5-2023

A Framework for the Verification and Validation of Artificial Intelligence Machine Learning Systems

Swala B. Burns

A FRAMEWORK FOR THE VERIFICATION AND VALIDATION OF ARTIFICIAL INTELLIGENCE MACHINE LEARNING SYSTEMS

A Dissertation

Submitted to the Graduate Faculty of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

in

Systems Engineering

by
Swala B. Burns
B.S. Mathematics, University of West Florida, 2001
M.S. Mathematics, University of West Florida, 2004
May 2023

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A3C	Asynchronous Advantage Actor Critic
AHP	Analytical Hierarchy Process
AI	Artificial Intelligence
AUC	Area Under Receiver Operating Characteristic Curve
BBN	Bayesian Belief Network
BMS	Bayesian Model Selection
BN	Bayesian Network
BP	Backpropagation
BVA	Boundary Value Analysis
CART	Classification and Regression Tree
CHAID	Chi-Square Automatic Interaction Detector
CIT	Combinatorial Interaction Testing
CNN	Convolution Neural Network
CRF	Conditional Random Fields
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning

DQN	Deep Q-Network
ECP	Equivalence Class Partitioning
ELM	Extreme Machine Learning
EM	Ensemble Model
EO	Evolutionary Optimization
FFNN	Feedforward Neural Network
FNI	Functional Network Inference
GA	Genetic Algorithm
GBM	Gradient Boosting Machine
GD	Gradient Descent
G-MCTS	Goal-Oriented Monte Carlo Tree Search
GS	Grid Search
HER	Hindsight Experience Replay
HMM	Hidden Markov Model
I2A	Imagination Augmented Agents
ID3	Iterative Dichotomizer 3
IEEE	Institute of Electrical and Electronics Engineers
JMV	Joint Multi-view
K-FCV	K-Fold Cross Validation
KNN	K Nearest Neighbor
LOCV	Leave One-Out Cross Validation
LR	Logistic Regression
LSTM	Long Short-Term Memory

LVQ	Learning Vector Quantization
LWL	Locally Weighted Learning
MAD	Mean Absolute Difference
MBSE	Model-Based Systems Engineering
MCDA	Multiple Criteria Decision Analysis
MCIA	Multicriteria Intelligence Aid
M-CNN	Multi-column Convolutional Neural Network
MCVS	Mean Cross Validation Score
MDA	Mean Decrease Accuracy
MDI	Mean Decrease Impurity
ML	Machine Learning
MLP	Multilayer Perceptron
MLR	Multiple Linear Regression
MNN	Modular Neural Network
MR	Multivariate Regression
MRH	Multiple Reduced Hypercube
MSA	Model Sensitivity Analysis
MUFS	Multiview Unsupervised Feature Selection
NN	Neural Network
OSLR	Online Structure Learner by Revision
PCA	Principal Component Analysis
PG	Policy Gradient
PNN	Probabilistic Neural Network

PPO	Proximal Policy Optimization
PROMETHEE	Preference Ranking Organization Method for Enrichment Evaluation
R2	R-Square
RBF	Radial Basis Function
R-CNN	Regional-based Convolutional Neural Network
RF	Random Forest
RLT	Reinforced Learning Tree
RNN	Recurrent Neural Network
RMSE	Root Mean Square Error
RS	Random Search
SARSA	State-Action-Reward-State-Action
SLP	Single Layer Perceptron
SLR	Simple Linear Regression
SOM	Self-Organizing Map
SPM	Spatial Pyramid Matching-Based
SR	Stepwise Regression
SVM	Support Vector Machine
SysML	System Modeling Language
TRPO	Trust Region Policy Optimization
UAT	User Acceptance Testing
UML	Unified Modeling Language
V&V	Verification and Validation

LIST OF TERMS

Analytic Hierarchy Process (AHP) - A quantitative linear combination method used in the organization and analysis of complex decisions. (Zhao, et al., 2013)

Asynchronous Advantage Actor Critic (A3C) – A policy gradient reinforcement learning algorithm that uses a neural network architecture. (Mnih, et al., 2016)

Area Under the Curve (AUC) – Measures the area underneath a Receiver Operating Characteristic Curve. (Kelleher, et al., 2020)

Backpropagation (BP) – A supervised deep-learning algorithm used to train feed-forward artificial neural networks. (LeCun, et al., 2015)

Bayesian Belief Network (BBN) – See Bayesian Network.

Bayesian Model Selection (BMS) – A Bayesian method that identifies the most important features for ML model generation. (Mamun, 2019)

Bayesian Network (BN) – Also known as a Bayesian Belief Network, a probabilistic graphical model that uses Bayesian inference for calculating decision probabilities. (Bhattacharya & Mishra, 2018)

Bootstrapping – For very small datasets (less than 300 instances) the bootstrap iteratively performs multiple evaluation experiments using slightly different training and test sets each time to evaluate the expected performance of a model. (Kelleher, et al., 2020)

Boundary Value Analysis (BVA) – A black-box test design techniques, which is used to find errors at boundaries of input domain. (Celestial Systems, 2023)

C5.0 – A classification algorithm that builds either a decision tree or a ruleset and measures purity by entropy. (Kelleher, et al., 2020)

C51 – A value-based learning algorithm based on categorical distributional reinforcement learning. (Rowland, et al., 2018)

Charade – An algorithm that automatically learns consistent rule systems from a description language, a set of axioms reflecting the language semantics and a set of examples. (Ganascia, 1987)

Chi-square Test – For machine learning, a feature selection method that calculates the Chi-square between each feature and target and then selects the desired number of features with the best Chi-square scores. (Vergara & Estevez, 2013)

Chi-square Automatic Interaction Detector (CHAID) - A method for building classification trees and regression trees from a learning sample comprising already-classified objects. An essential feature is the use of the chi-square test for contingency tables to decide which variables are of maximal importance for classification. Another aspect of CHAID is its ability to build non-binary classification trees. (Anon., 2023)

Classification and Regression Tree (CART) – A variant of the ID3, an algorithm that uses the Gini Index (a commonly used measure of impurity) instead of information gain to select features to add to a decision tree. CART is sufficient with continuous target features. (Mowbray, et al., 2020)

CN2 – An induction algorithm of simple, comprehensible production rules in domains where problems of poor description language and noise may be present. (Clark & Niblett, 1989)

Combinatorial Interaction Test Sampling (CIT) – Using combinatorial coverage for selecting and characterizing test and training sets for machine learning. (Cody, et al., 2022)

Conditional Inference Tree (unbiased recursive partitioning) – A non-parametric decision tree algorithm that uses recursive partitioning of dependent variables based on the value of correlations. This algorithm works for continuous and multivariate response variables, but is not good for data with missing values for learning. (Hothorn, et al., 2012)

