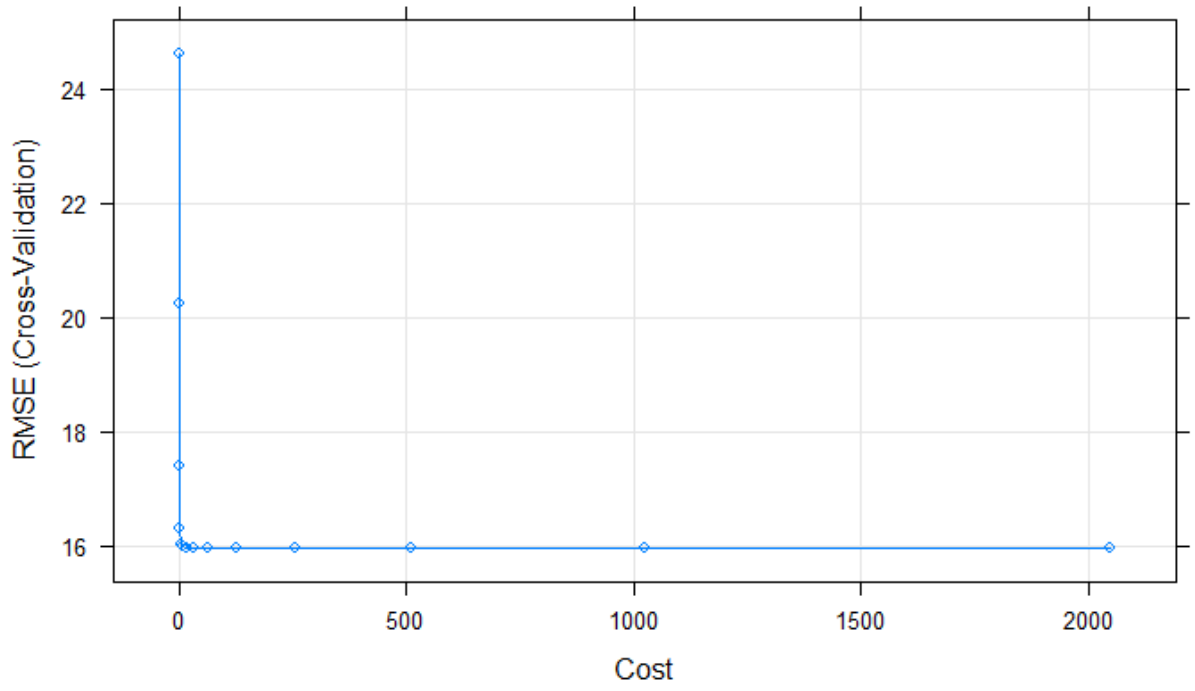


**Figure 35.** Support Vector Machine Root Mean Squared Error as a Function of Complexity Parameter

Note that the above model was developed using a radial basis kernel, and models were similarly developed using a linear kernel. A polynomial kernel was attempted, but it was extremely slow to train and therefore discarded.

Next, the selected support vector machine models for each response variable were used to predict the CFAST output for the reserved 135,000 testing samples. Figure 36 illustrates the model predicted versus CFAST estimated response parameters.

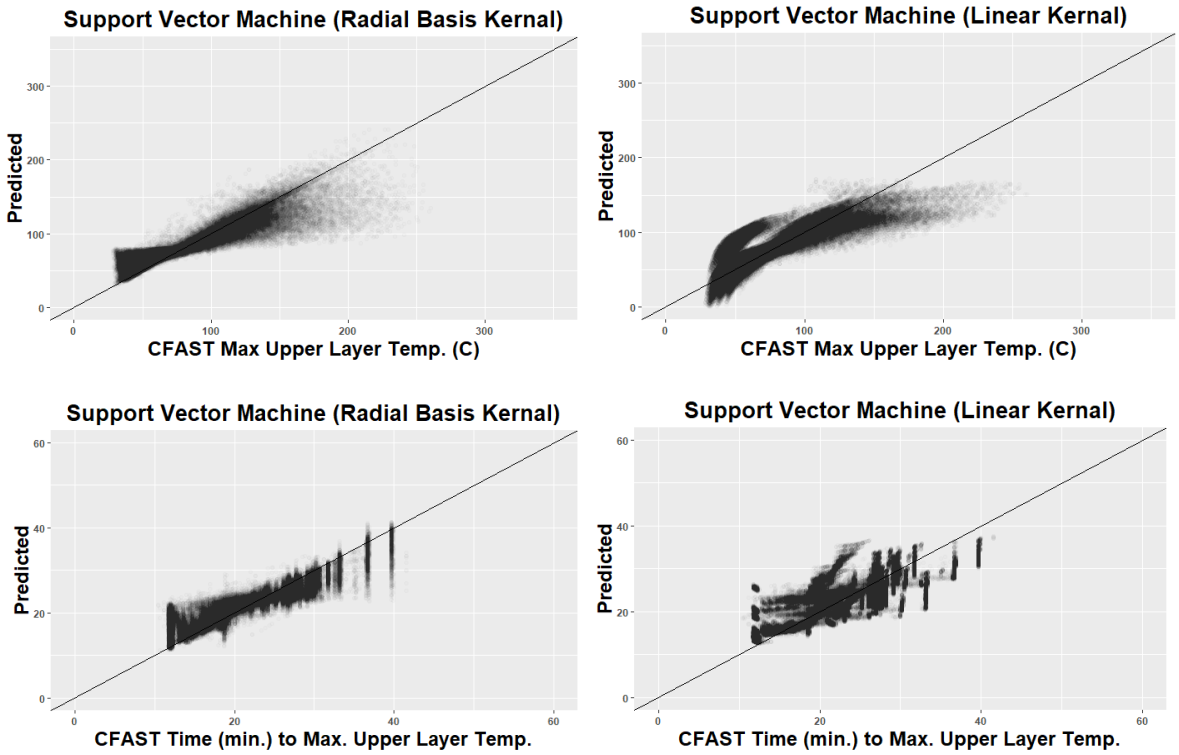


**Figure 36**. Support Vector Machine Predicted versus CFAST Estimated

Support vector regression did not perform well in this application, and this could in part be due to its complexity. The model was so slow to train that only 5,000 samples (<1% of the data population) could be used for training, which may have limited its prediction accuracy. Slow training also erodes the benefit of a reduced order model, especially for CFAST, which is a relatively efficient code.

## 3.9 COMPARISON TO ALGEBRAIC FIRE MODELS

Typical nuclear power plant fire probabilistic risk assessments postulate on the order of 1,000 (often several thousand) fire scenarios. The fire PRA is performed iteratively, with increasing

levels of modeling realism commensurate with risk significance. At the end of the process, the PRA includes conservative modeling of low risk fire scenarios, very detailed modeling (and understanding) of the most risk-significant scenarios, and a sliding scale of modeling detail for scenarios of intermediate risk significance. (Worrell & Rochon, 2016) provide an overview of the fire probabilistic risk assessment process.

There are three general classes of fire modeling tools: algebraic models, two-zone models like CFAST, and computational fluid dynamics models. The algebraic models are straightforward to implement but generally yield conservative results. Two-zone models like CFAST provide more realistic results but come at some expense, primarily in terms of model setup time. Computational fluid dynamics models are the most accurate but involve great computational expense.

Algebraic models are usually implemented for most of the fire probabilistic risk assessment scenarios because they are fast and can be implemented by spreadsheet. CFAST is generally only applied to a handful of scenarios, primarily due to the time it takes to setup, run, and evaluate a scenario through the CFAST graphical user interface. CFAST can be automated, but this requires setting up some infrastructure to manage the input data, batch run CFAST, and extract the results.

There are two algebraic fire models typically used for fire probabilistic risk assessments. The first model is referred to as MQH, which is applicable to naturally ventilated compartments, for example a room with an open door or window. The second model is referred to as FPA, which is applicable to mechanically ventilated compartments.

The MQH model of naturally ventilated compartments was developed by McCaffrey, Quintiere, and Harkleroad and is specified by the following equation (Karlssson & Quintiere, 1999):

$$\Delta T_{HGL} = 6.85\left(\frac{\dot{Q}^2}{A_O\sqrt{H_O}h_k A_T}\right)^{1/3}$$

Where,

$\Delta T_{HGL}$ = Hot gas layer temperature rise (°C)

$\dot{Q}$ = Fire heat release rate (kW)

$A_O$ = Total ventilation opening area ($m^2$)

$H_O$ = Average height of ventilation opening (m)

$h_k$ = Convective heat transfer coefficient ($kW/m^2$-K)

$A_T$ = Total internal surface area of compartment boundaries excluding ventilation openings ($m^2$)

The FPA model of mechanically ventilated compartments was developed by Foote, Pagni, and Alvares and is specified by the following equation (Iqbal et al., 2004):

$$\frac{\Delta T_{HGL}}{T_\infty} = 0.63\left(\frac{\dot{Q}}{\dot{m}_g c_p T_\infty}\right)^{0.72}\left(\frac{h_k A_T}{\dot{m}_g c_p}\right)^{-0.36}$$

Where,

$\Delta T_{HGL}$ = Hot gas layer temperature rise (K)

$T_\infty$ = Ambient temperature (K)

$\dot{Q}$ = Fire heat release rate (kW)

$\dot{m}_g$ = Ventilation rate (kg/s)

$c_p$ = Specific heat of air (kJ/kg-K)

$h_k$ = Convective heat transfer coefficient (kW/m$^2$-K)

$A_T$ = Total internal surface area of compartment boundaries excluding ventilation openings (m$^2$)

Figure 37 compares the maximum upper layer temperature estimation of the MQH and FPA algebraic models, and the *k*NN metamodel developed in this thesis, to the CFAST calculation for 135,000 randomly selected data points.

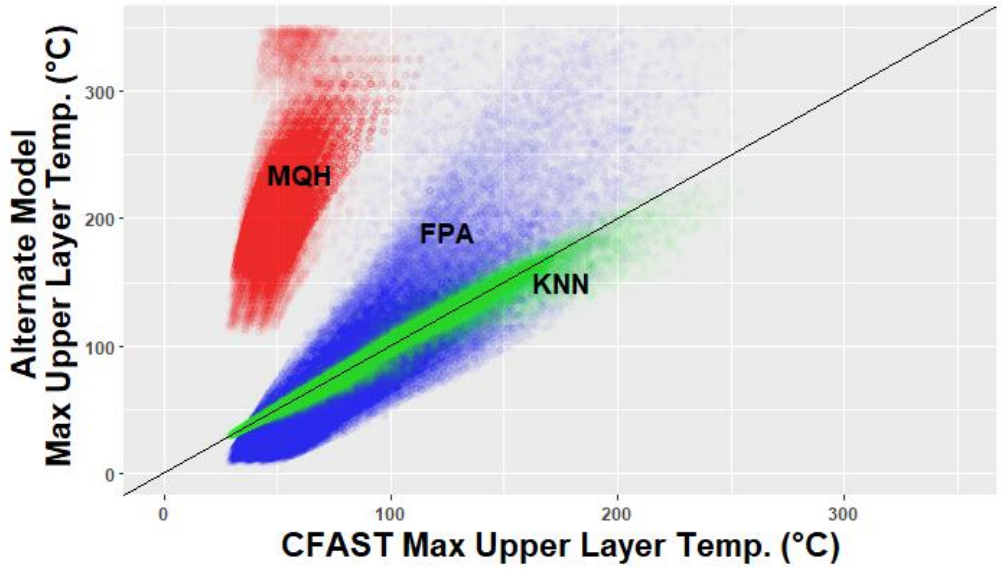


**Figure 37**. Comparison of CFAST to *k*NN and Algebraic Models

MQH is clearly very conservative relative to CFAST for this application. This is because MQH was designed for naturally ventilated compartments, and the range of tested scenario configurations are not naturally ventilated; they are mechanically ventilated. Assuming that mechanical ventilation trips or is manually shut down early in the fire, the only natural ventilation leakage path specified is a total of 0.2 $m^2$ through gaps underneath the doorways. Examining the MQH equation, the ventilation opening area and height are both in the denominator, and so very small values of either lead to high upper layer temperature estimations. As the compartment becomes more naturally ventilated, for example if the fire brigade opens the door(s), the MQH upper layer temperature predictions would move toward the CFAST estimation.

The FPA model of mechanically ventilated compartments is more physically applicable to the switchgear room fires under consideration (i.e., FPA is applicable to mechanically ventilated rooms like the postulated switchgear room scenarios, and MQH is more relevant to

naturally ventilated rooms with open doors or windows). The FPA results are more centered around the CFAST calculation; however, the variance is quite large. A significant portion of the FPA results are conservative relative to CFAST, especially for the more severe fire conditions. Furthermore, an appreciable portion of the FPA estimations, over most of the evaluated range, is actually non-conservative relative to CFAST. Because of this apparent non-conservatism (nearly 50% in some cases) of FPA relative to CFAST in this example, it is possible a safety factor would need to be applied if the model is to be used for certain design applications. FPA might however still be a good choice for probabilistic risk assessment where it is desirable the mean value be best estimate, and where model bias and uncertainty can be explicitly considered in the probabilistic simulation.

As previously discussed, the *k*-nearest neighbors model performed well, with the vast majority of *k*NN model predictions within ±10% of the CFAST calculated values.

## 3.10 ACCURACY-EFFICIENCY TRADEOFF

Model selection in many applications involves a tradeoff between desired accuracy and computational efficiency. In fire modeling, algebraic fire models take seconds to run, two-zone models like CFAST take minutes, and CFD models take hours (often days) to run. While fast, algebraic models tend to be conservative, and while very slow, CFD models can be more realistic. Modeling realism of two-zone models falls in between the algebraic and CFD models.

Applications like uncertainty quantification and sensitivity analysis may require models to be run hundreds, thousands, or even tens of thousands of times. Using CFD, and even CFAST,

for such studies is usually not feasible at present, even with high performance computing systems.

While metamodels are simplifications of higher fidelity codes, and they therefore introduce some error, metamodels provide opportunity to improve modeling realism over what would be computationally feasible otherwise. For example, in the fire context, users are generally limited to algebraic models for uncertainty quantification; however, using a CFAST metamodel, while not as accurate as CFAST itself, can be more accurate than the algebraic models.

Metamodels also have computational expense, primarily in their training. The *k*-nearest neighbors model developed here required 100,000 CFAST runs. Figure 38 and Figure 39 illustrate how metamodel accuracy improves as a function of training sample size.

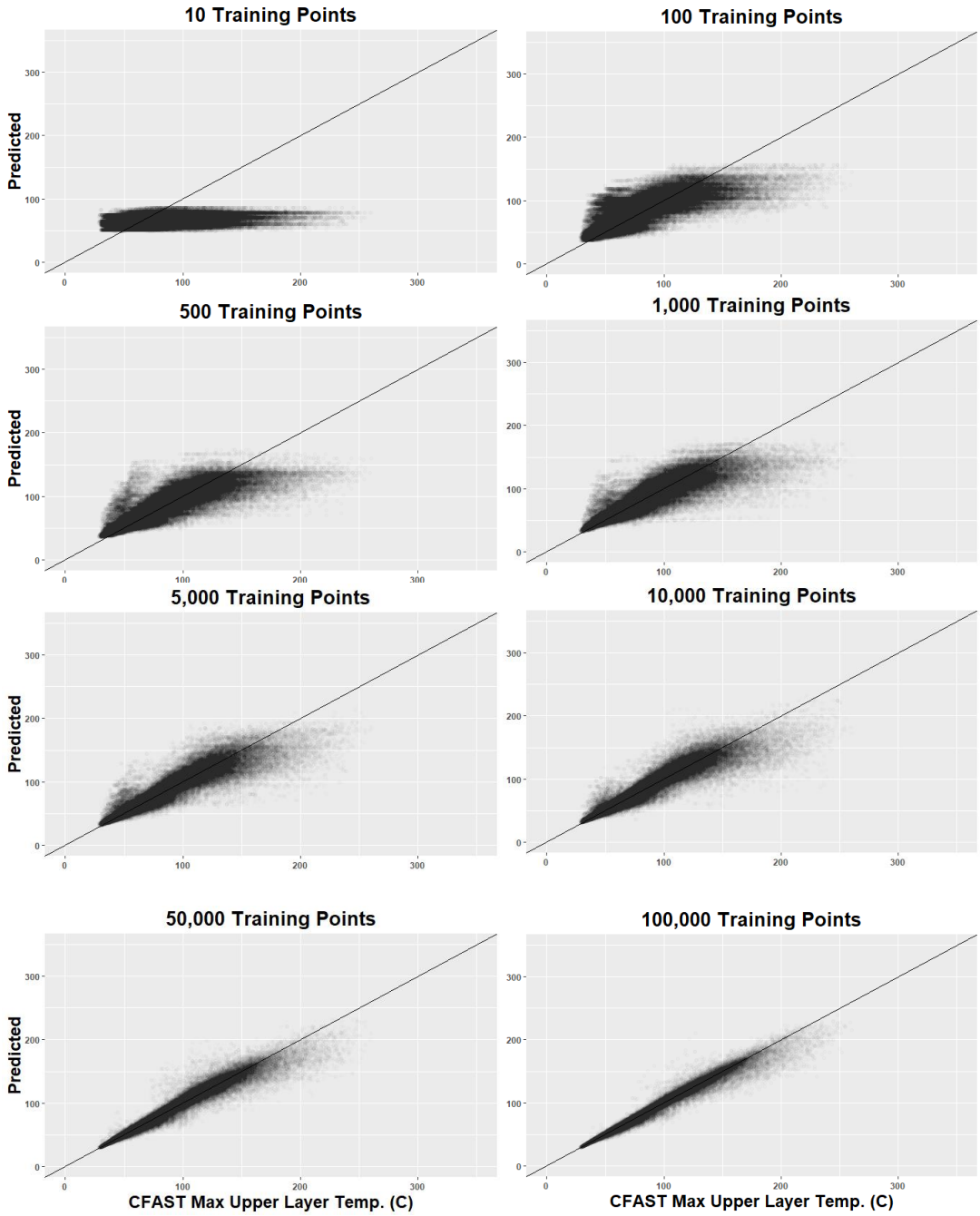


**Figure 38**. *k*NN Predicted vs. CFAST Calculated over Range of Training Sample Sizes

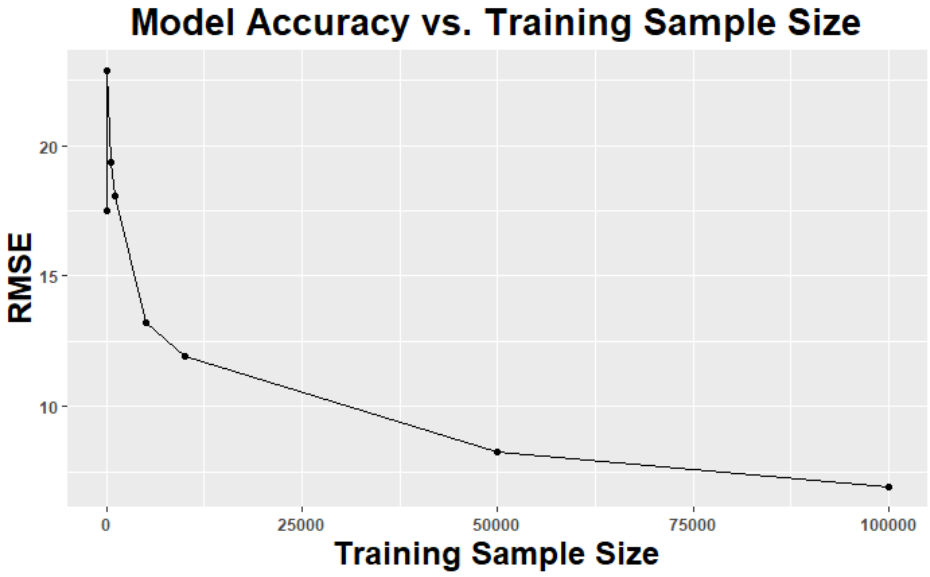


**Figure 39**. *k*NN Root Mean Squared Error vs. Training Sample Size

Metamodel accuracy generally improves with training sample size. The desired accuracy and required training sample size (and computational expense) is specific to the intended model application. Note in Figure 39 that the RMSE increases between the first (10 samples) and second (100 samples) points. Ten (10) samples is clearly insufficient for CFAST metamodel training, and it is likely the RMSE for models trained on so few points will vary significantly, depending on which samples are randomly selected for the training.