Conditional Random Fields (CRF) – Related to the Hidden Markov Model, an algorithm using a conditional distribution combined with an associated graphical structure. Because the model is conditional, dependencies among the features do not need to be explicitly represented. (Sutton & McCallum, 2012)

Convolutional Neural Network (CNN) – A deep learning unsupervised algorithm that uses backpropagation to utilize multiple layers of simple nonlinear processing modules for feature extraction and transformation. (LeCun, 2018)

Correlation Coefficient – For feature selection; identifies the features with the highest correlation and linear dependence having similar effect on the target variable. One of the two correlated variables can be dropped. (Vergara & Estevez, 2013)

Decision Stump – A decision tree which uses only a single attribute for splitting. For discrete attributes, this typically means that the tree consists only of a single interior node. If the attribute is numerical, the tree may be more complex. (Sammut & Webb, 2011)

Decision Table Testing – A black-box test design technique used for different combinations of input (conditions) resulting in different outcomes (actions). (Kumar, 2021)

Deep Belief Network (DBN) - An unsupervised learning algorithm consisting of two different types of neural networks being Belief Networks and Restricted Boltzmann Machines. (Simplilearn, 2023)

Deep Boltzmann Machine (DBM) – A kind of Markov Random Field that has an undirected probabilistic graphical model containing one visible layer and several hidden layers. A DBM learns the input's complex internal representations using few labeled data for fine-tuning the representation created with a set of unlabeled input. (Lee, et al., 2021)

Deep Deterministic Policy Gradient (DDPG) - A model-free, online, off-policy reinforcement learning method. A DDPG agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward. (MathWorks, 2022)

Deep Learning (DL) - Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. (LeCun, et al., 2015)

Deep Q-Network (DQN) – A model-free reinforcement algorithm that uses the modern deep learning technique. DQN algorithms use Q-learning to learn the best action to take in the given state and a deep neural network or convolutional neural network to estimate the Q value function. (De-Yu, 2021)

Directional Test – A model validation test that observes how the change in perturbations, from the input, affects the model behavior. (Lau, 2022)

Dispersion Ratio – A filter feature selection tool that disperses the features through a ration of the arithmetic mean and the geometric mean. The higher the ratio, the more relevant the feature. (Gupta, 2023)

Ensemble Model (EM) – A set of models that make predictions by aggregating the outputs of these models. (Kelleher, et al., 2020)

Equivalence Class Partitioning (ECP) – A black-box system verification technique that divides the domain data into different equivalence data classes that are expected to

exhibit similar behavior. Thus, it reduces the number of test cases to a finite list of testable test cases covering maximum possibilities. (Celestial Systems, 2023)

Evolutionary Optimization (EO) – An algorithm that implements a meta-heuristic model of the behavior of biological evolution. These algorithms can be used to find approximate solutions to difficult numerical minimization problems. (McCaffrey, 2012)

Extreme Learning Machine (ELM) – An algorithm for the single hidden layer feedforward neural network (SLFN), which converges much faster than traditional methods. (Wang, et al., 2022)

Feedforward Neural Network (FFNN) – Also called deep network, multi-layer perceptron (MLP), or simply neural network. Feedforward neural networks are deep learning algorithms that progress forward without recycling. (Kelleher, et al., 2020)

Fischer's Score Method– For supervised feature selection, a method which returns the ranks of the variables based on the Fisher's Score in descending order. (Vergara & Estevez, 2013)

Functional Network Inference (FNI) – An algorithm that relies on a group of variables, which have measurable values, and interact in some way such that the value of one variable affects the value of another. (Zanin, 2021)

Functional Testing - A type of black-box testing that seeks to establish whether each application function works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. (Katalon, 2023)

Fuzzy Sets – Also known as fuzzy logic, is a mathematical way of representing vague or imprecise measures in decision science. Fuzzy sets are functions that map each member of a set to a real number in [0, 1] to indicate the degree of membership of that element. (Dayo-Olupona, et al., 2020)

Gaussian Naïve Bayes - A classification algorithm used in machine learning based on the probabilistic approach and Gaussian distribution. Gaussian Naive Bayes assumes that each feature is an independent predictor of the target variable. (Kelleher, et al., 2020)

Genetic Algorithm (GA) – An algorithm that searches the solution space using a nature-inspired process based on genetic evolution. GA uses a population of candidate solutions, which are evaluated based on their fitness. Solutions with a higher fitness are more likely to continue in the genetic process, whereas poor solutions are phased out. (Schulte, et al., 2021)

Gradient Boosting Machine (GBM) – An algorithm for training ensemble models using boosting where each new model added to an ensemble is biased to pay more attention to

instances that previous models misclassified. This is done by incrementally adapting the dataset used to train the models. (Yu, et al., 2020)

Gradient Descent (GD) – An optimization algorithm that trains machine learning models and neural networks by using a guided search from a random starting position. (Kelleher, et al., 2020)

Grid Search (GS) – An algorithm for hyperparameter optimization that defines a search space as a grid of hyperparameter values and evaluates every position in the grid. (Brownlee, 2020)

Hidden Markov Model (HMM) - A class of probabilistic graphical models where implicit and stochastic processes can be inferred indirectly through a sequence of observed states. HMM is commonly used in speech recognition, handwriting recognition, and biological sequences. (Karaca, et al., 2022)

Hierarchical Clustering – An unsupervised clustering algorithm which involves creating clusters that have predominant ordering from top to bottom. (Kelleher, et al., 2020)

Hindsight Experience Replay (HER) – A reinforcement algorithm that converts a low-reward trajectory of one desired goal to a high-reward trajectory for the unintended goal. (Li, Pinto, et al., 2020)

Hold-Out Sampling – Removing a test set from the data to use in evaluating the performance of the model. The test set is not used to train the model. (Kelleher, et al., 2020)

Human Factors Testing – Operational validation testing process to observe how a representative user actually uses a system in the intended environment. The testing gauges the performance in terms of the likelihood of error or difficulty in use. (Harper, 2016)

Imagination-Augmented Agents (I2A) – A reinforced learning algorithm that uses both model-based and model-free learning. (Abbasi, et al., 2021)

Integration Testing – System validation testing of the combined modules and components to ensure functionality as a group. (Katalon, 2023)

Invariant Test – A model validation method to test whether a ML model produces consistent results under different conditions. (Liao, et al., 2022)

Iterative Dichotomizer 3 (ID3) – An algorithm that attempts to create the shallowest decision tree that is consistent with the data given. (Kelleher, et al., 2020)

K-Fold Cross Validation (K-FCV) – A commonly used sampling method that divides available data into k equal-sized folds (or partitions), and k separate evaluation experiments are performed. (Kim & Shin, 2020)

K-Means – An unsupervised algorithm that partitions the dataset into k non-overlapping cluster groups, where each data point is a member of one and only one group. K-means use the mean vector of each cluster. (Kelleher, et al., 2020)

K-Means Clustering – An unsupervised algorithm that analyzes clustering in pattern recognition and machine learning. (Sinaga & Yang, 2020)

K-Medians – An unsupervised algorithm that partitions the dataset into k non-overlapping cluster groups, where each data point is a member of one and only one group. K-medians use the median vector of each cluster. (Kelleher, et al., 2020)

K Nearest Neighbor (KNN) – A nearest neighbor algorithm that is resistant to noise introduced by the training data. The algorithm returns the majority target level within the set of k nearest neighbors to the query. (Chen, et al., 2020)

Learning Vector Quantization (LVQ) - An artificial neural network algorithm that lets you choose how many training instances to retain and learns exactly what those instances should look like. (Brownlee, 2020)

Leave-One-Out Cross Validation (LOCV) – Also known as jackknifing, is an extreme form of k-fold cross validation in which the number of folds is the same as the number of training instances. (Kelleher, et al., 2020)

Locally Weighted Learning (LWL) – An algorithm that uses a class of function approximation techniques, where a local model is created for each point of interest based on neighboring data, instead of building a global model for the whole function space. LWL methods are non-parametric. (Yen, et al., 2021)

Logistic Regression (LR) – A supervised learning classification algorithm that assigns instances to a set of classes or categorical target variable values. (Kelleher, et al., 2020)

Long Short-Term Memory (LSTM) – A combination of recurrent neural networks that are capable of long-term memory of order in sequence prediction. (Martinez, et al., 2020)

M5 – A tree algorithm that assigns linear regression functions at the terminal nodes and fits a multivariate linear regression model to each subspace by classifying or dividing the whole data space into several sub spaces. The M5 tree method deals with continuous class problems instead of discrete classes and can handle tasks with very high dimensionality. (Sihag, et al., 2019)

Mean Absolute Difference (MAD) – A filter feature selection method that computes the absolute difference from the mean value. The higher the calculated MAD, the better the discriminatory power between features. (Gupta, 2023)

Mean Decrease in Accuracy (MDA) – A method that ranks feature importance by removing each feature from the model and measuring the decrease in accuracy or the increase in the mean square error. (Bhattacharya & Mishra, 2018)

Mean Decrease in Impurity (MDI) – A ranking method to measure the feature importance as the sum over the number of splits (across all decision trees) that include the feature, proportionally to the number of samples it splits. (Bhattacharya & Mishra, 2018)

Minimum Functionality Test – A modeling performance method that applies a collection of simple examples to check model behavior within a capability. (Ribeiro, et al., 2020)

Model Sensitivity Analysis (MSA) – A technique for measuring model sensitivity that involves a series of methods to quantify how the uncertainty in the output of a model is related to the uncertainty in its inputs. (Salciccioli, et al., 2016)

Modular Neural Network (MNN) – The combining of multiple neural networks that behave as modules to solve parts of a problem by integrating the responses of the modules into a final output of a system. (Shukla, et al., 2010)

Multiple Criteria Decision Analysis (MCDA) – A decision-making process that takes into account multi-dimensional factors. (Zhao, et al., 2013)

Multilayer Perceptron (MLP) – A supervised algorithm that has at least one extra hidden layer of perceptron network allowing for learning of non-linear functions. (Yu, et al., 2020)

Multiple Linear Regression (MLR) – A supervised algorithm relating multiple feature variables to a single target variable. (Kelleher, et al., 2020)

Multiple Reduced Hypercube (MRH) – A high-speed computer network algorithm based on a hypercube. (Sim, et al., 2010)

Multivariate Regression (MR) – A supervised algorithm relating multiple feature variables to multiple target variables. (Kelleher, et al., 2020)

Naïve Bayes – A supervised learning algorithm based on the Bayes Theorem to classify objects for prediction with a naïve assumption of conditional independence between every pair of features given the value of the class (target) variable. (Kelleher, et al., 2020)

Negative Testing – A system validation technique that verifies if the application is equipped to handle challenging or invalid inputs. (Joseph, 2021)

Online Structure Learner by Revision (OSLR) – An algorithm that learns a model online from a stream of examples that represent what happens in a time instance of the data. (Guimaraes & Costa, 2022)

Orthogonal Arrays – A method used in sampling to combine orthogonal trials, thereby lowering the sample size while still providing adequate sample coverage for testing. This method can only be used when trial combinations are small enough to be accessible. (Jarosz, et al., 2019)

Out of Time Sampling – The use of time dimensional data to make the training set from one period of time and the test set from another period of time. (Kelleher, et al., 2020)

Policy Gradient (PG) – A model-free, online, on policy reinforcement learning algorithm. A PG agent is a policy-based reinforcement learning agent that uses the reinforcement algorithm in searches for an optimal policy that maximizes the expected cumulative long-term reward. (Williams, 1992)

Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) - A method for evaluating the criteria of alternatives by pairwise comparison in multi-criteria decision-making. (Dayo-Olupona, et al., 2020)

Principal Component Analysis (PCA) – A statistical approach that can be used to examine high-dimensional data and capture the most important information for a model. The original data is transformed into a lower-dimensional space while collating highly correlated variables together. (Datacamp, 2023)

Probabilistic Neural Network (PNN) – A feedforward neural network classifier that estimates the probability density function (pdf) of the data. (Mohebali, et al., 2020)

Progol – An inductive logic programming algorithm that combines inverse entailment with general-to-specific search via a refinement graph. (Muggleton, 1995)

Proximal Policy Optimization (PPO) – A policy gradient algorithm for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a surrogate objective function using stochastic gradient ascent. (Schulman, et al., 2017)

Q-learning – A reinforced model-free, value-based, off-policy learning algorithm. (Kelleher, et al., 2020)

R-Square (R2) – A domain independent measure of model performance that is frequently used for prediction problems with a continuous target variable. R2 has a range from [0,1). Larger values indicate a better model performance. (Kelleher, et al., 2020)

Radial Basis Function (RBF) – A neural network in its simplest form as a three-layer feedforward neural network. The first layer corresponds to the inputs to the network; the second layer is hidden and consists of a number of RBF non-linear activation units; and the third layer corresponds to the final output of the network. (Hoang & Bui, 2017)

Random Forest (RF) – Model ensembles (sets of models for prediction) combining bagging, subspace sampling, and decision trees. Once the individual models in an ensemble have been induced, the ensemble makes predictions by returning the majority vote or the median, depending on the type of prediction required. (Bhattacharya & Mishra, 2018)

Random Search (RS) - An algorithm for hyperparameter optimization that defines a search space as a bounded domain of hyperparameter values and randomly chooses sample points in that domain. (Brownlee, 2020)

Recurrent Neural Network (RNN) – A deep learning unsupervised algorithm for data with a sequential varying-length structure. RNN has the capacity to remember relevant information from earlier in the sequence. (Sung, et al., 2019)

Region-based Convolutional Neural Network (R-CNN) - A two-stage detection algorithm that combines rectangular region proposals with convolutional neural network features. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region. (Ren, et al., 2015)

Regression Testing – A system validation test of stability and functionality performed every time alterations are made to a ML system. (Katalon, 2023)

Reinforced learning tree (RLT) – A traditional random forests algorithm that is able to split the variable selection and mute the noise variable. (Gajewski & Valis, 2020)

Requirements Traceability – The ability to match system requirements to the objectives and results of test cases in a testing event.