Despite introducing some error, it is important to note that error can be quantified and therefore explicitly treated in the model application. Metamodel error quantification is a standard output of the training process.

Acknowledging that metamodel development requires computational expense to generate training data, Table 14 summarizes a comparison of model speed and accuracy across several modeling options. This comparison was performed assuming that 500 fire scenarios are to be evaluated, and the peak heat release rate distribution for each scenario is discretized into 25

intervals, for a total of 12,500 runs required. The computation time estimates are based on experience during this thesis using a Hewlett Packard Z640 engineering workstation with two 2.20 GHz processors, 64 GB memory, and a 64-bit Windows 7 operating system.

**Table 14.** Comparison of Computer Run Time and Accuracy across Several Modeling Options

| **Model** | **Computer Run Time** | **Accuracy Relative to CFAST (for Maximum Upper Layer Temperature)** |
|---|---|---|
| MQH (algebraic) | Negligible | Potentially very conservative if applied to scenarios where natural ventilation to/from the compartment is limited, which is the case for many postulated nuclear power plant fire scenarios. |
| FPA (algebraic) | Negligible | RMSE = 34.0°C<br><br>Majority of FPA predictions were within ±50°C of the corresponding CFAST predictions in this study. Table 3 indicates a model bias factor of 1.44 for the algebraic fire models. |
| *k*NN (using CFAST training data generated over 1 day of computer run time) | 1 day (using 2 processors) | RMSE = 7.4°C |
| *k*NN (using CFAST training data generated over 2 days of computer run time) | 2 days (using 2 processors) | RMSE = 5.3°C |
| *k*NN (using CFAST training data generated over 15 days of computer run time) | 15 days (using 2 processors) | The *k*NN model in this study could not be trained on greater than 100,000 samples due to computational limitations. Generating 15 days of training data would therefore not improve *k*NN model accuracy. |
| CFAST | ~1.5 days (using 2 processors) | N/A. The model accuracies in this comparison are relative to what CFAST would calculate. |

The comparison in Table 14 illustrates that developing and running an accurate *k*NN metamodel would require a similar computing time as running CFAST for an application requiring 12,500 model runs. If running these cases were a one-time operation, it would be reasonable to run CFAST rather than a metamodel. However, if the analysis needed to be repeated, for example to reflect updated heat release rate distributions or adjustments to the input parameters, then use of the accurate *k*NN model could save significant computer run time (note that once the *k*NN model is trained, it can be executed over thousands of scenarios in a few seconds). Finally, a significant benefit of the *k*NN model is its portability; once trained, it can easily be applied by many users over a range of conditions.

## 3.11 LIMITATIONS

Metamodels can only be, at their best, as good as the model(s) they mimic. As summarized in Section 3.1, CFAST is a reasonably accurate predictor of hot gas layer temperature (bias factor, $\delta$, of 1.06 per Table 3) for the experimental fire tests considered by the NUREG-1824 (Hill et al., 2007) verification and validation efforts. The *k*NN metamodel developed in this thesis introduces some additional error, although small, because the metamodel is an approximation of what CFAST calculates.

The CFAST validated ranges of applicability developed by NUREG-1934 (McGrattan et al., 2012) and summarized in Table 8 are also applicable to any metamodel approximation of CFAST. When developing metamodel training data through a sampling process, it is possible that some sampled configurations could exceed the CFAST validated ranges of applicability.

This is not necessarily a problem, as it will simply result in training the metamodel to mimic CFAST even in the more extreme configurations; however, it is important that application of the metamodel be within the CFAST validated ranges of applicability, or otherwise be justified if the ranges are exceeded.

Similarly, metamodel applicability is generally limited to the range of parameter space over which it was trained. In this thesis, the parameter space consists of rectangular rooms ranging from 10-35 meters long, 10-35 meters wide, 5-10 meters tall and other scenario characteristics described in Table 9. Caution would be warranted when using the resulting metamodel to extrapolate to fire scenarios exceeding these characteristics. Developing guidance for how metamodels of physics-based codes could safely be used for extrapolation may be an area for future research.

Metamodels are applicable only to the predicted quantities they are trained to predict. For example in this thesis, the *k*NN metamodel of CFAST predicts maximum upper layer temperature and its timing; whereas, CFAST itself calculates many other quantities such as lower layer temperature, radiative heat flux, smoke optical density. Separate metamodels would need to be developed if predicting these other quantities were required. One area of future research might be development of metamodels that can predict multiple disparate quantities.

Finally, in the heavily regulated nuclear power industry, the regulator (Nuclear Regulatory Commission in the United States) has not necessarily endorsed or considered metamodeling. High-fidelity physics-based codes are typically run for a manageable number of scenarios to support design and licensing applications.

However, the movement toward risk-informed regulation relies heavily on probabilistic risk assessments. An important aspect of risk-informed decision-making is understanding

uncertainty, and metamodel approximations of physics-based codes provide a feasible means of leveraging higher fidelity models during uncertainty quantification. For example, in the fire context, users are generally limited to algebraic models for uncertainty quantification; however, using a CFAST metamodel, while not as accurate as CFAST itself, can be more accurate than the algebraic models.

Several industry-level initiatives might be required in order to gain regulatory "endorsement" of, or some comfort level in, use of metamodels. First, a range of potentially acceptable applications would need to be identified. For example, probabilistic applications such as uncertainty analysis are good candidates; whereas certain design applications that require the most accurate (or conservative) physics-based representations may not be good candidates for metamodeling.

Ultimately, a consensus process for training, validating, and applying metamodels would be required. Such an industry consensus process might include guidance on which metamodel types are most appropriate for each of the physics-based models intended to be mimicked. Minimum requirements might be established on the scale and range of training data, as well as the use of cross validation. The process might impose limitations on the complexity (and transparency) of metamodels. And of course some minimum level of accuracy would be required, and the accuracy would need to be quantifiable so that it could be considered explicitly by metamodel applications. Depending on stakeholder comfort level, the consensus process could essentially be guidelines on development and use of metamodels, or if more control is desired, consensus metamodels could be developed, verified, and validated similar to the fire modeling validation efforts of NUREG-1824 (Hill et al., 2007) and NUREG-1934 (McGrattan et al., 2012).

## 3.12 FUTURE RESEARCH

Section 3.11 discusses the potential development of an industry consensus process for training, validating, and applying metamodels. Such a process would promote consistent and responsible application of metamodels. The following paragraphs discuss some specific areas of future research that could help generate interest and ultimately support an industry consensus process.

This thesis focused on metamodeling the physics of fire behavior, which primarily involve heat transfer and fluid mechanics. There are many other hazard models used for probabilistic risk assessments, such as modeling building response during an earthquake, modeling the flow and accumulation of water throughout a plant due to flood events, and modeling projectile motion of objects during high wind events. There are also a variety of physics-based models to estimate plant response during accident conditions, calculating quantities such as fuel cladding temperature as a function of time through an accident. Future work could broadly review each of the computationally intensive models used by the nuclear power industry for their potential to benefit from metamodeling.

The literature review performed for this thesis did not identify a significant body of guidance for determining which metamodels might be most suited for various types of applications. Consequently, this study followed a successive screening process where 25 of the available metamodels were initially tested, three were selected for further analysis, and one was ultimately successful. During this process, it became clear that certain metamodel types have

attributes making them more suited for some applications and less suited for others. For example, the linear models struggled to accurately mimic the non-linear physics of upper layer temperature development. Future work might develop guidance as to how to select promising metamodel types based on the attributes of the application.

Section 3.10 explores a tradeoff between accuracy and efficiency, leading to the following question: Under what circumstances does metamodeling become feasible and attractive? When the higher fidelity code runs quickly, as is the case for CFAST, it is more feasible to generate metamodel training data; however, the resulting metamodel can only improve overall efficiency for applications where an extremely large number of model runs is required. On the contrary, when the high fidelity model is computationally heavy, as is the case for CFD, metamodels have potential to improve efficiency; however, generating the training data becomes more difficult. Future research could explore these factors and develop guidance for quickly assessing whether a particular application might benefit from the metamodeling process.

The *k*NN model developed here was trained over a parameter space surrounding switchgear room fire scenarios, with the attributes defined in in Table 9. A natural question might be: To what extent can the resulting model extrapolate to scenarios outside the training parameter space? It is unclear at what point a metamodel becomes a *generalized* representation of the computations performed by the high fidelity model, and to what extent the metamodel accuracy is specific to the training space. Future research could develop designs of experiment focused on how the high-fidelity model performs its computation, as opposed to focusing on a particular scenario of interest, with the aim of developing generalized metamodels accurate over the full validation range of the high-fidelity model.

The predictor variables in this thesis are scalar quantities (maximum upper layer temperature, and time to reach maximum upper layer temperature). Many nuclear power plant accident analyses estimate the temporal and spatial evolution of quantities. In addition to introducing additional dimensionality, temporal and spatial evolution introduces correlation between parameters. For example, the upper layer temperature at one time step is highly correlated to the temperature at the previous time step. Future work could explore metamodel development for temporally and spatially evolving systems.

Finally, while metamodels are generally benchmarked against the models they attempt to mimic, future work might compare trained metamodel against higher fidelity models or even physical experiments. These exercises could be an important part of convincing regulatory bodies and standards organizations that metamodels can accurately represent the physical realities of interest.

# 4.0 CONCLUSIONS

This study explored the use of machine learning to generate metamodel approximations of a physics-based fire hazard model. The process involved scenario definition, design of experiment, generating training data by iteratively running the hazard model over a range of input space, exploratory data analysis and feature selection, initial testing and screening of a broad set of metamodel types, and finally metamodel selection and tuning.

The study identified several key factors that should be considered when metamodeling a physics-based computer code. First, the input space should be limited to a manageable scale and number of parameters; otherwise generating sufficient training data becomes infeasible. Second, there is a relationship between the physics being characterized and the metamodel types that will successfully mimic those physics. Heat transfer and fluid flow are highly non-linear, and therefore the linear-based metamodels struggle to emulate those phenomena. Finally, despite initial development costs, it is possible for the resulting metamodel to accurately mimic the physics-based code, and to run at a fraction of the computational expense. Once developed, the trained metamodel is portable and can easily be applied by many users over a range of modeling conditions.

A well-fitting *k*-nearest neighbor approximation of the computer fire model called CFAST was generated. The specific parameters predicted were compartment fire-generated

maximum upper layer temperature, and time to reach maximum upper layer temperature, over a range of input space typical of postulated nuclear power plant fire scenarios. The model centers on switchgear room fires, which have consistently been identified by probabilistic risk assessments as risk-significant. The resulting metamodel is applicable to other scenarios to the extent they are within the input space used for model training.

The Idaho National Laboratory software called RAVEN, which is open-source and available at their 'github' (Idaho National Laboratory, n.d.), was used to run a CFAST fire model 675,000 times over a full grid-sampled input space of seven (7) uncertain parameters described in Table 15.

**Table 15.** Input Parameter Space used for K-Nearest Neighbor Model Training

| Parameter | Range |
|---|---|
| Compartment Length | 10-35 meters |
| Compartment Width | 10-35 meters |
| Compartment Height | 5-10 meters |
| Fire Heat Release Rate | 30 unique profiles |
| Ambient Temperature | 297-311 Kelvin |
| Fire Height | 0-90% of room height |
| Ventilation Rate per Room Volume | 0.00047-0.00189 ($m^3$/s per $m^3$ of room volume) |

The input heat release rate profiles were decomposed into field-observable factors, such as the type of originating electrical cabinet, number of adjacent cabinets, and overhead cable tray configuration. In this sense the resulting *k*-nearest neighbors metamodel is actually an aggregate representation of two models (the translation of field-observable factors into heat release rate profiles, and the CFAST estimation of maximum upper layer temperature and its timing).

Twenty five (25) reduced order model types available within RAVEN were exercised using 80% (540,000) of the CFAST runs for training and the remaining 20% (135,000 runs) for model testing. The CFAST runs used for training and testing were selected randomly.

The linear models tested were generally unable to predict maximum upper layer temperature and its timing with reasonable accuracy. Tested models include: variations of ordinary linear regression, lasso, ridge, elastic net, least angle, orthogonal matching pursuit, and others. The poor fit of linear models is fundamentally because the physics of fire growth and compartment energy balance are non-linear. Fire growth follows an exponential profile, and the conservation of energy equations solved by CFAST are highly non-linear.

Initial testing of tree-based, neighbor-based, and support vector machine models showed promise and were selected for model tuning. Additional evaluation identified that the initially good fit of the tree-based models was due to over-fitting, specifically by not limiting the tree depth.

Support vector regression ultimately did not generate a reasonable fit, and this could in part be due to its complexity. The model was so slow to train that only 5,000 samples (<1% of the data population) could be used for training, which may have limited its prediction accuracy. Slow training also erodes the benefit of a reduced order model, especially for CFAST, which is a relatively efficient code.

*k*-nearest neighbor model tuning generated a $k$ =4 model that fit the vast majority of CFAST calculations within ±10% for both maximum upper layer temperature and its timing. This model shows good generalization due to the use of 10-fold cross validation, and testing on an dataset that was not used for training.

The resulting *k*NN model predictions were compared to those made by the algebraic models known as MQH and FPA. Both MQH and FPA were generally conservative relative to CFAST; whereas the *k*NN model very closely mimicked CFAST over the full range of sampled input space.

This comparison illustrated the potential of metamodels to improve modeling realism over the simpler models often selected for computational feasibility. While the *k*NN metamodel is a simplification of the higher fidelity CFAST code, the error introduced is quantifiable and can be explicitly considered in applications of the metamodel.

Finally, it is acknowledged that metamodels do come with some computational expense, particularly for training. Metamodel accuracy generally improves as a function of training sample size, and in this case the *k*NN model required 100,000 training samples to achieve sufficient accuracy. After this upfront training cost, the resulting trained metamodel is portable and can easily be applied by many users over a range of modeling conditions.