Root Mean Square Error (RMSE) – The standard deviation of the prediction errors (residuals) found in model validation. The RMSE determines model accuracy. (Kelleher, et al., 2020)

RULEX – A logic-logic based algorithm that uses if-then statements to formulate decisions. (Shen, 2020)

Sanity Testing – A system validation test conducted for a new build with minor bug fixes or code changes. (Katalon, 2023)

Security Testing – A validation testing technique that uncovers vulnerabilities, threats, and risks in a system to prevent malicious attacks from intruders. (Hamilton, 2023)

Self-Organizing Map (SOM) – An unsupervised artificial neural network algorithm based on a competitive learning strategy that generates a low-dimensional discrete map by learning data in a high-dimensional input space. (Song & Xia, 2022)

Spatial Pyramid Matching-based (SPM) Traditional Image Kernel – An image classification algorithm that incorporates the spatial information of local visual descriptors into the histograms. (Xie, et al., 2018)

Simple Linear Regression (SLR) – A regression algorithm that models the relationship between a feature variable and a target variable using a straight line. (Kelleher, et al., 2020)

Single Layer Perceptron (SLP) – A feed-forward single layer artificial neural network algorithm that classifies linear cases with a binary target variable. (Li, Hu, et al., 2020)

Smoke Testing – A system validation test performed for a new build to test the most critical and core functionalities to confirm that they yield expected results. (Katalon, 2023)

State-Action-Reward-State-Action (SARSA) – An on-policy algorithm used in reinforcement learning to train a Markov decision process model on a new policy. (Gautam, 2023)

State Transition Testing – A process-oriented test design technique that focuses on states, events that initiate a transition to another state, and actions resulting from such events. (Sogeti, 2023)

Stepwise Regression (SR) – The step-by-step iterative search process involving the selection of feature variables to be used in a final model. (Kelleher, et al., 2020)

Supervised Algorithm – An algorithm that learns from labeled training data to make output predictions based on input data. (Kelleher, et al., 2020)

Support Vector Machine (SVM) – A supervised algorithm that uses decision boundary instances, known as support vectors, to maximize the margin of error and separate the target feature levels. SVM applies in both classification and regression cases. (Kelleher, et al., 2020)

Trust Region Policy Optimization (TRPO) – A policy search algorithm in reinforcement learning in which a surrogate problem, that restricts consecutive policies to be close to one another, is iteratively solved. (Shani, et al., 2020)

Unsupervised Algorithm – An algorithm that identifies patterns in the data without data labels. (Kelleher, et al., 2020)

Unit Testing - Executed by developers, the purpose of unit testing is to validate the functionality of a unit or component, making sure the desired outputs are generated given a set of inputs. (Katalon, 2023)

User Acceptance Testing (UAT) – The final stage of validation testing, performed by actual users to determine if a system performs as expected in real-world application. (Elazar, 2023)

Variance Threshold – A simple baseline approach to feature selection by removal of all features whose variance does not meet some threshold. (Gupta, 2023)

Validation – The portion of testing that ensures the system properly performs with representative users in the intended environment. (Fisher, 2007)

Verification – The portion of testing that confirms that the system adheres to the requirements and specifications developed and given by the major stakeholders. (Fisher, 2007)

ABSTRACT

Swala B. Burns, Ph.D., University of South Alabama, May 2023. A Framework for the Verification and Validation of Artificial Intelligence Machine Learning Systems. Chair of Committee: Kari, J, Lippert, D.Sc.

An effective verification and validation (V&V) process framework for the white-box and black-box testing of artificial intelligence (AI) machine learning (ML) systems is not readily available. This research uses grounded theory to develop a framework that leads to the most effective and informative white-box and black-box methods for the V&V of AI ML systems. Verification of the system ensures that the system adheres to the requirements and specifications developed and given by the major stakeholders, while validation confirms that the system properly performs with representative users in the intended environment and does not perform in an unexpected manner.

Beginning with definitions, descriptions, and examples of ML processes and systems, the research results identify a clear and general process to effectively test these systems. The developed framework ensures the most productive and accurate testing results. Formerly, and occasionally still, the system definition and requirements exist in scattered documents that make it difficult to integrate, trace, and test through V&V. Modern system engineers along with system developers and stakeholders collaborate to produce a full system model using model-based systems engineering (MBSE). MBSE employs a Unified Modeling Language (UML) or System Modeling Language (SysML) representation of the system and its requirements that readily passes from each stakeholder for system information and additional input. The comprehensive and detailed MBSE model allows for direct traceability to the system requirements.

To thoroughly test a ML system, one performs either white-box or black-box testing or both. Black-box testing is a testing method in which the internal model structure, design, and implementation of the system under test is unknown to the test engineer. Testers and analysts are simply looking at performance of the system given input and output. White-box testing is a testing method in which the internal model structure, design, and implementation of the system under test is known to the test engineer. When possible, test engineers and analysts perform both black-box and white-box testing. However, sometimes testers lack authorization to access the internal structure of the system. The researcher captures this decision in the ML framework.

No two ML systems are exactly alike and therefore, the testing of each system must be custom to some degree. Even though there is customization, an effective process exists. This research includes some specialized methods, based on grounded theory, to use in the testing of the internal structure and performance. Through the study and organization of proven methods, this research develops an effective ML V&V framework. Systems engineers and analysts are able to simply apply the framework for various white-box and black-box V&V testing circumstances.

CHAPTER I

OVERVIEW

1.1 Introduction

In the last decade, artificial intelligence (AI) machine learning (ML) has grown to be a market of over $16 billion and is expected to reach a value of $164.05 billion by 2028 (SkyQuest Technology, 2022). ML is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. It focuses on developing computer programs that can access data and use that data to learn for itself (Selig, 2022). ML systems are a subtype of software systems. In traditional software systems, humans write the logic which interacts with data to produce a desired behavior. In these systems, software tests help ensure that the written logic aligns with the actual expected behavior. In ML software systems, humans provide desired behavior as examples during training and the model optimization process produces logic of the system (Jordan, 2020). Figure 1 below shows the difference between the traditional software system and the ML software system performance flow logic.