# APPENDIX A

# CODING

## A.1 XML CODE DEFINING THE RAVEN ANALYSIS

The <Simulation> block defines the RAVEN analysis, and verbosity is set to "debug" to facilitate troubleshooting during code development.

```
<Simulation verbosity="debug">
  .
  .
  .
</Simulation>
```

The <RunInfo> block defines the sequences of steps that RAVEN will execute. <WorkingDir> defines the working directory name, and <batchSize> is set to 1 indicating that computations will be performed serially (as opposed to with parallel processing).

```
<Simulation verbosity="debug">
  <RunInfo>
    <Sequence>SampleRunExport</Sequence>
    <WorkingDir>RAVEN_CFAST</WorkingDir>
    <batchSize>1</batchSize>
  </RunInfo>
```

The <Files> block defines identifies all files to be used in the RAVEN analysis. "RAVEN_CFAST.in" is the base CFAST input file (text format), and the "Fire*.o" files define each of the CFAST fire objects.

```
  <Files>
    <Input name="RAVEN_CFAST.in" type="">RAVEN_CFAST.in</Input>
    <Input name="Fire243.o" type="">Fire243.o</Input>
    <Input name="Fire244.o" type="">Fire244.o</Input>
    <Input name="Fire245.o" type="">Fire245.o</Input>
    <Input name="Fire246.o" type="">Fire246.o</Input>
    <Input name="Fire248.o" type="">Fire248.o</Input>
    <Input name="Fire250.o" type="">Fire250.o</Input>
    <Input name="Fire253.o" type="">Fire253.o</Input>
    <Input name="Fire256.o" type="">Fire256.o</Input>
    <Input name="Fire257.o" type="">Fire257.o</Input>
    <Input name="Fire258.o" type="">Fire258.o</Input>
    <Input name="Fire259.o" type="">Fire259.o</Input>
    <Input name="Fire260.o" type="">Fire260.o</Input>
    <Input name="Fire262.o" type="">Fire262.o</Input>
    <Input name="Fire265.o" type="">Fire265.o</Input>
    <Input name="Fire267.o" type="">Fire267.o</Input>
    <Input name="Fire311.o" type="">Fire311.o</Input>
    <Input name="Fire314.o" type="">Fire314.o</Input>
    <Input name="Fire317.o" type="">Fire317.o</Input>
    <Input name="Fire319.o" type="">Fire319.o</Input>
    <Input name="Fire320.o" type="">Fire320.o</Input>
    <Input name="Fire321.o" type="">Fire321.o</Input>
    <Input name="Fire323.o" type="">Fire323.o</Input>
    <Input name="Fire330.o" type="">Fire330.o</Input>
    <Input name="Fire335.o" type="">Fire335.o</Input>
    <Input name="Fire336.o" type="">Fire336.o</Input>
    <Input name="Fire337.o" type="">Fire337.o</Input>
    <Input name="Fire341.o" type="">Fire341.o</Input>
    <Input name="Fire343.o" type="">Fire343.o</Input>
    <Input name="Fire354.o" type="">Fire354.o</Input>
  </Files>
```

The `<Databases>` block defines the *.H5* database to which the RAVEN input and output results are stored. This *.H5* database is a hierarchical (as opposed to relational) HDF5 format, which is later post-processed using MATLAB.

```
<Databases>
  <HDF5 name="results_db" readMode="overwrite"/>
</Databases>
```

The `<Models>` block defines each of the external models that will be exercised by RAVEN, which in this case is CFAST. The executable file and command line arguments are specified so that CFAST can be run from the MSYS64 command line. This block is used by the RAVEN-CFAST interface written in Python.

```
<Models>
  <Code name="CFAST" subType="CFASTinterface">
    <executable>"C:\Program Files (x86)\CFAST6\CFAST"</executable>
    <clargs arg="cmd ///c" type="prepend"/>
    <clargs extension=".in" type="input"/>
  </Code>
</Models>
```

The `<Functions>` block declares user-defined functions, which are each defined in stand-alone python files. In this analysis, the `'CalcFloorArea'` function calculates the compartment floor area based on sampled length and width, the `'CalcFireHeight'` function determines the height of the fire above the floor based on a sampled fraction of the room height,

and the `'CalcHRRindex'` function simply returns the nearest integer value of each heat release rate sample (since the heat release rates are indexed).

```
  <Functions>
    <External name='FunFloorArea' file='FunFloorArea.py'>
        <variable>length</variable>
        <variable>width</variable>
    </External>
    <External name='FunFireHeight' file='FunFireHeight.py'>
        <variable>height</variable>
        <variable>fireHeightFract</variable>
    </External>
    <External name='FunHRRindex' file='FunHRRindex.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRpdf' file='FunHRRpdf.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray1' file='FunHRRtray1.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray2' file='FunHRRtray2.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray3' file='FunHRRtray3.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray4' file='FunHRRtray4.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray5' file='FunHRRtray5.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRtray6' file='FunHRRtray6.py'>
        <variable>hrr</variable>
    </External>
    <External name='FunHRRadj' file='FunHRRadj.py'>
        <variable>hrr</variable>
    </External>
  </Functions>
```

The <Distributions> block defines all input parameter distributions. Many distribution types are available, such as Gaussian, Gamma, and Weibull.

```
<Distributions>
  <Uniform name="lengthDist">
    <lowerBound>10</lowerBound>
    <upperBound>35</upperBound>
  </Uniform>
  <Uniform name="widthDist">
    <lowerBound>10</lowerBound>
    <upperBound>35</upperBound>
  </Uniform>
  <Uniform name="heightDist">
    <lowerBound>5</lowerBound>
    <upperBound>10</upperBound>
  </Uniform>
  <Uniform name="hrrDist">
    <lowerBound>0.5</lowerBound>
    <upperBound>30.5</upperBound>
  </Uniform>
  <Uniform name="tempAmbDist">
    <lowerBound>297</lowerBound>
    <upperBound>311</upperBound>
  </Uniform>
  <Uniform name="fireHeightFractDist">
    <lowerBound>0.0</lowerBound>
    <upperBound>0.9</upperBound>
  </Uniform>
  <Uniform name="ventPerVolDist">
    <lowerBound>0.00047</lowerBound>
    <upperBound>0.00189</upperBound>
  </Uniform>
</Distributions>
```

The <Samplers> block defines the sampler type to be used for each distribution. Many sampler types are available, such as grid, Monte Carlo, and adaptive. This analysis uses a uniform grid over each input parameter distribution.

```
  <Samplers>
    <Grid name="grid">
      <variable name="length">
        <distribution>lengthDist</distribution>
        <grid construction="equal" steps="9" type="CDF">0 1</grid>
      </variable>
      <variable name="width">
        <distribution>widthDist</distribution>
        <grid construction="equal" steps="9" type="CDF">0 1</grid>
      </variable>
      <variable name="floorArea">
        <function>FunFloorArea</function>
      </variable>
      <variable name="height">
        <distribution>heightDist</distribution>
        <grid construction="equal" steps="4" type="CDF">0 1</grid>
      </variable>
      <variable name="hrr">
        <distribution>hrrDist</distribution>
        <grid construction="equal" steps="2" type="CDF">0 0.067</grid>
      </variable>
      <variable name="hrrIndex">
        <function>FunHRRindex</function>
      </variable>
      <variable name="hrrPDF">
        <function>FunHRRpdf</function>
      </variable>
      <variable name="hrrTray1">
        <function>FunHRRtray1</function>
      </variable>
      <variable name="hrrTray2">
        <function>FunHRRtray2</function>
      </variable>
      <variable name="hrrTray3">
        <function>FunHRRtray3</function>
      </variable>
      <variable name="hrrTray4">
        <function>FunHRRtray4</function>
      </variable>
      <variable name="hrrTray5">
        <function>FunHRRtray5</function>
      </variable>
      <variable name="hrrTray6">
        <function>FunHRRtray6</function>
      </variable>
      <variable name="hrrAdj">
        <function>FunHRRadj</function>
      </variable>
      <variable name="tempAmb">
        <distribution>tempAmbDist</distribution>
        <grid construction="equal" steps="2" type="CDF">0 1</grid>
      </variable>
      <variable name="fireHeightFract">
        <distribution>fireHeightFractDist</distribution>
        <grid construction="equal" steps="4" type="CDF">0 1</grid>
```

```
      </variable>
      <variable name="fireHeight">
        <function>FunFireHeight</function>
      </variable>
      <variable name="ventPerVol">
        <distribution>ventPerVolDist</distribution>
        <grid construction="equal" steps="2" type="CDF">0 1</grid>
      </variable>
    </Grid>
  </Samplers>
```

The <Steps> block defines each of the steps that RAVEN will execute. The step called "SampleRunExport" identifies the CFAST input files, the sampler to be used, the external model (CFAST) to be run, and a "DataObject" to organize the sampled input parameters and associated CFAST results.

```
  <Steps>
    <MultiRun name="SampleRunExport">
        <Input class="Files" type="">RAVEN_CFAST.in</Input>
        <Input class="Files" type="">Fire243.o</Input>
        <Input class="Files" type="">Fire244.o</Input>
        <Input class="Files" type="">Fire245.o</Input>
        <Input class="Files" type="">Fire246.o</Input>
        <Input class="Files" type="">Fire248.o</Input>
        <Input class="Files" type="">Fire250.o</Input>
        <Input class="Files" type="">Fire253.o</Input>
        <Input class="Files" type="">Fire256.o</Input>
        <Input class="Files" type="">Fire257.o</Input>
        <Input class="Files" type="">Fire258.o</Input>
        <Input class="Files" type="">Fire259.o</Input>
        <Input class="Files" type="">Fire260.o</Input>
        <Input class="Files" type="">Fire262.o</Input>
        <Input class="Files" type="">Fire265.o</Input>
        <Input class="Files" type="">Fire267.o</Input>
        <Input class="Files" type="">Fire311.o</Input>
        <Input class="Files" type="">Fire314.o</Input>
        <Input class="Files" type="">Fire317.o</Input>
        <Input class="Files" type="">Fire319.o</Input>
        <Input class="Files" type="">Fire320.o</Input>
        <Input class="Files" type="">Fire321.o</Input>
        <Input class="Files" type="">Fire323.o</Input>
        <Input class="Files" type="">Fire330.o</Input>
        <Input class="Files" type="">Fire335.o</Input>
        <Input class="Files" type="">Fire336.o</Input>
```

```
        <Input class="Files" type="">Fire337.o</Input>
        <Input class="Files" type="">Fire341.o</Input>
        <Input class="Files" type="">Fire343.o</Input>
        <Input class="Files" type="">Fire354.o</Input>
        <Input class="Files" type="">Fire356.o</Input>
        <Sampler class="Samplers" type="Grid">grid</Sampler>
        <Model class="Models" type="Code">CFAST</Model>
        <Output class="Databases" type="HDF5">results_db</Output>
    </MultiRun>
  </Steps>
```

## A.2 PYTHON INTERFACE BETWEEN RAVEN AND CFAST

The following .py file was developed as an interface between RAVEN and CFAST. This file defines the `CFASTinterface` class, which has two methods: `generateCommand` and `createNewInput`. The `generateCommand` method generates the MSYS64 command line that executes CFAST with the specified input file. The `createNewInput` method generates a CFAST input file for each sampled set of inputs generated by RAVEN.

```
from __future__ import division, print_function, unicode_literals,
absolute_import
from CodeInterfaceBaseClass import CodeInterfaceBase
import warnings
warnings.simplefilter('default',DeprecationWarning)
if not 'xrange' in dir(__builtins__):
  xrange = range
import os
import sys

class CFASTinterface(CodeInterfaceBase):

  def generateCommand(self,inputFiles,executable,clargs=None, fargs=None):
    todo = ''
    todo += clargs['pre']+' '
    todo += executable
    todo+=' RAVEN_CFAST'
    outfile = 'RAVEN_CFAST_zone'
    returnCommand = [('parallel',todo)],outfile
    print('Execution Command: '+str(returnCommand[0]))
    return returnCommand
```

```
from __future__ import division, print_function, unicode_literals,
absolute_import
from CodeInterfaceBaseClass import CodeInterfaceBase
import warnings
warnings.simplefilter('default',DeprecationWarning)
if not 'xrange' in dir(__builtins__):
  xrange = range
import os
import sys

class CFASTinterface(CodeInterfaceBase):

  def generateCommand(self,inputFiles,executable,clargs=None, fargs=None):
    todo = ''
    todo += clargs['pre']+' '
    todo += executable
    todo+=' RAVEN_CFAST'
    outfile = 'RAVEN_CFAST_zone'
    returnCommand = [('parallel',todo)],outfile
    print('Execution Command: '+str(returnCommand[0]))
    return returnCommand

  def
createNewInput(self,currentInputFiles,origInputFiles,samplerType,**Kwargs):
    modDict = Kwargs['SampledVars']
    outfile=currentInputFiles[0]
    outfile.open('w')
    outfile.write('VERSN,6,RAVEN_CFAST\n')
    outfile.write('!!\n')
    outfile.write('!!Environmental Keywords\n')
    outfile.write('!!\n')
    outfile.write('TIMES,3600,50,10,10,0\n')
    outfile.write('EAMB,')
    outfile.write(str(modDict.get('tempAmb')))
    outfile.write(',101325,30.48\n')
    outfile.write('TAMB,')
    outfile.write(str(modDict.get('tempAmb')))
    outfile.write(',101325,30.48,50\n')
    outfile.write('CJET,WALLS\n')
    outfile.write('CHEMI,10,488\n')
    outfile.write('WIND,0,10,0.16\n')
    outfile.write('!!\n')
    outfile.write('!!Compartment keywords\n')
    outfile.write('!!\n')
    outfile.write('COMPA,Compartment1')
    outfile.write(',')
    outfile.write(str(modDict.get('length')))
    outfile.write(',')
    outfile.write(str(modDict.get('width')))
    outfile.write(',')
    outfile.write(str(modDict.get('height')))
    outfile.write(',')
    outfile.writelines('0,0,0,CONCRETE,CONCRETE,CONCRETE\n')
    outfile.write('!!\n')
    outfile.write('!!vent keywords\n')
```

```
    outfile.write('!!\n')
    outfile.write('HVENT,1,2,1,2,0.025,0,1,')
    outfile.write((str(modDict.get('length')/2-1)))
    outfile.write(',0,1,1\n')
    outfile.write('HVENT,1,2,2,2,0.025,0,1,')
    outfile.write((str(modDict.get('width')/2-1)))
    outfile.write(',0,2,1\n')
    outfile.write('HVENT,1,2,3,2,0.025,0,1,')
    outfile.write((str(modDict.get('length')/2-1)))
    outfile.write(',0,3,1\n')
    outfile.write('HVENT,1,2,4,2,0.025,0,1,')
    outfile.write((str(modDict.get('width')/2-1)))
    outfile.write(',0,4,1\n')
    outfile.write('MVENT,1,2,1,V,')
    outfile.write(str(0.75*modDict.get('height')))
    outfile.write(',1.0,V,')
    outfile.write(str(0.75*modDict.get('height')))
    outfile.write(',1.0,')

outfile.write(str(modDict.get('ventPerVol')*modDict.get('length')*modDict.get
('width')*modDict.get('height')))
    outfile.write(',200,300,1.0\n')
    outfile.write('MVENT,2,1,2,V,')
    outfile.write(str(0.75*modDict.get('height')))
    outfile.write(',1.0,V,')
    outfile.write(str(0.75*modDict.get('height')))
    outfile.write(',1.0,')

outfile.write(str(modDict.get('ventPerVol')*modDict.get('length')*modDict.get
('width')*modDict.get('height')))
    outfile.write(',200,300,1.0\n')
    outfile.write('EVENT,M,1,2,1,600,0,1\n')
    outfile.write('EVENT,M,2,1,2,600,0,1\n')
    outfile.write('!!\n')
    outfile.write('!!fire keywords\n')
    outfile.write('!!\n')
    if modDict.get('hrrIndex')==1:
        fireObject='Fire243'
    if modDict.get('hrrIndex')==2:
        fireObject='Fire244'
    if modDict.get('hrrIndex')==3:
        fireObject='Fire245'
    if modDict.get('hrrIndex')==4:
        fireObject='Fire246'
    if modDict.get('hrrIndex')==5:
        fireObject='Fire248'
    if modDict.get('hrrIndex')==6:
        fireObject='Fire250'
    if modDict.get('hrrIndex')==7:
        fireObject='Fire253'
    if modDict.get('hrrIndex')==8:
        fireObject='Fire256'
    if modDict.get('hrrIndex')==9:
        fireObject='Fire257'
    if modDict.get('hrrIndex')==10:
```

```
            fireObject='Fire258'
        if modDict.get('hrrIndex')==11:
            fireObject='Fire259'
        if modDict.get('hrrIndex')==12:
            fireObject='Fire260'
        if modDict.get('hrrIndex')==13:
            fireObject='Fire262'
        if modDict.get('hrrIndex')==14:
            fireObject='Fire265'
        if modDict.get('hrrIndex')==15:
            fireObject='Fire267'
        if modDict.get('hrrIndex')==16:
            fireObject='Fire311'
        if modDict.get('hrrIndex')==17:
            fireObject='Fire314'
        if modDict.get('hrrIndex')==18:
            fireObject='Fire317'
        if modDict.get('hrrIndex')==19:
            fireObject='Fire319'
        if modDict.get('hrrIndex')==20:
            fireObject='Fire320'
        if modDict.get('hrrIndex')==21:
            fireObject='Fire321'
        if modDict.get('hrrIndex')==22:
            fireObject='Fire323'
        if modDict.get('hrrIndex')==23:
            fireObject='Fire330'
        if modDict.get('hrrIndex')==24:
            fireObject='Fire335'
        if modDict.get('hrrIndex')==25:
            fireObject='Fire336'
        if modDict.get('hrrIndex')==26:
            fireObject='Fire337'
        if modDict.get('hrrIndex')==27:
            fireObject='Fire341'
        if modDict.get('hrrIndex')==28:
            fireObject='Fire343'
        if modDict.get('hrrIndex')==29:
            fireObject='Fire354'
        if modDict.get('hrrIndex')==30:
            fireObject='Fire356'
        outfile.write('OBJECT,')
        outfile.write(fireObject)
        outfile.write(',1,')
        outfile.write(str(0.5*modDict.get('length')))
        outfile.write(',')
        outfile.write(str(0.5*modDict.get('width')))
        outfile.write(',')
        outfile.write(str(modDict.get('fireHeight')))
        outfile.write(',1,1,0,0,0,1\n')
        outfile.close()
        return currentInputFiles
```

## A.3 USER-DEFINED PYTHON FUNCTIONS

The *FunFireHeight.py* user-defined function defines the fire elevation above floor level, which is based on a sampled fraction, between zero (0) and 90%, of the room height.

```
def evaluate(self):
    return self.height*self.fireHeightFract
```

The *FunFloorArea.py* user-defined function calculates the compartment floor area based on its sampled length and width.

```
def evaluate(self):
    return self.length*self.width
```

The *FunHRRadj.py* user-defined function extracts the number of adjacent cabinets from the *IgnitSourceData.csv* data file.

```
import numpy as np
def evaluate(self):
   data = np.loadtxt('IgnitSourceData.csv',delimiter=',')
   data=np.array(data)
   hrrIndex=int(round(self.hrr))
   return data[8,hrrIndex-1]
```

The *FunHRRdata.py* user-defined function extracts various heat release rate parameters from the *IgnitSourceData.csv* data file.

```
import numpy as np
def evaluate(self):
   data = np.loadtxt('IgnitSourceData.csv',delimiter=',')
   data=np.array(data)
   hrrIndex=int(round(self.hrr))
   return data[8,hrrIndex-1]
```

The *FunHRRindex.py* user-defined function simply rounds the sampled heat release rate to the nearest integer, since the heat release rate profiles are indexed.

```
def evaluate(self):
    return int(round(self.hrr))
```

The *FunHRRpdf.py* user-defined function extracts the heat release rate probability density function identifier from the *IgnitSourceData.csv* data file.

```
import numpy as np
def evaluate(self):
   data = np.loadtxt('IgnitSourceData.csv',delimiter=',')
   data=np.array(data)
   hrrIndex=int(round(self.hrr))
   return data[1,hrrIndex-1]
```