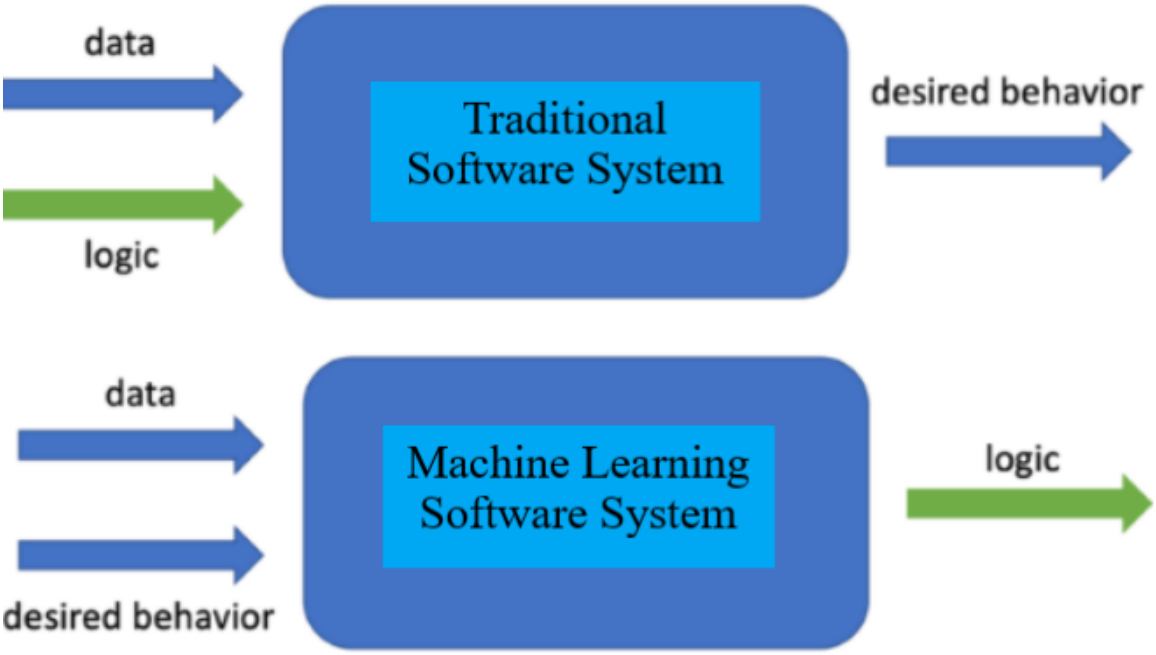

Figure 1. Traditional Software System vs. the ML Software System.

Today, ML is widely used in many applications for decision making and prediction. One common ML application is voice recognition software (Figure 2). Voice recognition software accepts speech and analyzes it for transcription into text. It works by splitting speech into individual sound bites and analyzing the sounds with ML algorithms to determine a system model. The model contains system feature values that predict the words, thereby transcribing sounds into text. The ML algorithms that perform most effective for this application are deep learning (DL) neural networks (NN).

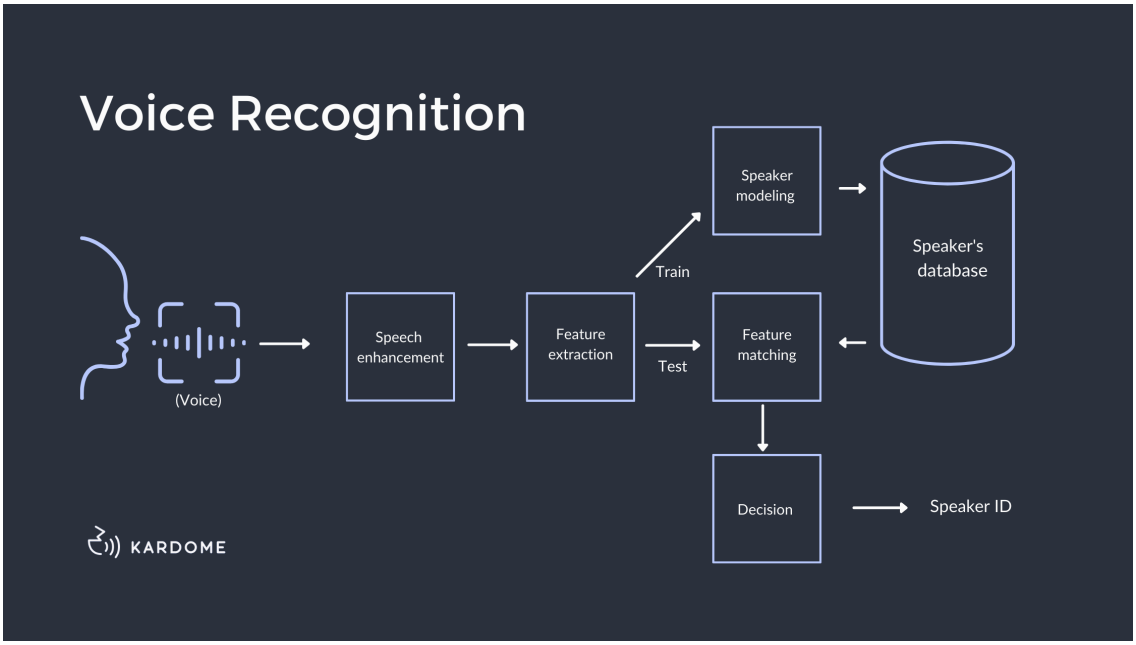

Figure 2. Machine Learning Voice Recognition Software Process Flow Diagram. (KARDOME, 2022)

Another common ML application system is one that makes a medical diagnosis. For example, doctors now use ML systems to detect and identify pneumonia. Figure 3 below shows a patient's lungs with acute pneumonia present. The ML algorithm convolutional neural network (CNN) is applied here to process x-ray images for pneumonia risk detection.