The *FunHRRtray*.py* user-defined functions extract the number of cable trays at each elevation in the tray stack from the *IgnitSourceData.csv* data file.

```
import numpy as np
def evaluate(self):
   data = np.loadtxt('IgnitSourceData.csv',delimiter=',')
   data=np.array(data)
   hrrIndex=int(round(self.hrr))
   return data[2,hrrIndex-1]
```

## A.4 MODIFIED .CSV LOADER

The following lines in the *Csv_loader.py* file were modified to accommodate the CFAST-generated .CSV output files, in which the first two rows contain header information, and the data starts on the third row. These modifications were necessary because the base *Csv_loader.py* file that comes with the RAVEN framework expects the data start on the second row.

```
  def loadCsvFile(self,myFile):
    """
      Function to load a csv file into a numpy array (2D)
      It also retrieves the headers
      The format of the csv must be:
      STRING,STRING,STRING,STRING
      FLOAT ,FLOAT ,FLOAT ,FLOAT
      ...
      FLOAT ,FLOAT ,FLOAT ,FLOAT
      @ In, fileIn, string, Input file name (absolute path)
      @ Out, data, numpy.ndarray, the loaded data
    """
    # open file
    myFile.open(mode='rb')
    # read the field names
    #head = myFile.readline().decode()
    lines=myFile.readlines() #MODIFICATION
    head=lines[1] #MODIFICATION
    self.allFieldNames = head.split(',')
    for index in range(len(self.allFieldNames)): self.allFieldNames[index] =
self.allFieldNames[index].strip()
```

```
    # load the table data (from the csv file) into a numpy nd array
    data =
np.loadtxt(myFile,dtype='float',delimiter=',',ndmin=2,skiprows=2,usecols=(0,1
,2,3,4,5,6,7,8,9)) #MODIFICATION
    # close file
    myFile.close()
    return data
```

## A.5 MODIFIED H5 PYTHON INTERFACE

Line 269 of the *h5py_interface_creator.py* file was similarly modified to accommodate the CFAST-generated .CSV output files, in which the first two rows contain header information, and the data starts on the third row. This modification was necessary because the base *h5py_interface_creator.py* file that comes with the RAVEN framework expects the data start on the second row.

```
data =
np.loadtxt(f,dtype='float',delimiter=',',ndmin=2,skiprows=2,usecols=(0,1,2,3,
4,5,6,7,8,9,10,11,12)) #MODIFICATION
```

## A.6 MATLAB HDF5 PROCESSING SCRIPT

The following MATLAB script imports the HDF5 database of results generated by RAVEN and creates various plots use in this report. Note that HDF5 is a hierarchical (versus relational) database format that cannot be accessed by Microsoft Access or Excel.

```
%% FRONTMATTER
clc
clear all
```

```
close all
set(0,'DefaultTextFontname', 'Times New Roman');
set(0,'DefaultAxesFontname','Times New Roman');
set(0,'defaultfigurecolor',[1 1 1]);

%Initialize
filename=strcat('DatabaseStorage\',uigetfile('*.h5'));
%filename='GoldStandardRun\run2_db.h5';
h5disp(filename);
fileinfo=hdf5info(filename);
toplevel=fileinfo.GroupHierarchy;
numSamples=length(toplevel.Groups.Groups);
numTimeSteps=360;
sampleIDs=cell(numSamples,1);
ravenOUTdata=zeros(numTimeSteps,13,numSamples);
%MATLAB2015b

ravenINdata=zeros(numSamples,length(eval(cell2mat((h5readatt(filename,topleve
l.Groups.Groups(1).Name,'inputSpaceValues'))))));
%MATLAB2016b

%ravenINdata=zeros(numSamples,length((h5readatt(filename,toplevel.Groups.Grou
ps(1).Name,'inputSpaceValues'))));

%Import RAVEN Input Data from HDF5 into MATLAB Matrix
ravenINdataNames=h5readatt(filename,toplevel.Groups.Groups(1).Name,'inputSpac
eHeaders');
ravenINdataNames=transpose(ravenINdataNames);
for sample=1:numSamples
    %MATLAB2015b

sampledInputValues=h5readatt(filename,toplevel.Groups.Groups(sample).Name,'in
putSpaceValues'); %produces cell
        sampledInputValues=eval(cell2mat(sampledInputValues)); %converts cell
to matrix
    %MATLAB2016b

%sampledInputValues=h5readatt(filename,toplevel.Groups.Groups(sample).Name,'i
nputSpaceValues');
    ravenINdata(sample,:)=sampledInputValues;
    sampleIDs{sample,1}=toplevel.Groups.Groups(sample).Name;
end
clear sampledInputValues

%Translate Cabinet HRR PDF Index to 98th Percentile Peak HRR
for sample=1:numSamples
    if ravenINdata(sample,2)==1
        ravenINdata(sample,2)=211;
    elseif ravenINdata(sample,2)==9
        ravenINdata(sample,2)=170;
    elseif ravenINdata(sample,2)==10
        ravenINdata(sample,2)=130;
    end
end
```

```
%Import RAVEN Output Data from HDF5 into MATLAB Matrix
ravenOUTdataNames={'Time(s)','ULT(C)','LLT(C)','HGT(m)','PRS(Pa)','ULOD(1/m)'
,'LLOD(1/m)','HRR(kW)','FLHGT(m)','FBASE(m)','FAREA(m2)','HVENT_1(m2)','HVENT
_2(m2)'};
for sample=1:numSamples
    tempDataSetName=toplevel.Groups.Groups(sample).Datasets.Name;
    tempDataSet = transpose(h5read(filename,tempDataSetName));
    ravenOUTdata(:,:,sample)=tempDataSet(1:360,:);
end
clear tempDataSet tempDataSetName

% Collect Predictor and Response Variables into one .CSV File for RAVEN
responseVars=zeros(numSamples,2);
ravenROMdata=zeros(numSamples,18+2);
for sample=1:numSamples
    [M,I]=max(ravenOUTdata(:,2,sample));
    responseVars(sample,1)=M;
    responseVars(sample,2)=ravenOUTdata(I,1,sample)./60;
    ravenROMdata(sample,1:18)=ravenINdata(sample,:);
    ravenROMdata(sample,19:20)=responseVars(sample,:);
end
ravenROMdata=array2table(ravenROMdata);
ravenROMdata.Properties.VariableNames = {'hrrIndex' 'hrrPDF' 'tempAmb' 'hrr'
'floorArea' 'width' 'length' 'ventPerVol' 'fireHeightFract' 'fireHeight'
'hrrTray4' 'hrrTray5' 'hrrTray6' 'height' 'hrrAdj' 'hrrTray1' 'hrrTray2'
'hrrTray3' 'maxULT' 'timeToMaxULT'};
runDescriptor=char(inputdlg('Type Run Desriptor for Results SaveAs
Filenames'));
writetable(ravenROMdata,strcat('C:\msys64\home\worrelcl\raven\clarence\cfast\
RAVEN_CFAST\GoldStandardRun\Run',runDescriptor,'ravenROMdata.csv'));
hist(ravenINdata(:,1));
```

## A.7 MATLAB FIGURES OF CFAST RESULTS

The following MATLAB script generates plots of various CFAST-calculated quantities such as upper layer temperature, upper layer height, and compartment pressure as functions of time.

```
%% Initialize
clc; clear all; close all;
set(0,'DefaultTextFontname', 'Times New Roman');
```

```
set(0,'DefaultAxesFontname','Times New Roman');
set(0,'defaultfigurecolor',[1 1 1]);
cd ('C:\msys64\home\worrelcl\raven\clarence\CFAST\CFAST_Output_Figures');
filename='MC500_results.h5';
h5disp(filename);
fileinfo=hdf5info(filename);
toplevel=fileinfo.GroupHierarchy;
numSamples=length(toplevel.Groups.Groups);
numTimeSteps=360;
sampleIDs=cell(numSamples,1);
ravenOUTdata=zeros(numTimeSteps,13,numSamples);
%MATLAB2015b

ravenINdata=zeros(numSamples,length(h5readatt(filename,toplevel.Groups.Groups
(1).Name,'inputSpaceHeaders')));
%MATLAB2016b

%ravenINdata=zeros(numSamples,length((h5readatt(filename,toplevel.Groups.Grou
ps(1).Name,'inputSpaceValues'))));

% Import RAVEN Input Data from HDF5 into MATLAB Matrix
ravenINdataNames=h5readatt(filename,toplevel.Groups.Groups(1).Name,'inputSpac
eHeaders');
ravenINdataNames=transpose(ravenINdataNames);
for sample=1:numSamples
    %MATLAB2015b

sampledInputValues=h5readatt(filename,toplevel.Groups.Groups(sample).Name,'in
putSpaceValues'); %produces cell
        sampledInputValues=eval(cell2mat(sampledInputValues)); %converts cell
to matrix
    %MATLAB2016b

%sampledInputValues=h5readatt(filename,toplevel.Groups.Groups(sample).Name,'i
nputSpaceValues');
    ravenINdata(sample,:)=sampledInputValues;
    sampleIDs{sample,1}=toplevel.Groups.Groups(sample).Name;
end
clear sampledInputValues

%Import RAVEN Output Data from HDF5 into MATLAB Matrix
ravenOUTdataNames={'Time(s)','ULT(C)','LLT(C)','HGT(m)','PRS(Pa)','ULOD(1/m)'
,'LLOD(1/m)','HRR(kW)','FLHGT(m)','FBASE(m)','FAREA(m2)','HVENT_1(m2)','HVENT
_2(m2)'};
for sample=1:numSamples
    tempDataSetName=toplevel.Groups.Groups(sample).Datasets.Name;
    tempDataSet = transpose(h5read(filename,tempDataSetName));
    ravenOUTdata(:,:,sample)=tempDataSet(1:360,:);
end
clear tempDataSet tempDataSetName

%% PLOT CABINET FIRE HRR
figure

%Cabinet Fire HRR (Prescribed Input)
```

```
subplot(2,1,1)
hrrInputFile=strcat(pwd,'\figures.xlsx');
inputQ=xlsread(hrrInputFile, 'HRRdata', 'B18:XFD137');

[numRow,numCol]=size(inputQ);
for col=1:numCol
    if mod(col,2)==1
        plot(inputQ(:,col),inputQ(:,col+1)); hold on;
    end
end
set(gca, 'fontweight', 'normal')
title('Prescribed Heat Release Rate (CFAST Input)','FontSize', 12); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Heat Release Rate (kW)', 'fontweight','bold', 'FontSize', 12); hold
on;
xlim([0 60]); ylim([0 2500]);

subplot(2,1,2) %Cabinet Fire HRR (RAVEN-CFAST Output)
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,8,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Realized Heat Release Rate (CFAST Output)','FontSize', 12); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Heat Release Rate (kW)', 'fontweight','bold', 'FontSize', 12); hold
on;
xlim([0 60]); ylim([0 2500]);

clear col hrrInputFile sample col row numCol numRow

%% PLOT UPPER LAYER TEMPERATURE RESULTS (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,2,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Upper Layer Temperature vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Upper Layer Temperature (C)', 'fontweight','bold', 'FontSize', 12);
hold on;
xlim([0 60]); ylim([0 400]);

clear sample

%% PLOT LOWER LAYER TEMPERATURE RESULTS  (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,3,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Lower Layer Temperature vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (s)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Lower Layer Temperature (C)', 'fontweight','bold', 'FontSize', 12);
hold on;
xlim([0 60]); ylim([0 400]);
```

```
clear sample

%% PLOT UPPER LAYER HEIGHT RESULTS  (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,4,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Upper Layer Height vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Upper Layer Height (m)', 'fontweight','bold', 'FontSize', 12); hold
on;
xlim([0 60]); ylim([0 10]);

clear sample

%% PLOT COMPARTMENT PRESSURE RESULTS  (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,5,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Compartment Pressure vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Compartment Pressure (Pa)', 'fontweight','bold', 'FontSize', 12);
hold on;
xlim([0 60]); %ylim([0 10]);

clear sample

%% PLOT UPPER LAYER OPTICAL DENSITY RESULTS  (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,6,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Optical Density vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Upper Layer Optical Denisty (1/m)', 'fontweight','bold', 'FontSize',
12); hold on;
xlim([0 60]); ylim([0 50]);

clear sample

%% PLOT FLAME HEIGHT RESULTS  (RAVEN-CFAST Output)
figure
for sample=1:numSamples
    plot(ravenOUTdata(:,1,sample)./60,ravenOUTdata(:,9,sample)); hold on;
end
set(gca, 'fontweight', 'normal')
title('Flame Height vs. Time (500 Runs)','FontSize', 14); hold on;
xlabel('Time (min)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Flame Height (m)', 'fontweight','bold', 'FontSize', 12); hold on;
xlim([0 60]); %ylim([0 3]); clear sample
```

## A.8 ASSEMBLY OF RAVEN-CFAST OUTPUT FOR ROM TRAINING

The following MATLAB script assembles the input and output parameter values of the RAVEN-CFAST simulation in preparation for ROM training and testing. This script also generates histograms of the input parameters.

```
%% CONSOLIDATE RAVEN OUTPUT FILES INTO TRAINING AND TEST SETS

%% Initialize
clc; clear all; close all;
set(0,'DefaultTextFontname', 'Times New Roman');
set(0,'DefaultAxesFontname','Times New Roman');
set(0,'defaultfigurecolor',[1 1 1]);
cd ('C:\msys64\home\worrelcl\raven\clarence\CFAST\ROM\ROM_Pre_Process');
files = dir('Run*ravenROMdata.csv');

%% Read header
fid = fopen(files(1).name);
header = textscan(fid,'%s',20,'Delimiter',',');
header=header{1};
fclose(fid); clear fid ans;

%% Read all data into one table
allData = csvread(files(1).name,1);
for n = 2:numel(files)
    temp = csvread(files(n).name,1);
    allData = vertcat(allData, temp);
end
clear files n temp

%% Center and scale all parameters
% Comment / uncomment as desired
% for col=1:length(header)
%     allData(:,col)=(allData(:,col)-
mean(allData(:,col)))./std(allData(:,col));
% end
% clear col

%% This code section added because it was discovered, after the
%  GoldStandard run was completed, that the RAVEN-CFAST code was
```

```
%  grabbing the assocated tray counts, cabinet hrr PDF, and a number of
%  adjacent cabinets incorrectly. These parameters are predictors, but
%  they were not actually needed directly for the CFAST calculations. The
%  CFAST data generated by RAVEN has been reviewed and is correct. The
%  code below replaces the affected parameters with the correct values,
%  using HRRindex as the key. Affected parameters include: hrrPDF, hrrTray1,
%  hrrTray2, hrrTray3, hrrTray4, hrrTray5, hrrTray6, and hrrAdj. Also added
%  fireObject for clarification.

ignitSourceData=csvread('IgnitSourceData.csv');
    %Note the column numbers of ignitSourceData correspond to HRRindex
numSamples=length(allData);
for row=1:numSamples
    hrrIndex=allData(row,1);
    allData(row,21)=ignitSourceData(1,hrrIndex); %fireObject
    allData(row,2)=ignitSourceData(2,hrrIndex);  %hrrPDF
    allData(row,16)=ignitSourceData(3,hrrIndex); %hrrTray1
    allData(row,17)=ignitSourceData(4,hrrIndex); %hrrTray2
    allData(row,18)=ignitSourceData(5,hrrIndex); %hrrTray3
    allData(row,11)=ignitSourceData(6,hrrIndex); %hrrTray4
    allData(row,12)=ignitSourceData(7,hrrIndex); %hrrTray5
    allData(row,13)=ignitSourceData(8,hrrIndex); %hrrTray6
    allData(row,15)=ignitSourceData(9,hrrIndex); %hrrAdj
end
header{21}='fireObject'

%Translate Cabinet HRR PDF Index to 98th Percentile Peak HRR
for sample=1:numSamples
    if allData(sample,2)==1
        allData(sample,2)=211;
    elseif allData(sample,2)==9
        allData(sample,2)=170;
    elseif allData(sample,2)==10
        allData(sample,2)=130;
    end
end

clear col hrrIndex numSamples row sample

%% Split randomly into train and test data
numSamples=length(allData);
idx=randperm(numSamples);
trainData=allData(idx(1:0.8*numSamples),:);
testData=allData(idx(1:0.2*numSamples),:);
clear idx numRecords

% Write training data to CSV file
fileID = fopen('_trainData.csv','w');
fprintf(fileID,'%s,',header{1:end-1,1});
fprintf(fileID,'%s\n',header{end,1});
fclose(fileID);
dlmwrite('_trainData.csv',trainData,'delimiter',',','-append');
clear fileID ans

% Write testing data to CSV file
```

```
fileID = fopen('_testData.csv','w');
fprintf(fileID,'%s,',header{1:end-1,1});
fprintf(fileID,'%s\n',header{end,1});
fclose(fileID);
dlmwrite('_testData.csv',testData,'delimiter',',','-append');
clear fileID ans

%% HISTOGRAMS OF INPUT PARAMETERS FOR ENTIRE DATA SET
figure

subplot(3,3,1) %Room Length
col=regexp(header, regexptranslate('wildcard','length*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Room Length (m)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,2) %Room Width
col=regexp(header, regexptranslate('wildcard','width*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Room Width (m)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,3) %Room Height
col=regexp(header, regexptranslate('wildcard','height*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col),5)
set(gca, 'fontweight', 'normal')
xlabel('Room Height (m)','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,4) %Ambient Temperature
col=regexp(header, regexptranslate('wildcard','tempAmb*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Ambient Temperature (K)','fontweight','bold', 'FontSize', 12); hold
on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,5) %Vent Rate per Room Volume
col=regexp(header, regexptranslate('wildcard','ventPerVol*')); %finds
parameter location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Vent Rate per Room Vol (m^3/s-m^3)','fontweight','bold', 'FontSize',
12); hold on;
```

```
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,6) %Fire Height Fraction
col=regexp(header, regexptranslate('wildcard','fireHeightFract*')); %finds
parameter location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col),5)
set(gca, 'fontweight', 'normal')
xlabel('Fire Height (Fraction)','fontweight','bold', 'FontSize', 12); hold
on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,7) %Heat Release Rate Index
col=regexp(header, regexptranslate('wildcard','hrrIndex*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Heat Release Rate Index','fontweight','bold', 'FontSize', 12); hold
on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

figure

subplot(3,3,1) %Heat Release Rate PDF
col=regexp(header, regexptranslate('wildcard','hrrPDF*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Cabinet Peak Heat Release Rate ','fontweight','bold', 'FontSize',
12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,2) %Tray1
col=regexp(header, regexptranslate('wildcard','hrrTray1*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 1','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,3) %Tray2
col=regexp(header, regexptranslate('wildcard','hrrTray2*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 2','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,4) %Tray3
col=regexp(header, regexptranslate('wildcard','hrrTray3*')); %finds parameter
location, dealing with its trailing spaces
```

```
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 3','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,5) %Tray4
col=regexp(header, regexptranslate('wildcard','hrrTray4*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 4','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,6) %Tray5
col=regexp(header, regexptranslate('wildcard','hrrTray5*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 5','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,7) %Tray6
col=regexp(header, regexptranslate('wildcard','hrrTray6*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Tray Elev 6','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

subplot(3,3,8) %AdjCab
col=regexp(header, regexptranslate('wildcard','hrrAdj*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Num Adjacent Cabinets','fontweight','bold', 'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;

%% HISTOGRAMS OF RESPONSE VARIABLES FOR ENTIRE DATA SET
figure

subplot(2,1,1) %Max ULT
col=regexp(header, regexptranslate('wildcard','maxULT*')); %finds parameter
location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Max Upper Layer Temperature (C)','fontweight','bold', 'FontSize',
12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;
```

```
subplot(2,1,2) %Time to Max ULT
col=regexp(header, regexptranslate('wildcard','timeToMaxULT*')); %finds
parameter location, dealing with its trailing spaces
col(cellfun('isempty',col))={0}; col = find([col{:}] == 1);
hist(allData(:,col))
set(gca, 'fontweight', 'normal')
xlabel('Time (min) to Max Upper Layer Temperature','fontweight','bold',
'FontSize', 12); hold on;
ylabel('Count', 'fontweight','bold', 'FontSize', 12); hold on;
```

## A.9 RAVEN REDUCED ORDER MODEL INITIAL TRAINING AND TESTING

The following RAVEN input file, written in XML, performs an initial training and testing of twenty five (25) reduced order model types available within RAVEN.

```
<?xml version="1.0" ?>
<Simulation verbosity="debug">

  <RunInfo>
    <WorkingDir>ROM/ROM_Post_Process/InitialTesting(of26)</WorkingDir>
    <Sequence>
        read_train_data,
        train_ROMmsrA,test_ROMmsrA,
        train_ROMmsrB,test_ROMmsrB,
        train_ROMndInvDW,test_ROMndInvDW,
        train_ROMbayesRidge,test_ROMbayesRidge,
        train_ROMelasticNet,test_ROMelasticNet,
        train_ROMelasticNetCV,test_ROMelasticNetCV,
        train_ROMlars,test_ROMlars,
        train_ROMlarsCV,test_ROMlarsCV,
        train_ROMlasso,test_ROMlasso,
        train_ROMlassoCV,test_ROMlassoCV,
        train_ROMlassoLars,test_ROMlassoLars,
        train_ROMlassoLarsCV,test_ROMlassoLarsCV,
        train_ROMlassoLarsIC,test_ROMlassoLarsIC,
        train_ROMlinearReg,test_ROMlinearReg,
        train_ROMmultiTaskLasso,test_ROMmultiTaskLasso,
        train_ROMmultiTaskElasticNet,test_ROMmultiTaskElasticNet,
        train_ROMomp,test_ROMomp,
        train_ROMompCV,test_ROMompCV,
        train_ROMpassAggReg,test_ROMpassAggReg,
        train_ROMridge,test_ROMridge,
        train_ROMridgeCV,test_ROMridgeCV,
        train_ROMsgdRegressor,test_ROMsgdRegressor,
        train_ROMsvr,test_ROMsvr,
```

```
        train_ROMknnReg,test_ROMknnReg,
        train_ROMradiusNeighReg,test_ROMradiusNeighReg,
        train_ROMdecTreeReg,test_ROMdecTreeReg,
        train_ROMextraTreeReg,test_ROMextraTreeReg
    </Sequence>
    <batchSize>1</batchSize>
  </RunInfo>

  <Files>
    <Input name="trainROM">_trainData.csv</Input>
    <Input name="testROM">_testData.csv</Input>
  </Files>

  <Models>
    <ROM name="ROMmsrA" subType="MSR">

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT</Target>
        <simplification>0.0</simplification>
        <persistence>difference</persistence>
        <gradient>steepest</gradient>
        <graph>beta skeleton</graph>
        <beta>1</beta>
        <knn>8</knn>
        <partitionPredictor>kde</partitionPredictor>
        <kernel>gaussian</kernel>
        <smooth/>
        <bandwidth>0.2</bandwidth>
    </ROM>
    <ROM name="ROMmsrB" subType="MSR">

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>timeToMaxULT</Target>
        <simplification>0.0</simplification>
        <persistence>difference</persistence>
        <gradient>steepest</gradient>
        <graph>beta skeleton</graph>
        <beta>1</beta>
        <knn>8</knn>
        <partitionPredictor>kde</partitionPredictor>
        <kernel>gaussian</kernel>
        <smooth/>
        <bandwidth>0.2</bandwidth>
    </ROM>
    <ROM name="ROMndInvDW" subType="NDinvDistWeight">

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <p>3</p>
    </ROM>
    <ROM name='ROMbayesRidge' subType='SciKitLearn'>
```

```
<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|BayesianRidge</SKLtype>
    </ROM>
    <ROM name='ROMelasticNet' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|ElasticNet</SKLtype>
    </ROM>
    <ROM name='ROMelasticNetCV' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|ElasticNetCV</SKLtype>
    </ROM>
    <ROM name='ROMlars' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|Lars</SKLtype>
    </ROM>
    <ROM name='ROMlarsCV' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LarsCV</SKLtype>
    </ROM>
    <ROM name='ROMlasso' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|Lasso</SKLtype>
    </ROM>
    <ROM name='ROMlassoCV' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LassoCV</SKLtype>
    </ROM>
    <ROM name='ROMlassoLars' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LassoLars</SKLtype>
    </ROM>
    <ROM name='ROMlassoLarsCV' subType='SciKitLearn'>
```

```
<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LassoLarsCV</SKLtype>
    </ROM>
    <ROM name='ROMlassoLarsIC' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LassoLarsIC</SKLtype>
    </ROM>
    <ROM name='ROMlinearReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|LinearRegression</SKLtype>
        <fit_intercept>True</fit_intercept>
        <normalize>False</normalize>
    </ROM>
    <ROM name='ROMmultiTaskLasso' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|MultiTaskLasso</SKLtype>
    </ROM>
    <ROM name='ROMmultiTaskElasticNet' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
    <Target>maxULT,timeToMaxULT</Target>
    <SKLtype>linear_model|MultiTaskElasticNet</SKLtype>
    </ROM>
    <ROM name='ROMomp' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|OrthogonalMatchingPursuit</SKLtype>
    </ROM>
    <ROM name='ROMompCV' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|OrthogonalMatchingPursuitCV</SKLtype>
    </ROM>
    <ROM name='ROMpassAggReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
```

```
        <SKLtype>linear_model|PassiveAggressiveRegressor</SKLtype>
    </ROM>
    <!--ROM name='ROMperceptron' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|Perceptron</SKLtype>
    </ROM-->
    <ROM name='ROMridge' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|Ridge</SKLtype>
    </ROM>
    <ROM name='ROMridgeCV' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|RidgeCV</SKLtype>
    </ROM>
    <ROM name='ROMsgdRegressor' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>linear_model|SGDRegressor</SKLtype>
    </ROM>
    <ROM name='ROMsvr' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>svm|SVR</SKLtype>
    </ROM>
    <ROM name='ROMknnReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>neighbors|KNeighborsRegressor</SKLtype>
    </ROM>
    <ROM name='ROMradiusNeighReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>neighbors|RadiusNeighborsRegressor</SKLtype>
    </ROM>
    <ROM name='ROMdecTreeReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
```

```
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>tree|DecisionTreeRegressor</SKLtype>
    </ROM>
    <ROM name='ROMextraTreeReg' subType='SciKitLearn'>

<Features>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hr
rTray6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Features>
        <Target>maxULT,timeToMaxULT</Target>
        <SKLtype>tree|ExtraTreeRegressor</SKLtype>
    </ROM>

  </Models>

  <Samplers>
    <CustomSampler name="customSamplerFile">
        <Source class="Files" type="">testROM</Source>
        <variable name="hrrIndex"/>
        <variable name="hrrPDF"/>
        <variable name="tempAmb"/>
        <variable name="hrr"/>
        <variable name="floorArea"/>
        <variable name="width"/>
        <variable name="length"/>
        <variable name="ventPerVol"/>
        <variable name="fireHeightFract"/>
        <variable name="fireHeight"/>
        <variable name="hrrTray4"/>
        <variable name="hrrTray5"/>
        <variable name="hrrTray6"/>
        <variable name="height"/>
        <variable name="hrrAdj"/>
        <variable name="hrrTray1"/>
        <variable name="hrrTray2"/>
        <variable name="hrrTray3"/>
        <variable name="maxULT"/>
        <variable name="timeToMaxULT"/>
    </CustomSampler>
  </Samplers>

  <Steps>
    <IOStep name="read_train_data">
      <Input class="Files" type="">trainROM</Input>
      <Output class="DataObjects" type="PointSet">trainROM_pointset</Output>
    </IOStep>

    <RomTrainer name="train_ROMmsrA">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMmsrA</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMmsrB">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMmsrB</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMndInvDW">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
```

```
    <Output class="Models" type="ROM">ROMndInvDW</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMbayesRidge">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMbayesRidge</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMelasticNet">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMelasticNet</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMelasticNetCV">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMelasticNetCV</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlars">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlars</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlarsCV">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlarsCV</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlasso">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlasso</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlassoCV">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlassoCV</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlassoLars">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlassoLars</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlassoLarsCV">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlassoLarsCV</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlassoLarsIC">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlassoLarsIC</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMlinearReg">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMlinearReg</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMmultiTaskLasso">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMmultiTaskLasso</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMmultiTaskElasticNet">
    <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
    <Output class="Models" type="ROM">ROMmultiTaskElasticNet</Output>
  </RomTrainer>
  <RomTrainer name="train_ROMomp">
```

```
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMomp</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMompCV">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMompCV</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMpassAggReg">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMpassAggReg</Output>
    </RomTrainer>
    <!--RomTrainer name="train_ROMperceptron">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMperceptron</Output>
    </RomTrainer-->
    <RomTrainer name="train_ROMridge">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMridge</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMridgeCV">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMridgeCV</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMsgdRegressor">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMsgdRegressor</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMsvr">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMsvr</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMknnReg">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMknnReg</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMradiusNeighReg">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMradiusNeighReg</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMdecTreeReg">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMdecTreeReg</Output>
    </RomTrainer>
    <RomTrainer name="train_ROMextraTreeReg">
      <Input class="DataObjects" type="PointSet">trainROM_pointset</Input>
      <Output class="Models" type="ROM">ROMextraTreeReg</Output>
    </RomTrainer>

    <MultiRun name="test_ROMmsrA">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMmsrA</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMmsrA_output_pointset</Output>
```

```
      <Output class="OutStreams" type="Print">ROMmsrA</Output>
    </MultiRun>
    <MultiRun name="test_ROMmsrB">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMmsrB</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMmsrB_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMmsrB</Output>
    </MultiRun>
    <MultiRun name="test_ROMndInvDW">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMndInvDW</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMndInvDW_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMndInvDW</Output>
    </MultiRun>
    <MultiRun name="test_ROMbayesRidge">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMbayesRidge</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMbayesRidge_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMbayesRidge</Output>
    </MultiRun>
    <MultiRun name="test_ROMelasticNet">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMelasticNet</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMelasticNet_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMelasticNet</Output>
    </MultiRun>
    <MultiRun name="test_ROMelasticNetCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMelasticNetCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMelasticNetCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMelasticNetCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMlars">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlars</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlars_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlars</Output>
```

```
    </MultiRun>
    <MultiRun name="test_ROMlarsCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlarsCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlarsCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlarsCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMlasso">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlasso</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlasso_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlasso</Output>
    </MultiRun>
    <MultiRun name="test_ROMlassoCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlassoCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlassoCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlassoCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMlassoLars">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlassoLars</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlassoLars_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlassoLars</Output>
    </MultiRun>
    <MultiRun name="test_ROMlassoLarsCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlassoLarsCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlassoLarsCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlassoLarsCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMlassoLarsIC">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlassoLarsIC</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlassoLarsIC_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlassoLarsIC</Output>
    </MultiRun>
```

```
    <MultiRun name="test_ROMlinearReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMlinearReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMlinearReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMlinearReg</Output>
    </MultiRun>
    <MultiRun name="test_ROMmultiTaskLasso">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMmultiTaskLasso</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMmultiTaskLasso_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMmultiTaskLasso</Output>
    </MultiRun>
    <MultiRun name="test_ROMmultiTaskElasticNet">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMmultiTaskElasticNet</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMmultiTaskElasticNet_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMmultiTaskElasticNet</Output>
    </MultiRun>
    <MultiRun name="test_ROMomp">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMomp</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMomp_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMomp</Output>
    </MultiRun>
    <MultiRun name="test_ROMompCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMompCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMompCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMompCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMpassAggReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMpassAggReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMpassAggReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMpassAggReg</Output>
    </MultiRun>
    <!--MultiRun name="test_ROMperceptron">
```

```
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMperceptron</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMperceptron_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMperceptron</Output>
    </MultiRun-->
    <MultiRun name="test_ROMridge">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMridge</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMridge_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMridge</Output>
    </MultiRun>
    <MultiRun name="test_ROMridgeCV">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMridgeCV</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMridgeCV_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMridgeCV</Output>
    </MultiRun>
    <MultiRun name="test_ROMsgdRegressor">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMsgdRegressor</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMsgdRegressor_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMsgdRegressor</Output>
    </MultiRun>
    <MultiRun name="test_ROMsvr">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMsvr</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMsvr_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMsvr</Output>
    </MultiRun>
    <MultiRun name="test_ROMknnReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMknnReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMknnReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMknnReg</Output>
    </MultiRun>
    <MultiRun name="test_ROMradiusNeighReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
```

```
      <Model class="Models" type="ROM">ROMradiusNeighReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMradiusNeighReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMradiusNeighReg</Output>
    </MultiRun>
    <MultiRun name="test_ROMdecTreeReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMdecTreeReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMdecTreeReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMdecTreeReg</Output>
    </MultiRun>
    <MultiRun name="test_ROMextraTreeReg">
      <Input class="DataObjects" type="PointSet">testROM_pointset</Input>
      <Model class="Models" type="ROM">ROMextraTreeReg</Model>
      <Sampler class='Samplers' type =
'CustomSampler'>customSamplerFile</Sampler>
      <Output class="DataObjects"
type="PointSet">testROMextraTreeReg_output_pointset</Output>
      <Output class="OutStreams" type="Print">ROMextraTreeReg</Output>
    </MultiRun>
  </Steps>

  <DataObjects>
    <PointSet name="trainROM_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROM_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMmsrA_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT</Output>
    </PointSet>
    <PointSet name="testROMmsrB_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMndInvDW_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
```

```
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMbayesRidge_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMelasticNet_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMelasticNetCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlars_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlarsCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlasso_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlassoCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlassoLars_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlassoLarsCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
```

```
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlassoLarsIC_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMlinearReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMmultiTaskLasso_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMmultiTaskElasticNet_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMomp_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMompCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMpassAggReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMperceptron_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMridge_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
```

```
    </PointSet>
    <PointSet name="testROMridgeCV_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMsgdRegressor_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMsvr_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMknnReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMradiusNeighReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMdecTreeReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
    <PointSet name="testROMextraTreeReg_output_pointset">

<Input>hrrPDF,tempAmb,floorArea,ventPerVol,fireHeight,hrrTray4,hrrTray5,hrrTr
ay6,height,hrrAdj,hrrTray1,hrrTray2,hrrTray3</Input>
      <Output>maxULT,timeToMaxULT</Output>
    </PointSet>
  </DataObjects>

  <OutStreams>
    <Print name="ROMmsrA">
      <type>csv</type>
      <source>testROMmsrA_output_pointset</source>
    </Print>
    <Print name="ROMmsrB">
      <type>csv</type>
      <source>testROMmsrB_output_pointset</source>
    </Print>
    <Print name="ROMndInvDW">
```

```
      <type>csv</type>
      <source>testROMndInvDW_output_pointset</source>
    </Print>
    <Print name="ROMbayesRidge">
      <type>csv</type>
      <source>testROMbayesRidge_output_pointset</source>
    </Print>
    <Print name="ROMelasticNet">
      <type>csv</type>
      <source>testROMelasticNet_output_pointset</source>
    </Print>
    <Print name="ROMelasticNetCV">
      <type>csv</type>
      <source>testROMelasticNetCV_output_pointset</source>
    </Print>
    <Print name="ROMlars">
      <type>csv</type>
      <source>testROMlars_output_pointset</source>
    </Print>
    <Print name="ROMlarsCV">
      <type>csv</type>
      <source>testROMlarsCV_output_pointset</source>
    </Print>
    <Print name="ROMlasso">
      <type>csv</type>
      <source>testROMlasso_output_pointset</source>
    </Print>
    <Print name="ROMlassoCV">
      <type>csv</type>
      <source>testROMlassoCV_output_pointset</source>
    </Print>
    <Print name="ROMlassoLars">
      <type>csv</type>
      <source>testROMlassoLars_output_pointset</source>
    </Print>
    <Print name="ROMlassoLarsCV">
      <type>csv</type>
      <source>testROMlassoLarsCV_output_pointset</source>
    </Print>
    <Print name="ROMlassoLarsIC">
      <type>csv</type>
      <source>testROMlassoLarsIC_output_pointset</source>
    </Print>
    <Print name="ROMlinearReg">
      <type>csv</type>
      <source>testROMlinearReg_output_pointset</source>
    </Print>
    <Print name="ROMmultiTaskLasso">
      <type>csv</type>
      <source>testROMmultiTaskLasso_output_pointset</source>
    </Print>
    <Print name="ROMmultiTaskElasticNet">
      <type>csv</type>
      <source>testROMmultiTaskElasticNet_output_pointset</source>
    </Print>
```

```
    <Print name="ROMomp">
      <type>csv</type>
      <source>testROMomp_output_pointset</source>
    </Print>
    <Print name="ROMompCV">
      <type>csv</type>
      <source>testROMompCV_output_pointset</source>
    </Print>
    <Print name="ROMpassAggReg">
      <type>csv</type>
      <source>testROMpassAggReg_output_pointset</source>
    </Print>
    <Print name="ROMperceptron">
      <type>csv</type>
      <source>testROMperceptron_output_pointset</source>
    </Print>
    <Print name="ROMridge">
      <type>csv</type>
      <source>testROMridge_output_pointset</source>
    </Print>
    <Print name="ROMridgeCV">
      <type>csv</type>
      <source>testROMridgeCV_output_pointset</source>
    </Print>
    <Print name="ROMsgdRegressor">
      <type>csv</type>
      <source>testROMsgdRegressor_output_pointset</source>
    </Print>
    <Print name="ROMsvr">
      <type>csv</type>
      <source>testROMsvr_output_pointset</source>
    </Print>
    <Print name="ROMknnReg">
      <type>csv</type>
      <source>testROMknnReg_output_pointset</source>
    </Print>
    <Print name="ROMradiusNeighReg">
      <type>csv</type>
      <source>testROMradiusNeighReg_output_pointset</source>
    </Print>
    <Print name="ROMdecTreeReg">
      <type>csv</type>
      <source>testROMdecTreeReg_output_pointset</source>
    </Print>
    <Print name="ROMextraTreeReg">
      <type>csv</type>
      <source>testROMextraTreeReg_output_pointset</source>
    </Print>
  </OutStreams>
</Simulation>
```