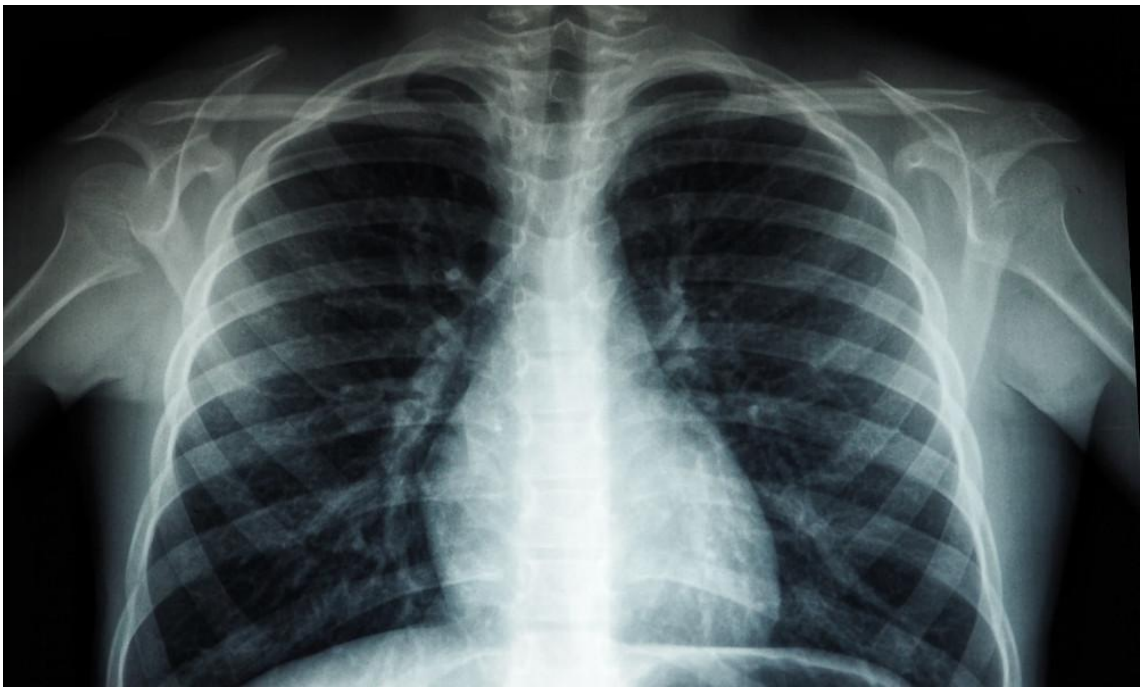

Figure 3. X-ray of Human Lungs with Pneumonia. (Umanoide, 2021)

These are only two examples of ML applications used commonly today. There now exist hundreds of ML systems. Working together, data scientists and systems engineers create and produce these systems to make lives better.

1.2 Background

V&V are required and important parts for the successful development and production of systems. Independent testing agencies should verify and validate the system during development, procurement, and during the remaining lifecycle of the system. Verification of the system ensures that the system adheres to the requirements and specifications developed and given by the major stakeholders, while validation confirms that the system properly performs with representative users in the intended

environment and does not perform in an unexpected manner. Systems engineers and analysts follow a stringent overarching V&V process for every system development and acquisition. However, the methods used in the process of V&V may differ from system to system. Figure 4 below, illustrates a specific V-model (named for its shape) that is used by NASA. Notice that the entire right side of the "V" represents the V&V process. Each step requires at least one separate testing event. "Operation and Maintenance" occur throughout the entire system lifecycle.

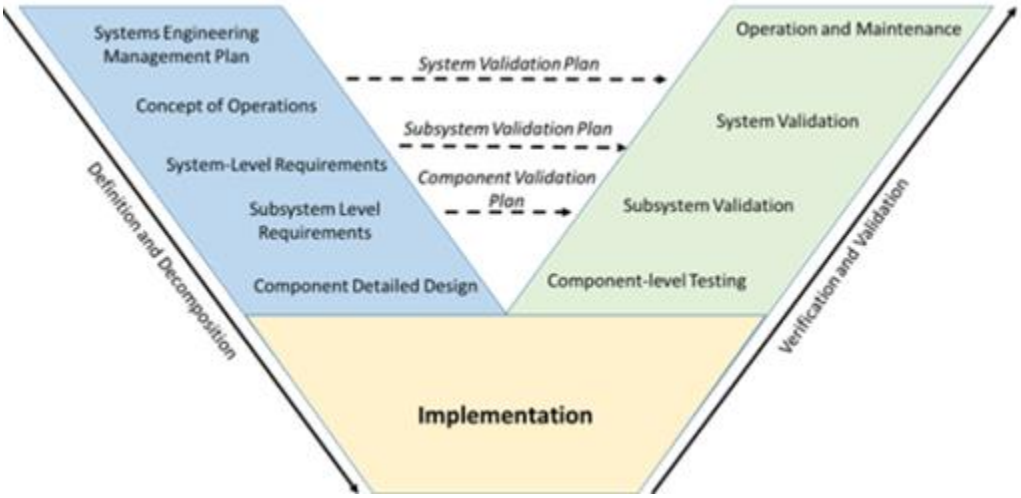

Figure 4. NASA V-Model for System Development. (Hart, 2018)

Testers use a more specific V-model for testing software systems. ML systems are a subtype of software systems which allows their testing to be applied to the software system V-model. Figure 5 below depicts the V&V for automotive software development components, which may include ML subsystem components. Notice that the V-model from Figure 4 is a generalization of Figure 5, specialized V-model. Testing is still required at each step on the right side of the V-model.

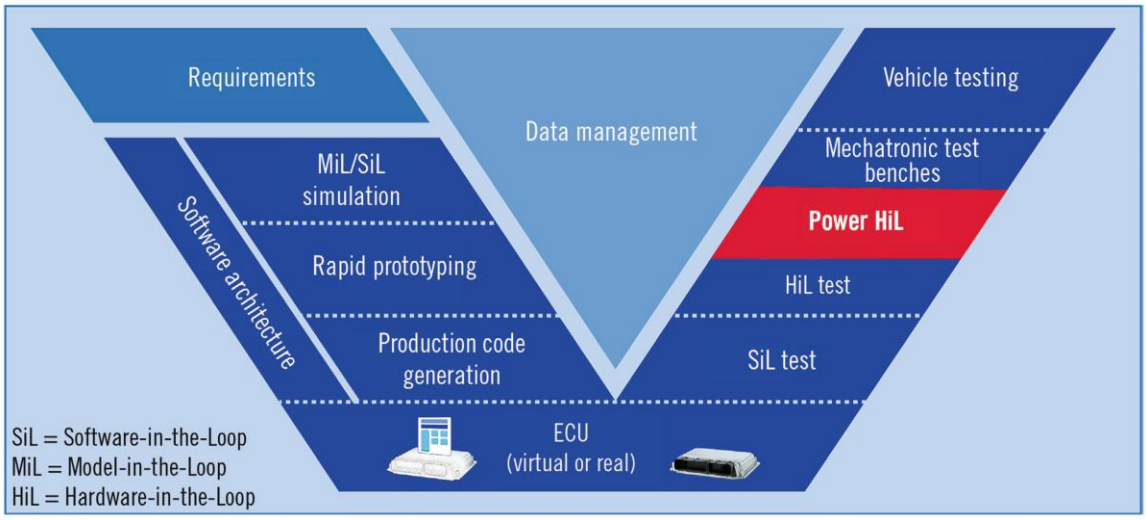

Figure 5. V-model for Automotive Software Development Components. (Mihalic, et al., 2022)

A single V&V testing process for a ML system begins with the start of the test design to the result and conclusion of the test trials. The software test design includes test objectives, assumptions, constraints and limitations, risks, controls and factors (features) with levels, randomization, and sampling. Each test trial includes designed inputs to and the resulting outputs from to the system. After the execution of all trials, the tester analyzes and reports the results using applicable tables and charts. The results also include any failures and all circumstances surrounding the failures, such as feature levels. The tester reports any deviations from the test design, so as to account for design changes and possible effect on the results.