## A.10 MATLAB PLOTTING OF INITIAL ROM TESTING

The following MATLAB script generates plots of observed versus predicted for the initial reduced order model training and testing.

```
%% Initialize
clc; clear all; close all;
cd
('C:\msys64\home\worrelcl\raven\clarence\CFAST\ROM\ROM_Post_Process\InitialTe
sting(of26)');
files = dir('ROM*.csv');
set(0,'DefaultTextFontname', 'Times New Roman');
set(0,'DefaultAxesFontname','Times New Roman');
set(0,'defaultfigurecolor',[1 1 1]);

% Read and consolidate all predictions into two tables (predMaxULT and
% predTimetoMaxULT)
predMaxULT = readtable('_testData.csv');
predMaxULT = (predMaxULT(:,'maxULT'));
predMaxULT.Properties.VariableNames{'maxULT'}='TestData';
for n = 1:numel(files)
    temp = readtable(files(n).name);
    if ismember('maxULT',temp.Properties.VariableNames)==1
        ROM=strread(files(n).name,'%s','delimiter','.'); ROM=ROM(1);
        temp.Properties.VariableNames{'maxULT'} = ROM{1};
        predMaxULT = horzcat(predMaxULT,temp(:,ROM{1}));
    end
end
predTimeToMaxULT = readtable('_testData.csv');
predTimeToMaxULT = (predTimeToMaxULT(:,'timeToMaxULT'));
predTimeToMaxULT.Properties.VariableNames{'timeToMaxULT'}='TestData';
for n = 1:numel(files)
    temp = readtable(files(n).name);
    if ismember('timeToMaxULT',temp.Properties.VariableNames)==1
        ROM=strread(files(n).name,'%s','delimiter','.'); ROM=ROM(1);
        temp.Properties.VariableNames{'timeToMaxULT'} = ROM{1};
        predTimeToMaxULT = horzcat(predTimeToMaxULT,temp(:,ROM{1}));
    end
end
MTXpredMaxULT=table2array(predMaxULT);
MTXpredTimeToMaxULT=table2array(predTimeToMaxULT);
clear temp ROM n files

%% Plots of LINEAR ROMs Actual vs. Predicted
% these are grouped for consistency with thesis documentatoin

fig=figure
```

```
subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,4));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Elastic Net','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,4));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Elastic Net','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,5));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Elastic Net (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,5));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Elastic Net (CV)','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,17));
```

```
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Multi-Task Elastic Net','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,17));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Multi-Task Elastic Net','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,10));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,10));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig= figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,11));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
```

```
title('Lasso (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,11));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso (CV)','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,18));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Multi-Task Lasso','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,18));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Multi-Task Lasso','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,12));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
```

```
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,12));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,13));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,13));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR (CV)','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,14));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR using BIC / AIC','fontweight','bold', 'FontSize',
14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off
```

```
subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,14));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Lasso fit with LAR using BIC / AIC','fontweight','bold', 'FontSize',
14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,8));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Least Angle','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,8));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Least Angle','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,9));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Least Angle (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
```

```
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,9));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Least Angle (CV)','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,16));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('MSR','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,16));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('MSR','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,15));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ordinary Linear Regression','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,15));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
```

```
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ordinary Linear Regression','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,20));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Orthogonal Matching Pursuit','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,20));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Orthogonal Matching Pursuit','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,21));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Orthogonal Matching Pursuit (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,21));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Orthogonal Matching Pursuit (CV)','fontweight','bold', 'FontSize', 14)
```

```
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,22));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Passive Aggressive Regression','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,22));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Passive Aggressive Regression','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,24));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ridge','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,24));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ridge','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
```

```
hold off

subplot(2,2,3)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,25));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ridge (CV)','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,25));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Ridge (CV)','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,2));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Bayesian Ridge','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,2));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Bayesian Ridge','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
```

```
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,26));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('SGD Regression','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,26));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('SGD Regression','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

%% Plots of TREE-BASED ROMs Actual vs. Predicted
% these are grouped for consistency with thesis documentatoin

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,3));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Decision Tree','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,3));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Decision Tree','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
```

```
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,6));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Extra Tree','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,6));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Extra Tree','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

%% Thesis Plots of NEIGHBOR-BASED ROMs Actual vs. Predicted
% these are grouped for consistency with thesis documentatoin

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,7));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('K-Nearest Neighbor','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,7));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('K-Nearest Neighbor','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,3)
```

```
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,23));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Radius-Based Neighbor','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,4)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,23));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Radius-Based Neighbor','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,19));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Inverse Distance Weight','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,19));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Inverse Distance Weight','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

%% Plots of SUPPORT VECTOR MACHINE ROMs Actual vs. Predicted
% these are grouped for consistency with thesis documentatoin
```

```
fig=figure

subplot(2,2,1)
hold on
scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,27));
xlabel('CFAST Max. Upper Layer Temp. (\circC)','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Support Vector Machine','fontweight','bold', 'FontSize', 14)
axis([0,350,0,350]);
refline(1,0);
set(gca,'box','on')
hold off

subplot(2,2,2)
hold on
scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,27));
xlabel('CFAST Time (min.) to Max. Upper Layer Temp.','fontweight','bold',
'FontSize', 12)
ylabel('Predicted','fontweight','bold', 'FontSize', 12)
title('Support Vector Machine','fontweight','bold', 'FontSize', 14)
axis([0,60,0,60]);
refline(1,0);
set(gca,'box','on')
hold off

print(fig,'-dpng')

%% Plot ROM Actual vs. Predicted for Max ULT
% % this section plots all ROMs in their raw order from RAVEN
% ROMnamesMax=predMaxULT.Properties.VariableNames;
% ROMnamesMax=ROMnamesMax(2:length(ROMnamesMax));
% for n=1:length(ROMnamesMax)
%     switch ROMnamesMax{n}
%         case 'ROMbayesRidge'
%             ROMnamesMax{n}='Bayesian Ridge';
%         case 'ROMdecTreeReg'
%             ROMnamesMax{n}='Decision Tree';
%         case 'ROMelasticNet'
%             ROMnamesMax{n}='Elastic Net';
%         case 'ROMelasticNetCV'
%             ROMnamesMax{n}='Elastic Net (CV)';
%         case 'ROMextraTreeReg'
%             ROMnamesMax{n}='Extra Tree';
%         case 'ROMknnReg'
%             ROMnamesMax{n}='K-Nearest Neighbors Regressor';
%         case 'ROMlars'
%             ROMnamesMax{n}='Least Angle';
%         case 'ROMlarsCV'
%             ROMnamesMax{n}='Least Angle (CV)';
%         case 'ROMlasso'
%             ROMnamesMax{n}='Lasso';
%         case 'ROMlassoCV'
%             ROMnamesMax{n}='Lasso (CV)';
```

```
%          case 'ROMlassoLars'
%              ROMnamesMax{n}='Lasso fit with LAR';
%          case 'ROMlassoLarsCV'
%              ROMnamesMax{n}='Lasso fit with LAR (CV)';
%          case 'ROMlassoLarsIC'
%              ROMnamesMax{n}='Lasso fit with LAR using BIC or AIC';
%          case 'ROMlinearReg'
%              ROMnamesMax{n}='Ordinary Linear Regression';
%          case 'ROMmsrA'
%              ROMnamesMax{n}='MSR';
%          case 'ROMmultiTaskElasticNet'
%              ROMnamesMax{n}='Multi-Task Elastic Net';
%          case 'ROMmultiTaskLasso'
%              ROMnamesMax{n}='Multi-Task Lasso';
%          case 'ROMndInvDW'
%              ROMnamesMax{n}='Inverse Distance Weighting';
%          case 'ROMomp'
%              ROMnamesMax{n}='Orthogonal Mathching Pursuit';
%          case 'ROMompCV'
%              ROMnamesMax{n}='Orthogonal Mathching Pursuit (CV)';
%          case 'ROMpassAggReg'
%              ROMnamesMax{n}='Passive Aggressive Regression';
%          case 'ROMradiusNeighReg'
%              ROMnamesMax{n}='Radius Based Neighbor';
%          case 'ROMridge'
%              ROMnamesMax{n}='Ridge';
%          case 'ROMridgeCV'
%              ROMnamesMax{n}='Ridge (CV)';
%          case 'ROMsgdRegressor'
%              ROMnamesMax{n}='SGD Regression';
%          case 'ROMsvr'
%              ROMnamesMax{n}='Support Vector Regression';
%      end
% end
%
% for n=1:6
%      figure
%      subplot(2,2,1)
%      hold on
%      scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,2+4*(n-1)));
%      xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%      ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%      title(ROMnamesMax{1+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%      axis([0,350,0,350]);
%      refline(1,0);
%      set(gca,'box','on')
%      hold off
%
%      subplot(2,2,2)
%      hold on
%      scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,3+4*(n-1)));
%      xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%      ylabel('Predicted','fontweight','bold', 'FontSize', 12)
```

```
%     title(ROMnamesMax{2+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,350,0,350]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,3)
%     hold on
%     scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,4+4*(n-1)));
%     xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesMax{3+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,350,0,350]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,4)
%     hold on
%     scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,5+4*(n-1)));
%     xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesMax{4+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,350,0,350]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     clear n ROM
% end
%
%     figure
%     subplot(2,2,1)
%     hold on
%     scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,26));
%     xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesMax{25},'fontweight','bold', 'FontSize', 14)
%     axis([0,350,0,350]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,2)
%     hold on
%     scatter(MTXpredMaxULT(:,1),MTXpredMaxULT(:,27));
%     xlabel('CFAST Max Upper Layer Temperature (C)','fontweight','bold',
'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesMax{26},'fontweight','bold', 'FontSize', 14)
%     axis([0,350,0,350]);
%     refline(1,0);
```

```
%     set(gca,'box','on')
%     hold off
%
% %% Plot ROM Actual vs. Predicted for Time to Max ULT
% ROMnamesTime=predTimeToMaxULT.Properties.VariableNames;
% ROMnamesTime=ROMnamesTime(2:length(ROMnamesTime));
% for n=1:length(ROMnamesTime)
%     switch ROMnamesTime{n}
%         case 'ROMbayesRidge'
%             ROMnamesTime{n}='Bayesian Ridge';
%         case 'ROMdecTreeReg'
%             ROMnamesTime{n}='Decision Tree';
%         case 'ROMelasticNet'
%             ROMnamesTime{n}='Elastic Net';
%         case 'ROMelasticNetCV'
%             ROMnamesTime{n}='Elastic Net (CV)';
%         case 'ROMextraTreeReg'
%             ROMnamesTime{n}='Extra Tree';
%         case 'ROMknnReg'
%             ROMnamesTime{n}='K-Nearest Neighbors Regressor';
%         case 'ROMlars'
%             ROMnamesTime{n}='Least Angle';
%         case 'ROMlarsCV'
%             ROMnamesTime{n}='Least Angle (CV)';
%         case 'ROMlasso'
%             ROMnamesTime{n}='Lasso';
%         case 'ROMlassoCV'
%             ROMnamesTime{n}='Lasso (CV)';
%         case 'ROMlassoLars'
%             ROMnamesTime{n}='Lasso fit with LAR';
%         case 'ROMlassoLarsCV'
%             ROMnamesTime{n}='Lasso fit with LAR (CV)';
%         case 'ROMlassoLarsIC'
%             ROMnamesTime{n}='Lasso fit with LAR using BIC or AIC';
%         case 'ROMlinearReg'
%             ROMnamesTime{n}='Ordinary Linear Regression';
%         case 'ROMmsrB'
%             ROMnamesTime{n}='MSR';
%         case 'ROMmultiTaskElasticNet'
%             ROMnamesTime{n}='Multi-Task Elastic Net';
%         case 'ROMmultiTaskLasso'
%             ROMnamesTime{n}='Multi-Task Lasso';
%         case 'ROMndInvDW'
%             ROMnamesTime{n}='Inverse Distance Weighting';
%         case 'ROMomp'
%             ROMnamesTime{n}='Orthogonal Mathching Pursuit';
%         case 'ROMompCV'
%             ROMnamesTime{n}='Orthogonal Mathching Pursuit (CV)';
%         case 'ROMpassAggReg'
%             ROMnamesTime{n}='Passive Aggressive Regression';
%         case 'ROMradiusNeighReg'
%             ROMnamesTime{n}='Radius Based Neighbor';
%         case 'ROMridge'
%             ROMnamesTime{n}='Ridge';
%         case 'ROMridgeCV'
```

```
%             ROMnamesTime{n}='Ridge (CV)';
%         case 'ROMsgdRegressor'
%             ROMnamesTime{n}='SGD Regression';
%         case 'ROMsvr'
%             ROMnamesTime{n}='Support Vector Regression';
%     end
% end
%
% for n=1:6
%     figure
%     subplot(2,2,1)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,2+4*(n-1)));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{1+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,2)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,3+4*(n-1)));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{2+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,3)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,4+4*(n-1)));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{3+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,4)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,5+4*(n-1)));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{4+4*(n-1)},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
```

```
%     set(gca,'box','on')
%     hold off
%
%     clear n ROM
% end
%
%     figure
%     subplot(2,2,1)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,26));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{25},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
%
%     subplot(2,2,2)
%     hold on
%     scatter(MTXpredTimeToMaxULT(:,1),MTXpredTimeToMaxULT(:,27));
%     xlabel('CFAST Time to Max Upper Layer Temperature
(minutes)','fontweight','bold', 'FontSize', 12)
%     ylabel('Predicted','fontweight','bold', 'FontSize', 12)
%     title(ROMnamesTime{26},'fontweight','bold', 'FontSize', 14)
%     axis([0,60,0,60]);
%     refline(1,0);
%     set(gca,'box','on')
%     hold off
```

## A.11 FINAL MODEL TUNING

The following 'R' script performs the final model tuning and results plotting for selected metamodel types.

```
###Load Libraries
```{r eval=TRUE, message=FALSE, warning=FALSE}
library('AppliedPredictiveModeling')
library('caret')
library('corrplot')
library('dplyr')
library('ggplot2')
library('Lahman')
library('nnet')
library('partykit')
library('rpart')
library('stats')
library('tidyr')
```

###Inport Data, Drop Redundant Parameters:
```{r eval=TRUE}
trainData<-read.table("_trainData.csv",header=TRUE,sep=",",dec=".")
testData<-read.table("_testData.csv",header=TRUE,sep=",",dec=".")

drops <- c("hrrIndex","hrr","fireHeightFract","width","length")
trainData<-trainData[ , !(names(trainData) %in% drops)]
testData<-testData[ , !(names(testData) %in% drops)]
rm(drops)

trainMaxTempData<-as.data.frame(c(trainData[1:14],trainData[16]))
testMaxTempData<-as.data.frame(c(testData[1:14],testData[16]))

trainTimeToMaxData<-
as.data.frame(c(trainData[1:13],trainData[15],trainData[16]))
testTimeToMaxData<-as.data.frame(c(testData[1:13],testData[15],testData[16]))
```

###Visualization
```{r eval=TRUE}
corrplot(cor(trainData),order="hclust",tl.cex = 1.0,title="Correlation
between all Parameters",mar = c(0,0,2,0))
```

###Decision Tree (Max ULT)
```{r, message=FALSE, warning=FALSE}
treeControl = rpart.control(minsplit = 1, minbucket = 1,maxdepth = 30,cp =
0.001)
```
```

```
rpartTree<-rpart(maxULT
~.,data=trainMaxTempData,method="anova",control=treeControl)

printcp(rpartTree)
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(rpartTree)

selectedFit<-prune(rpartTree,0.002)
printcp(selectedFit)
plot(as.party(selectedFit))

predict1<-data.frame(testMaxTempData[1:13])
predict1$predicted<-(predict(selectedFit,testMaxTempData))
predict1$observed<-data.frame(testMaxTempData$maxULT)

ggplot(data=predict1, aes(x=observed,y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Decision Tree")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))
rm(predict1)
```

```
###Decision Tree (Time to Max ULT)
```{r, message=FALSE, warning=FALSE}
treeControl = rpart.control(minsplit = 1, minbucket = 1,maxdepth = 30,cp =
0.001)
rpartTree<-rpart(timeToMaxULT
~.,data=trainTimeToMaxData,method="anova",control=treeControl)

printcp(rpartTree)
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(rpartTree)

selectedFit<-prune(rpartTree,0.002)
printcp(selectedFit)
plot(as.party(selectedFit))

predict1<-data.frame(testTimeToMaxData[1:13])
predict1$predicted<-(predict(selectedFit,testTimeToMaxData))
predict1$observed<-data.frame(testTimeToMaxData$timeToMaxULT)

ggplot(data=predict1, aes(x=observed,y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Decision Tree")+
  xlim(0,60)+
  ylim(0,60)+
  xlab("CFAST Time to Max Upper Layer Temp. (minutes)")+
```
```

```
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))
rm(predict1)
```

###K-Nearest Neighbors (Max ULT)
```{r eval=TRUE}
knn<-
train(x=trainMaxTempData[1:10000,1:13],y=trainMaxTempData[1:10000,14],method=
"knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=1:30),trControl=trai
nControl(method="cv"))
print(knn)
plot(knn)

selectedFit<-
train(x=trainMaxTempData[1:100000,1:13],y=trainMaxTempData[1:100000,14],metho
d="knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=train
Control(method="cv"))

predict1<-data.frame(testMaxTempData[1:13])
predict1$predicted<-(predict(selectedFit,testMaxTempData[,1:13]))
predict1$observed<-data.frame(testMaxTempData$maxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)

ggplot(data=predict1, aes(x=observed, y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("K-Nearest Neighbors")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=ratio))+
  geom_histogram()+
  ggtitle("Max. Upper Layer Temp.")+
  xlim(0.7,1.3)+
  xlab("Ratio of Predicted to Observed")+
  ylab("Count")+
  theme(axis.title = element_text(face="bold",size = 16))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 16))
```

###K-Nearest Neighbors (Time to Max ULT)
```{r eval=TRUE}
knn<-
train(x=trainTimeToMaxData[1:10000,1:13],y=trainTimeToMaxData[1:10000,14],met
hod="knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=1:30),trControl=
trainControl(method="cv"))
```

```
print(knn)
plot(knn)

selectedFit<-
train(x=trainTimeToMaxData[1:100000,1:13],y=trainTimeToMaxData[1:100000,14],m
ethod="knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=5),trControl=t
rainControl(method="cv"))

predict1<-data.frame(testTimeToMaxData[1:13])
predict1$predicted<-(predict(selectedFit,testTimeToMaxData[,1:13]))
predict1$observed<-data.frame(testTimeToMaxData$timeToMaxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)

ggplot(data=predict1, aes(x=observed,y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("K-Nearest Neighbors")+
  xlim(0,60)+
  ylim(0,60)+
  xlab("CFAST Time to Max Upper Layer Temp. (minutes)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=ratio))+
  geom_histogram()+
  ggtitle("Time to Max. Upper Layer Temp.")+
  xlim(0.7,1.3)+
  xlab("Ratio of Predicted to Observed")+
  ylab("Count")+
  theme(axis.title = element_text(face="bold",size = 16))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 16))

rm(predict1,knn,selectedFit)
```

```
###Support Vector Machine (Max ULT)
```{r eval=TRUE}
svmRadial<-train(x=trainMaxTempData[1:5000,1:13],
y=trainMaxTempData[1:5000,14], method="svmRadial",
preProc=c("center","scale"), tuneLength=14,
trControl=trainControl(method="cv"))

svmRadial$finalModel
print(svmRadial)
plot(svmRadial)

selectedFit<-svmRadial

predict1<-data.frame(testMaxTempData[1:13])
predict1$predicted<-(predict(selectedFit,testMaxTempData[,1:13]))
predict1$observed<-data.frame(testMaxTempData$maxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)
```
```

```
ggplot(data=predict1, aes(x=observed, y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Support Vector Machine (Radial Basis Kernal)")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

svmLinear<-train(x=trainMaxTempData[1:5000,1:13],
y=trainMaxTempData[1:5000,14], method="svmLinear",
preProc=c("center","scale"), tuneLength=14,
trControl=trainControl(method="cv"))

svmLinear$finalModel
print(svmLinear)
#plot(svmLinear)

selectedFit<-svmLinear

predict1<-data.frame(testMaxTempData[1:13])
predict1$predicted<-(predict(selectedFit,testMaxTempData[,1:13]))
predict1$observed<-data.frame(testMaxTempData$maxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)

ggplot(data=predict1, aes(x=observed, y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Support Vector Machine (Linear Kernal)")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

rm(predict1,selectedFit)
```

```
###Support Vector Machine (Time to Max ULT)
```{r eval=TRUE}
svmRadial<-train(x=trainTimeToMaxData[1:5000,1:13],
y=trainTimeToMaxData[1:5000,14], method="svmRadial",
preProc=c("center","scale"), tuneLength=14,
trControl=trainControl(method="cv"))

svmRadial$finalModel
print(svmRadial)
plot(svmRadial)
```
```

```
selectedFit<-svmRadial

predict1<-data.frame(testTimeToMaxData[1:13])
predict1$predicted<-(predict(selectedFit,testTimeToMaxData[,1:13]))
predict1$observed<-data.frame(testTimeToMaxData$timeToMaxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)

ggplot(data=predict1, aes(x=observed, y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Support Vector Machine (Radial Basis Kernal)")+
  xlim(0,60)+
  ylim(0,60)+
  xlab("CFAST Time (min.) to Max. Upper Layer Temp.")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

svmLinear<-train(x=trainTimeToMaxData[1:5000,1:13],
y=trainTimeToMaxData[1:5000,14], method="svmLinear",
preProc=c("center","scale"), tuneLength=14,
trControl=trainControl(method="cv"))

svmLinear$finalModel
print(svmLinear)
#plot(svmLinear)

selectedFit<-svmLinear

predict1<-data.frame(testTimeToMaxData[1:13])
predict1$predicted<-(predict(selectedFit,testTimeToMaxData[,1:13]))
predict1$observed<-data.frame(testTimeToMaxData$timeToMaxULT)
predict1$ratio<-(predict1$predicted)/(predict1$observed)

ggplot(data=predict1, aes(x=observed, y=predicted))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Support Vector Machine (Linear Kernal)")+
  xlim(0,60)+
  ylim(0,60)+
  xlab("CFAST Time (min.) to Max. Upper Layer Temp.")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

rm(predict1,selectedFit)
```

```
###Comparision of KNN against MQH for Max ULT
```{r eval=TRUE}
#Fit model and build KNN data frame
testData<-read.table("_testData.csv",header=TRUE,sep=",",dec=".") #reload
with all parameters
```
```

```
knnFit<-
train(x=trainMaxTempData[1:100000,1:13],y=trainMaxTempData[1:100000,14],metho
d="knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=train
Control(method="cv"))

predictKNN<-data.frame(testMaxTempData[1:13])
predictKNN$length=testData["length"]
predictKNN$width=testData["width"]
predictKNN$fireObject<-testMaxTempData["fireObject"]
predictKNN$predicted<-(predict(knnFit,testMaxTempData[,1:13]))
predictKNN$observed<-(testMaxTempData$maxULT)
predictKNN$ratio<-(predictKNN$predicted)/(predictKNN$observed)

#Add to predictKNN dataframe the parameters Qmax and Wall Area needed for MQH
maxHRR<-read.table("MaxHRR.csv",header=TRUE,sep=",",dec=".")
predictKNN$maxHRR<-0
for (i in 1:135000) {
  predictKNN[i,"maxHRR"]=maxHRR[paste("X",predictKNN[i,"fireObject"],sep="")]
  }
predictKNN["AT"]=(2*predictKNN["floorArea"])+(2*predictKNN["length"]*predictK
NN["height"])+(2*predictKNN["width"]*predictKNN["height"]-1.2)

#MQH Calculation
predictKNN$MQH=6.85*((predictKNN$maxHRR^2)/(0.2*sqrt(0.0125)*0.0464*predictKN
N$AT))^(1/3)

#FPA Calculation
predictKNN$ventKGS = predictKNN$ventPerVol * predictKNN$floorArea *
predictKNN$height * 1.18
predictKNN$FPA = predictKNN$tempAmb *
((predictKNN$maxHRR/(predictKNN$ventKGS*1.01*predictKNN$tempAmb))^0.72) *
(((0.0464*predictKNN$AT)/(predictKNN$ventKGS*1.01))^-0.36)

ggplot()+
  geom_point(data=predictKNN, aes(x=observed, y=MQH), color="red",
alpha=0.01)+
  geom_point(data=predictKNN, aes(x=observed, y=FPA), color="blue",
alpha=0.01)+
  geom_point(data=predictKNN, aes(x=observed, y=predicted), color="green",
alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Comparsison of CFAST to Alternate Models")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (˚C)")+
  ylab("Alternate Model \n Max Upper Layer Temp. (˚C)")+
  theme(axis.title = element_text(face="bold",size = 17))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))+
  annotate("text",x=60,y=235,label="MQH",fontface=2,size=5)+
  annotate("text",x=125,y=190,label="FPA",fontface=2,size=5)+
  annotate("text",x=175,y=150,label="KNN",fontface=2,size=5)
```

```
###Accuracy-Efficiency
```

```
```{r eval=TRUE}
fit10<-
train(x=trainMaxTempData[1:10,1:13],y=trainMaxTempData[1:10,14],method="knn",
preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainControl(
method="cv"))

fit100<-
train(x=trainMaxTempData[1:100,1:13],y=trainMaxTempData[1:100,14],method="knn
",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainContro
l(method="cv"))

fit500<-
train(x=trainMaxTempData[1:500,1:13],y=trainMaxTempData[1:500,14],method="knn
",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainContro
l(method="cv"))

fit1000<-
train(x=trainMaxTempData[1:1000,1:13],y=trainMaxTempData[1:1000,14],method="k
nn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainCont
rol(method="cv"))

fit5000<-
train(x=trainMaxTempData[1:5000,1:13],y=trainMaxTempData[1:5000,14],method="k
nn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainCont
rol(method="cv"))

fit10000<-
train(x=trainMaxTempData[1:10000,1:13],y=trainMaxTempData[1:10000,14],method=
"knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainCo
ntrol(method="cv"))

fit50000<-
train(x=trainMaxTempData[1:50000,1:13],y=trainMaxTempData[1:50000,14],method=
"knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=trainCo
ntrol(method="cv"))

fit100000<-
train(x=trainMaxTempData[1:100000,1:13],y=trainMaxTempData[1:100000,14],metho
d="knn",preProc=c("center","scale"),tuneGrid=data.frame(.k=4),trControl=train
Control(method="cv"))

predict1<-data.frame(testMaxTempData[1:13])
predict1$observed<-data.frame(testMaxTempData$maxULT)
predict1$predicted10<-(predict(fit10,testMaxTempData[,1:13]))
predict1$predicted100<-(predict(fit100,testMaxTempData[,1:13]))
predict1$predicted500<-(predict(fit500,testMaxTempData[,1:13]))
predict1$predicted1000<-(predict(fit1000,testMaxTempData[,1:13]))
predict1$predicted5000<-(predict(fit5000,testMaxTempData[,1:13]))
predict1$predicted10000<-(predict(fit10000,testMaxTempData[,1:13]))
predict1$predicted50000<-(predict(fit50000,testMaxTempData[,1:13]))
predict1$predicted100000<-(predict(fit100000,testMaxTempData[,1:13]))

fit<-data.frame(1:8)
fit$RMSE=c(fit10$results$RMSE,fit100$results$RMSE,fit500$results$RMSE,fit1000
$results$RMSE,fit5000$results$RMSE,fit10000$results$RMSE,fit50000$results$RMS
```
```

```
E,fit100000$results$RMSE)
fit$numSamples=c(10,100,500,1000,5000,10000,50000,100000)

ggplot(data=fit, aes(x=numSamples, y=RMSE))+
  geom_point()+
  geom_line()+
  ggtitle("Model Accuracy vs. Training Sample Size")+
  xlab("Training Sample Size")+
  ylab("RMSE")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted10))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("10 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted100))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("100 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted500))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("500 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted1000))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("1,000 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
```

```
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted5000))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("5,000 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted10000))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("10,000 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted50000))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("50,000 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

ggplot(data=predict1, aes(x=observed, y=predicted100000))+
  geom_point(alpha=0.01)+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("100,000 Training Points")+
  xlim(0,350)+
  ylim(0,350)+
  xlab("CFAST Max Upper Layer Temp. (C)")+
  ylab("Predicted")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))

```

```
###Artificial Neural Network (Max ULT)
```{r, message=FALSE, warning=FALSE}
preProcData <- preProcess(trainData, method = c("center", "scale"))
preProcData <-  predict(preProcData, trainData)

set.seed(2064)
inTrain<-createDataPartition(y=preProcData$maxULT,p=0.80,list=FALSE)
training <- preProcData[inTrain,]
testing <- preProcData[-inTrain,]
remove(inTrain)

model1<-nnet(x = training[,1:18], y=training[,19],size=2,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model2<-nnet(x = training[,1:18], y=training[,19],size=3,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model3<-nnet(x = training[,1:18], y=training[,19],size=5,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model4<-nnet(x = training[,1:18], y=training[,19],size=7,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model5<-nnet(x = training[,1:18], y=training[,19],size=10,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)

predict1<-data.frame(testing[,1:18])
predict2<-data.frame(testing[,1:18])
predict3<-data.frame(testing[,1:18])
predict4<-data.frame(testing[,1:18])
predict5<-data.frame(testing[,1:18])

predict1$predicted<-(predict(model1,predict1))
predict2$predicted<-(predict(model2,predict2))
predict3$predicted<-(predict(model3,predict3))
predict4$predicted<-(predict(model4,predict4))
predict5$predicted<-(predict(model5,predict5))

predict1$observed<-data.frame(testing[19])
predict2$observed<-data.frame(testing[19])
predict3$observed<-data.frame(testing[19])
predict4$observed<-data.frame(testing[19])
predict5$observed<-data.frame(testing[19])

ObsFit1<-data.frame(predict1$observed,predict1$predicted)
names(ObsFit1) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit1)
RMSE1=round(modelstats[1],3)
Rsq1=round(modelstats[2],3)

ObsFit2<-data.frame(predict2$observed,predict2$predicted)
names(ObsFit2) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit2)
RMSE2=round(modelstats[1],3)
Rsq2=round(modelstats[2],3)

ObsFit3<-data.frame(predict3$observed,predict3$predicted)
names(ObsFit3) <- c("obs", "pred")
```
```

```
modelstats=defaultSummary(ObsFit3)
RMSE3=round(modelstats[1],3)
Rsq3=round(modelstats[2],3)

ObsFit4<-data.frame(predict4$observed,predict4$predicted)
names(ObsFit4) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit4)
RMSE4=round(modelstats[1],3)
Rsq4=round(modelstats[2],3)

ObsFit5<-data.frame(predict4$observed,predict5$predicted)
names(ObsFit5) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit5)
RMSE5=round(modelstats[1],3)
Rsq5=round(modelstats[2],3)

p1=ggplot(data=ObsFit1, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (2 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE1),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq1),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p2=ggplot(data=ObsFit2, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (3 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE2),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq2),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p3=ggplot(data=ObsFit3, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (5 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
```

```
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE3),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq3),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p4=ggplot(data=ObsFit5, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (10 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE5),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq5),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

multiplot(p1,p3,p2,p4,cols=2)
R=c(0,Rsq1,Rsq2,Rsq3,Rsq4,Rsq5)
L=c(0,1,3,5,7,10)
nnR2=data.frame(L,R)
ggplot(data=nnR2, aes(x=L,y=R))+
  geom_line(size=1.5)+
  ggtitle("Neural Net R^2 vs. Depth")+
  xlab("Number of Layers")+
  ylab("R^2")+
  ylim(0,1)+
  scale_x_continuous(breaks=c(0,1,2,3,4,5,6,7,8,9,10))+
  annotate("text", x = 4, y = 0.50, label = "Five Layers Appears
Optimal",size=6,colour="blue")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold",size = 5))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))
```

```
###Artificial Neural Network (Time to Max ULT)
```{r, message=FALSE, warning=FALSE}
preProcData <- preProcess(trainData, method = c("center", "scale"))
preProcData <-  predict(preProcData, trainData)

set.seed(2064)
inTrain<-createDataPartition(y=preProcData$timeToMaxULT,p=0.80,list=FALSE)
training <- preProcData[inTrain,]
testing <- preProcData[-inTrain,]
remove(inTrain)

model1<-nnet(x = training[,1:18], y=training[,20],size=2,decay=0.01,
```
```

```
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model2<-nnet(x = training[,1:18], y=training[,20],size=3,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model3<-nnet(x = training[,1:18], y=training[,20],size=5,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model4<-nnet(x = training[,1:18], y=training[,20],size=7,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)
model5<-nnet(x = training[,1:18], y=training[,20],size=10,decay=0.01,
maxit=1000, MaxNWts=100000,linout=T,trace=FALSE)

predict1<-data.frame(testing[,1:18])
predict2<-data.frame(testing[,1:18])
predict3<-data.frame(testing[,1:18])
predict4<-data.frame(testing[,1:18])
predict5<-data.frame(testing[,1:18])

predict1$predicted<-(predict(model1,predict1))
predict2$predicted<-(predict(model2,predict2))
predict3$predicted<-(predict(model3,predict3))
predict4$predicted<-(predict(model4,predict4))
predict5$predicted<-(predict(model5,predict5))

predict1$observed<-data.frame(testing[20])
predict2$observed<-data.frame(testing[20])
predict3$observed<-data.frame(testing[20])
predict4$observed<-data.frame(testing[20])
predict5$observed<-data.frame(testing[20])

ObsFit1<-data.frame(predict1$observed,predict1$predicted)
names(ObsFit1) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit1)
RMSE1=round(modelstats[1],3)
Rsq1=round(modelstats[2],3)

ObsFit2<-data.frame(predict2$observed,predict2$predicted)
names(ObsFit2) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit2)
RMSE2=round(modelstats[1],3)
Rsq2=round(modelstats[2],3)

ObsFit3<-data.frame(predict3$observed,predict3$predicted)
names(ObsFit3) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit3)
RMSE3=round(modelstats[1],3)
Rsq3=round(modelstats[2],3)

ObsFit4<-data.frame(predict4$observed,predict4$predicted)
names(ObsFit4) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit4)
RMSE4=round(modelstats[1],3)
Rsq4=round(modelstats[2],3)

ObsFit5<-data.frame(predict4$observed,predict5$predicted)
names(ObsFit5) <- c("obs", "pred")
modelstats=defaultSummary(ObsFit5)
```

```
RMSE5=round(modelstats[1],3)
Rsq5=round(modelstats[2],3)

p1=ggplot(data=ObsFit1, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (2 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE1),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq1),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p2=ggplot(data=ObsFit2, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (3 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE2),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq2),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p3=ggplot(data=ObsFit3, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (5 Layers)")+
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE3),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq3),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

p4=ggplot(data=ObsFit5, aes(x=obs,y=pred))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1)+
  ggtitle("Neural Net (10 Layers)")+
```

```
  xlab("Observed")+
  ylab("Predicted")+
  xlim(-1,4)+
  ylim(-1,4)+
  annotate("text", x = 0, y = 3, label = paste("RMSE =
",RMSE5),size=3,colour="red")+
  annotate("text", x = 0, y = 2.5, label = paste("R^2
=",Rsq5),size=3,colour="red")+
  theme(axis.title = element_text(face="bold"))+
  theme(axis.text = element_text(face="bold"))+
  theme(plot.title = element_text(hjust = 0.5,face="bold"))

multiplot(p1,p3,p2,p4,cols=2)
R=c(0,Rsq1,Rsq2,Rsq3,Rsq4,Rsq5)
L=c(0,1,3,5,7,10)
nnR2=data.frame(L,R)
ggplot(data=nnR2, aes(x=L,y=R))+
  geom_line(size=1.5)+
  ggtitle("Neural Net R^2 vs. Depth")+
  xlab("Number of Layers")+
  ylab("R^2")+
  ylim(0,1)+
  scale_x_continuous(breaks=c(0,1,2,3,4,5,6,7,8,9,10))+
  annotate("text", x = 4, y = 0.50, label = "Five Layers Appears
Optimal",size=6,colour="blue")+
  theme(axis.title = element_text(face="bold",size = 18))+
  theme(axis.text = element_text(face="bold",size = 5))+
  theme(plot.title = element_text(hjust = 0.5,face="bold",size = 20))
```

## BIBLIOGRAPHY


Alfonsi, A., Cristian, R., Manselli, D., Cogliati, J., Kinoshita, R., & Naviglio, A. (2014). RAVEN and Dynamic Probabilistic Risk Assessment: Software Review (INL/CON-14-31785). In *ESREL 2014 European Safety and Reliability Conference*. European Safety and Reliability Association (ESRA).