Finding the cause and effect of the failures and improving the system, involves the testing of the internal structure of the ML system (white-box testing). White-box testing involves specialized testing methods. The methods depend on the ML system

model type which is dependent on the system data features and targets. For example, if one has an output target variable that has a categorical binary response, then one may try a supervised neural network algorithm such as Logistic Regression (LR).

Figure 6 below categorizes the many different ML model types. All ML algorithms fall under at least one of these ML subsets. There are diagrams that include algorithms listed under the categories; however, these diagrams are not inclusive. There are over a hundred different algorithms today and engineers are consistently developing new algorithms to accommodate modern ML situations.

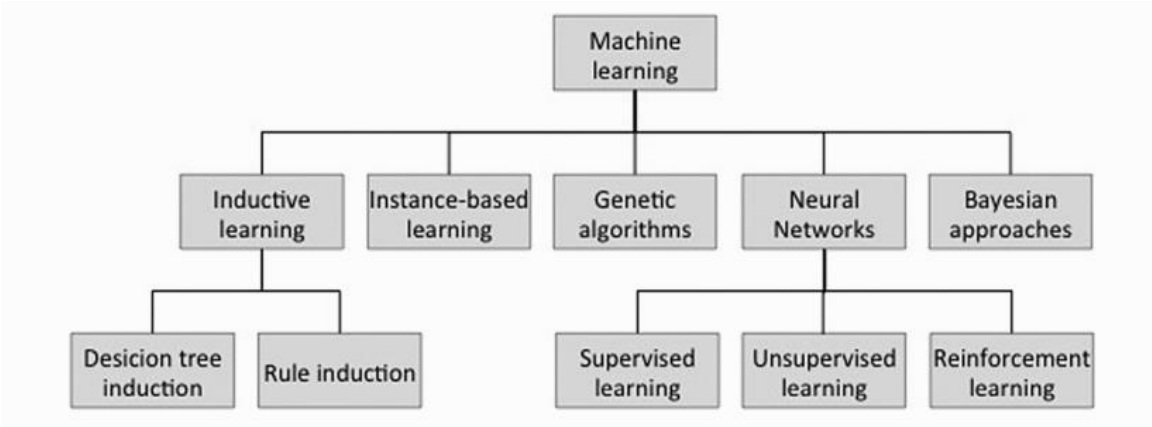

Figure 6. Machine Learning Category Diagram. (Wuest, et al., 2016)

Today, systems engineers and analysts are using MBSE as a modeling approach to systems engineering. MBSE allows direct collaboration of all stakeholders through a Unified Modeling Language (UML) or System Modeling Language (SysML) system model. The comprehensive and detailed system model allows for direct traceability to the system requirements. MBSE provides a comprehensive system representation that is extremely important to ensure a complete, clear and effective V&V process.

There are two main types of V&V testing methods: white-box and black-box. White-box testing is a testing method in which the internal model structure, design, and implementation of the system under test is known to the systems engineer, whereas, black-box testing is a testing method in which the internal model structure, design, and implementation is unknown to the systems engineer. In black-box testing, testers and analysts are simply looking at performance of the system given input and output. White-box testing is much more extensive. There are many tests used for analyzing the different parts of the internal structure of the ML system. When possible, testing engineers and analysts perform both white-box and black-box testing methods.

1.3 Problem Statement

An effective V&V process for testing ML systems is one that has easy to follow steps leading to the most accurate results. The process will correctly identify and locate any and all failures in the system while illuminating their cause and effect. This includes both white-box and black-box testing methods.

Currently, the ML methods and processes for the V&V of a ML system are extremely complex and decentralized. They are scattered throughout articles and texts with no single reference point. A research gap occurs when searching for a clear and general V&V procedure for the full white-box and black-box testing of a ML system. A systems engineer or analyst needs an effective framework to guide them through the V&V process of a ML system.

No two ML systems are exactly alike requiring the testing of each system to be custom to some degree. Despite the customization, an effective process is needed. This

must include methods for each complexity level of system model type to adequately contribute to a complete framework. The desired framework sets up a basic conceptional structure that explains the general ML V&V process. The problem statements for this research are:

- The ML process and the current methods for the V&V of a ML system are extremely complex and disorganized.
- There is no clear and general V&V procedure for the full white-box and black-box testing of a ML system.
- There must be available methods given for each different system model type for each ML process phase to adequately contribute to a complete framework.

1.4 Research Objectives

This research focuses on the following objectives:

1. To identify a general ML process from the systems engineer's perspective.
2. To generate an effective framework for the white-box and black-box V&V of a ML system by ML model type (Figure 6) for each phase of the ML process.

1.5 Research Methods

To identify and provide an effective V&V framework that makes testing of ML systems easier resulting in safe and reliable systems, the researcher performs a thorough literature review. This research involves reviewing a wide variety of articles on many

different ML topics dealing with process, methods, verification, validation, and performance. This includes the examination and categorizing of articles containing methods for the V&V of ML systems. The ML field of study is growing so rapidly making it important to review the most recent articles. The researcher uses a case study to validate the final framework product.

1.6 Anticipated Research Contributions

Many systems engineers, analysts, and other testers are not experienced or trained on the V&V for ML systems since it has only become relevant in the couple of decades. When given the task of performing the V&V for these systems, testers have no clear direction to begin the process. They must consult many articles, bringing the methods together and trying to make sense of it all. This research identifies a clear process for testers to use as guidance when testing ML systems.

Ultimately, this research provides guidance to all testing stakeholders for the V&V of ML systems. Since ML systems have become numerous and many are essential to the safety and security of humans, accurate and reliable V&V of these systems is important to all of society. This research helps to improve ML testing by providing clear and effective guidance thereby eliminating the risk of inaccurate testing results and unreliable systems.

CHAPTER II

LITERATURE REVIEW

Through literature review, this research examines the full ML process and the current methods of the V&V of a ML system. The researcher chooses the most effective methods based on simplicity, understandability, and consistency in quality reports from other researchers. It is necessary to understand that the ML system includes the ML model and all components of the process that work together to predict. Therefore, the literature review covers all parts of the ML system and processes. The literature review also considers both white-box and black-box system testing since the research delivers both in a framework. To reinforce the ML V&V test process that will develop from the research, the author also reviews case studies to identify proven effective methods. Case studies are identified during each separate ML process given below. Only the most recent and applicable articles with the most effective processes are summarized in this literature review.