Alfonsi, A., Rabiti, C., Cogliati, J., Mandelli, D., Sen, S., Kinoshita, R., … Smith, C. (2014). *Light Water Reactor Sustainability Program Dynamic Event Tree Advancements and Control Logic Improvements (INL/EXT-15-36758)*. Idaho Falls, ID.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J. J., & Kinoshita, R. A. (2013). Raven As a Tool for Dynamic Probabilistic Risk Assessment : Software Overview. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*. American Nuclear Society.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J. J., Kinoshita, R. A., & Naviglio, A. (2013). Dynamic Event Tree Analysis through RAVEN. In *International Topical Meeting on Probabilistic Safety Assessment and Analysis, ANS PSA 2013*. American Nuclear Society.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., & Kinoshita, R. (2013a). Performing Probabilistic Risk Assessment Through RAVEN (INL/CON-13-29472). In *ANS Annual Meeting*. American Nuclear Society.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., & Kinoshita, R. (2013b). *RAVEN : Dynamic Event Tree Approach Level III Milestone RAVEN (INL/EXT-13-30332)*. Idaho Falls, ID.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., & Kinoshita, R. (2014a). Adaptive Dynamic Event Tree in RAVEN Code (INL/CON-32595). In *ANS Winter Meeting*. American Nuclear Society.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., & Kinoshita, R. (2014b). *RAVEN: Development of the Adaptive Dynamic Event Tree Approach (INL/MIS-14-33246 Rev. 1)*. Idaho Falls, ID.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., Sen, S., & Smith, C. L. (2015). *Light Water Reactor Sustainability Program Improving Limit Surface Search Algorithms in RAVEN Using Acceleration Schemes (INL/EXT-15-36100)*. Idaho Falls, ID.

Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., Wang, C., Talbot, P. W., … Smith, C. (2017). *RAVEN Theory Manual and User Guide (INL/EXT-16-38178)*. Idaho Falls, ID.

American Society for Testing Materials International. (2012). *Standard Guide for Evaluating the Predictive Capability of Deterministic Fire Models (ASTM E1355-12)*. West Conshohocken, PA. https://doi.org/10.1520/E1355-12.2

Barton, R. R. (2010). Simulation experimental design. In *Proceedings of the 2010 Winter Simulation Conference* (pp. 75–86).

Barton, R. R. (2015). Tutorial: Simulation Metamodeling. In *Proceedings of the 2015 Winter Simulation Conference* (pp. 1765–1779). https://doi.org/10.1109/WSC.2015.7408294

Brandyberry, M., & Apostolakis, G. (1990). Response Surface Approximation of a Fire Risk Analysis Computer Code. *Reliability Engineering and System Safety*, *29*, 153–184.

Cohn, B., Denning, R., Aldemir, T., Sezen, H., & Hur, J. (2016). Implementation of Surrogate Models within RAVEN to Support SPRA Uncertainty Quantification. In *Proceedings of PSA2017*. American Nuclear Society.

Duke Energy. (2013). *Duke Energy Carolinas, LLC (Duke Energy) McGuire Nuclear Station, Units 1 and 2 Docket Numbers 50-369 and 50-370 License Amendment Request (LAR) to Adopt National Fire Protection Association (NFPA) 805 Performance-Based Standard for Fire Protection for Li*. Huntersville, NC.

Electric Power Research Institute, & U.S. Nuclear Regulatory Commission. (2005). *EPRI / NRC-RES Fire PRA Methodology for Nuclear Power Facilities (EPRI-1011989 and NUREG/CR-6850) Volume 2 : Detailed Methodology*. Washington, D.C.

Entergy Operations. (2014). *License Amendment Request to Adopt NFPA 805 Performance-Based Standard for Fire Protection for Light Water Reactor Generating Plants (2001 Edition) Arkansas Nuclear One – Unit 1 Docket No. 50-313 License No. DPR-51*. Russellville, AR.

Guler, A., Mandelli, D., Alfonsi, A., Cogliati, J., Rabiti, C., & Aldemir, T. (2014). Methodology for the Incorporation of Passive Component Aging Modeling into the RAVEN / RELAP-7 Environment. In *ANS Winter Meeting*. Anaheim, CA: American Nuclear Society.

Hastie, T., Tibshirani, R., & Friedman, J. (2016). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. *Springer Series in Statistics* (2nd ed.). New York, NY: Springer. https://doi.org/10.1007/978-0-387-98135-2

Hill, K., Joglar, F., Najafi, B., KcGrattan, K., Peacock, R., & Hamins, A. (2007). *Verification and Validation of Selected Fire Models for Nuclear Power Plant Applications Volume 1: Main Report, NUREG-1824*. Washington, D.C.

Ho, V., Siu, N., Apostolakis, G., & Flanagan, G. (1986). *COMPBRN III - A Computer Code for Modeling Compartment Fires (NUREG/CR-4566)*. Washington, D.C.

Hothorn, T., & Zeileis, A. (2015). partykit: A Modular Toolkit for Recursive Partitioning in R. *Journal of Machine Learning Research*, *16*, 3905–3908. Retrieved from http://jmlr.org/papers/v16/hothorn15a.html

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, *9*(3), 99–104. https://doi.org/10.1109/MCSE.2007.55

Idaho National Laboratory. (n.d.). RAVEN GitHub. Retrieved June 3, 2017, from https://github.com/idaholab/raven

Idaho National Laboratory. (2017). *Light Water Reactor Sustainability Program Integrated Program Plan (INL/EXT-11-23452)*. *INL/EXT-11-23452*. Idaho Falls, ID.

Iqbal, N., Salley, M. H., & Weerakkody, S. (2004). *Fire Dynamics Tools (FDTs): Quantitative Fire Hazard Analysis Methods for the U.S. Nuclear Regulatory Commission Fire Protection Inspection Program*. *NUREG-1805*. Washington, D.C.

Karlssson, B., & Quintiere, J. (1999). *Enclosure Fire Dynamics* (1st Edition). CRC Press.

Kuhn, M. (2017). caret: Classification and Regression Training. Retrieved from https://cran.r-project.org/package=caret

Kuhn, M., & Johnson, K. (2016). *Applied Predictive Modeling*. New York, NY: Springer Nature. https://doi.org/10.1007/978-1-4614-6849-3

Li, G., Rosenthal, C., & Rabitz, H. (2001). High Dimensional Model Representations. *J. Phys. Chem. A*, *105*(33), 7765–7777.

Mandelli, D., Prescott, S., Smith, C., Alfonsi, A., Rabiti, C., Cogliati, J., & Kinoshita, R. (2015). A Flooding Induced Station Blackout Analysis for a Pressurized Water Reactor Using the RISMC Toolkit. *Science and Technology of Nuclear Installations*, *2015*.

Mandelli, D., Rabiti, C., & Alfonsi, A. (2012). Pre-Processing of Cross- Sections Using Dimensionality Reduction Techniques (INL/CON-12-26342). In *ANS Winter Meeting*. American Nuclear Society.

Mandelli, D., Smith, C., Alfonsi, A., & Rabiti, C. (2014). Analysis of the Space Propulsion System Problem Using RAVEN. In *Probabilistic Safety Assessment and Management PSAM 12*. Honolulu, HI.

Mandelli, D., Smith, C. L., Alfonsi, A., Rabiti, C., & Cogliati, J. (2015). *Light Water Reactor Sustainability Program Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization Toolkit (INL/EXT-15-35933)*. Idaho Falls, ID.

Mandelli, D., Smith, C. L., Rabiti, C., Alfonsi, A., Cogliati, J., & Kinoshita, R. (2013). New Methods and Tools to Perform Safety Analysis Within RISMC, INL/CON-13-29552. In *ANS Winter Meeting*. American Nuclear Society.

Mandelli, D., Smith, C., Ma, Z., Riley, T., Nielsen, Alfonsi, A., … Cogliati, J. (2014). *Light Water Reactor Sustainability Program Risk-Informed Safety Margin Characterization Development Work (INL/EXT-14-33191)*. Idaho Falls, ID.

Mandelli, D., Smith, C., Riley, T., Nielsen, J., Schroeder, J., Rabiti, C., … Maljovec, D. (2014). *Overview of New Tools to Perform Safety Analysis: BWR Station Black Out Test Case, INL/CON-14-31571*. *Probabilistic Safety Assessment & Management Conference PSAM 12*.

Manselli, D., Smith, C., Rabiti, C., Alfonsi, A., Youngblood, R., Pascucci, V., … Zamalieva, D. (2013). Dynamic PRA : An Overview of New Algorithms to Generate , Analyze and Visualize Data (INL/CON-13-29508). In *ANS Winter Meeting*. American Nuclear Society.

McGrattan, K., Hostikka, S., McDermott, R., Floyd, J., Weinschenk, C., & Overholt, K. (2017). *Fire Dynamics Simulator Technical Reference Guide Volume 2 : Verification (NIST Special Publication 1018-2)*. Gaithersburg, MD.

McGrattan, K., Peacock, R., Milke, J., Wachowiak, R., Joglar, F., LeStrange, S., … Zee, K. (2012). *Nuclear Power Plant Fire Modeling Application Guidelines (NPP FIRE MAG)*. *NUREG-1934*. Washington, D.C.

Miller, C., Cubbage, A., Dorman, D., Grobe, J., Holahan, G., & Sanfilippo, N. (2011). *Recommendations for enhancing reactor safety in the 21th century*. *SECY-11-0093*. Washington, D.C.

National Fire Protection Association. (2001). *Performance-Based Standard for Fire Protection for Advanced Light Water Reactor Electric Generating Plants (NFPA-805)*. Quincy, MA.

Nebraska Public Power District. (2012). *Nebraska Public Power District - Cooper Nuclear Station Docket No. 50-298, License No. DPR-46 License Amendment Request to Revise the Fire Protection Licensing Basis to NFPA 805 Per 10 CFR 50.48(c)*. Brownville, NE.

NIST. (1988). *Experimental Data Set for the Accuracy Assessment of Room Fire Models (NBSIR 88-3752)*. Gaithersburg, MD.

Peacock, R. D., Forney, G. P., & Reneke, P. A. (2017). *CFAST – Consolidated Fire And Smoke Transport (Version 7) Volume 3: Verification and Validation Guide (NIST Technical Note 1889v3)*. Gaithersburg, MD.

Peacock, R., Jones, W., Reneke, P., & Forney, G. (2008). *NIST Special Publication 1026 CFAST – Consolidated Model of Fire Growth and Smoke Transport (Version 6) Technical Reference Guide*. *NIST Special Publication 1041*. Gaithersburg, MD.

Rabiti, C., Alfonsi, A., Cogliati, J., Mandelli, D., & Kinoshita, R. (2012). *Reactor Analysis and Virtual Control Environment (RAVEN) FY12 Report (INL/EXT-27351)*. Idaho Falls, ID.

Rabiti, C., Alfonsi, A., Cogliati, J., Mandelli, D., Kinoshita, R., Sen, S., … Chen, J. (2017). *RAVEN User Manual (INL/EXT-15-34123)*. Idaho Falls, ID.

Rabiti, C., Alfonsi, A., Huang, D., Gleicher, F., Wang, B., Abdel-khalik, H. S., … Smith, C. L. (2015). *Light Water Reactor Sustainability Program System Reliability Analysis Capability and Surrogate Model Application in RAVEN (INL/EXT-16-37243)*. Idaho Falls, ID.

Rabiti, C., Alfonsi, A., Mandelli, D., Cogliati, J., & Kinoshita, R. (2014). *Advanced Probabilistic Risk Analysis Using RAVEN adn RELAP-7, INL/EXT-14-32491*. Idaho Falls, ID.

Rabiti, C., Alfonsi, A., Mandelli, D., Cogliati, J., & Martineau, R. (2012). RAVEN as Control Logic and Probabilistic Risk Assessment Driver for RELAP-7 (INL/CON-12-26352). In *2012 ANS Winter Meeting*. American Nuclear Society.

Rabiti, C., Alfonsi, A., Mandelli, D., Cogliati, J., Martineau, R., & Smith, C. L. (2013). *Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout (INL/EXT-13-29510)*. Idaho Falls, ID.

Rabiti, C., Talbot, P. W., Alfonsi, A., Mandelli, D., & Cogliati, J. (2013). *Implementation of Stochastic Polynomials Approach in the RAVEN Code (INL/EXT-13-30611)*. Idaho Falls, ID.

Rabitia, C., Alfonsi, A., Cogliati, J., Mandelli, D., & Kinoshita, R. (2014). RAVEN, a New Software for Dynamic Risk Analysis (INL/CON-14-31610). In *Probabilistic Safety Assessment and Management PSAM 12*. Honolulu, HI.

R Core Team. (2016). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.r-project.org/

Sanchez, S. M. (2011). Better than a Petaflop: The Power of Efficeint Experimental Design. In *Proceedings of the 2011 Winter Simulation Conference* (pp. 1441–1455). IEEE.

Sargent, R. G. (2008). Verification and Validation of Simulation Models. In *Proceedings of the 2008 Winter Simulation Conference* (pp. 157–169). IEEE. https://doi.org/10.1109/WSC.2000.899697

Sen, S., Maljovec, D., Alfonsi, A., & Rabiti, C. (2015). *Developing and Implementing the Data Mining Algorithms in RAVEN (INL/EXT-15-36632)*. Idaho Falls, ID. Retrieved from https://inldigitallibrary.inl.gov/sti/6799591.pdf

SFPE, & NFPA. (2002). *SFPE Handbook of Fire Protection Engineering* (3rd ed.). Quincy, MA: National Fire Protection Association.

Smith, C., Mandelli, D., Prescott, S., Alfonsi, A., Rabiti, C., & Cogliati, J. (2014). *Light Water Reactor Sustainability Program Analysis of Pressurized Water Reactor Station Blackout Caused by External Flooding Using the RISMC Toolkit (INL/EXT-14-32906)*. Idaho Falls, ID.

Steinwart, I., & Christmann, A. (2008). *Support Vector Machines*. *Analytica Chimica Acta* (Vol. 703). New York, NY: Springer. https://doi.org/10.1016/j.aca.2011.07.027

Swiler, Laura, Mandelli, Diego, Rabiti, Crisitan, Alfonsi, A. (2013). *DAKOTA Reliability Methods applied to RAVEN/RELAP-7 (SAND2013-8439)*. *SAND2013-8439*. Albuquerque, NM.

Szilard, R. H., Frepoli, C., Yurko, J. P., Youngblood, R., Alfonsi, A., Zoino, A., … Smith, C. L. (2015). *Light Water Reactor Sustainability Program Industry Application Emergency Core Cooling System Cladding Acceptance Criteria Early Demonstration (INL/EXT-36541)*. Idaho Falls, ID.

USNRC, & EPRI. (2015). *Refining and Characterizing Heat Release Rates from Electrical Enclosures During Fire (RACHELLE-FIRE) (NUREG-2178)*. Washington, D.C.

USNRC, & Sandia National Laboratories. (1987). *Enclosure Environment Characterization Testing for the Base Line Validation of Computer Fire Simulation Codes (NUREG/CR-4681, SAND86-1296)*. Albuquerque, NM.

USNRC, & Sandia National Laboratories. (1989). *A Summary of Nuclear Power Plant Fire Safety Research at Sandia National Laboratories, 1975–1987 (NUREG/CR-5384, SAND89-1359)*. Albuquerque, NM.

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Worrell, C., & Rochon, C. (2016). Fire Probabilistic Risk Assessment and its Applications in the Nuclear Power Industry. *Fire Technology*, *52*, 443–467.