2.1 Feature Selection Algorithms and Ranking

The goal of any feature selection approach is to identify the smallest subset of descriptive features that maintain overall model performance. For a research project involving the verification and validation of ML systems, it is crucial for the analyst and

systems engineer to review and inspect the techniques used during preparation, learning, model building, and implementation of the system model. The overall performance of the system ML model is highly dependent on the correct selection and use of these techniques. The most commonly used decision-making tool is the multi-criteria decision analysis (MCDA) technique which determines an optimal set for feature selection. From the following reviewed articles, MCDA tools demonstrate excellent performance in feature selection and ranking.

Bhattacharya and Mishra discuss in their article that when using decision trees, such as random forest, two methods to use for feature selection and ranking are Mean Decrease Impurity (MDI) and Mean Decrease Accuracy (MDA) (Bhattacharya & Mishra, 2018). MDI measures the number of times a feature splits a node on the decision tree, weighted by the number of samples it splits. In other words, each feature importance is the sum over the number of splits (across all trees) that include the feature, proportionally to the number of samples it splits. MDA ranks feature importance by removing each feature from the model and measuring the decrease in accuracy or the increase in the mean square error. Simply performing a regression analysis with software calculates the mean square error and shows model accuracy. These two methods are successful at choosing features and ranking to form an accurate model for facies and fracture predictions in reservoirs.

The MCDA Analytic Hierarchy Process (AHP) is a quantitative linear combination method used in the organization and analysis of complex decisions. AHP helps to decide between alternatives based on given criteria. Five different reviewed articles use MCDA AHP in some form. One article applies MCDA and ML to ship

classification with the High Resolution TerraSAR-X Imagery system (Zhao, et al., 2013). This article uses AHP with the ML algorithm, K-Nearest Neighbors (KNN). The authors apply AHP before ML to select the best features for ship classification by the KNN algorithm and to choose the best measures for each feature. In a similar process, another article that discusses and chooses the best urban land use, applies AHP to a ML algorithm in a novel goal-oriented Monte Carlo Tree Search (G-MCTS) method (Chen, et al., 2020). Again, this process applies AHP first to weight the features and to decide on the best heterogeneous goals that provide optimal land use solutions. Using the selected goals, the authors develop a goal-reasoning AI ML method for urban planning tasks, G-MCTS. The authors show that goal-oriented ML is better for this situation than value-oriented ML, necessitating the development of the novel G-MCTS. In another example, MCDA AHP combined with two hybrid algorithms, facilitate the assessment of flood risk in Vietnam (Pham, et al., 2020). The main objective in the article is to develop a novel approach which is a combination of hybrid ML models, AdaBoost-DT and Bagging-DT, and MCDA techniques for creating a flood risk assessment map for Quang Nam province which is one of the flood-prone areas of Vietnam. The authors use a Decision Table (DT) which is a base algorithm for flood susceptibility assessment and mapping.

Another type of MCDA tool often used in conjunction with AHP is fuzzy set theory. Fuzzy sets are functions that map each member of a set to a real number in [0, 1] to indicate the degree of membership of that element. Fuzzy sets remove ambiguity from natural language. The use of fuzzy sets is important in an article that selects the best technology for surface mines by combining the MCDA techniques AHP and preference ranking organization method of enrichment (PROMETHEE) (Dayo-Olupona, et al.,

2020). After weight determination with AHP, PROMETHEE with fuzzy set theory helps to perform the process evaluation. In another article, seeking to anticipate decisions by the opposing team in intelligence, the authors extend MCDA methodology to the context of intelligence analysis and propose Multicriteria Intelligence Aid (MCIA) with fuzzy set theory (Anissa, 2017). In the case of intelligence analysis, one cannot, as in the classical MCDA, structure the problem along with the decision-maker, but rather one needs to use available data to predict the decision-maker's preferences and anticipate his future decisions. MCIA adapts MCDA concepts to account for the differences in intelligence analysis. The steps for the study consist of (i) structuring the competitor/threat decision problem (MCIA), (ii)handling imperfect data (fuzzy set theory), (iii) modeling the analyst's risk attitude (MCDA), and (iv) aggregating the performance of the generated potential actions (AI / ML).

Sometimes feature selection does not consider the target variable. Such is the case with unsupervised NN. For unsupervised feature selection, data engineers will usually either drop incomplete features, drop features with high multicollinearity, or drop features with (near-)zero variance. However, some recent articles use analytical techniques for unsupervised learning models including two such methods: Joint Multi-View (JMV) and Multi-view Unsupervised Feature Selection (MUFS) (Fang, et al., 2022) (Huang, et al., 2022).

2.2 Sampling Algorithms and Risk Determination

Kelleher, Mac Namee, and D'Arcy recommend several sampling methods (Kelleher, et al., 2020). One method is known as hold-out sampling. Using hold-out

sampling, the authors take one sample from the overall dataset to use to train a model and another separate sample to test the model. This method is most appropriate for very large datasets and can include a third set called the validation set. The validation set is used when data outside the training set is required in order to tune particular aspects of a model. There are no fixed recommendations for how large the datasets should be, although the ratio of training:validation:test splits of 50:20:30 or 40:20:40 are common. Even though the authors do not provide a method for sample size to minimize risk of inability to discover failure in the system, they do endorse the avoidance of overfitting. Overfitting occurs when the prediction model selected by the algorithm is so complex that the model fits the dataset too closely and becomes sensitive to noise in the data. These models do not generalize well and, therefore, will not make good predictions. Overfitting is usually tested with the validation set of data. By testing and avoiding overfitting, the sample size is controlled and set in an amount that provides the best accuracy in the results, thereby lowering the risk of failures. Testers must have enough data though to have a large enough validation set to detect where the model begins to disimprove.

For small datasets; Kelleher, Mac Namee, and D'Arcy employ bootstrapping (Kelleher, et al., 2020). Bootstrapping iteratively performs multiple evaluation experiments using slightly different training and test sets each time to evaluate the expected performance of a model. To generate partitions for an iteration of the bootstrap, the tester takes a random selection of instances from the full dataset to generate a test set, and uses the remaining instances as the training set. Using the training set to train a model and the test set to evaluate it, the tester calculates a performance measure (or

measures) for the iteration. The process repeats for k iterations and the average of the individual performance measures gives the overall performance of the model. Typically, k is set to values greater than 200.

K-Fold Cross Validation (K-FCV) assists in choosing samples from the original dataset for training, validation, and test of k evaluation experiments. K-FCV minimizes sample bias issues. One reviewed article utilizes K-FCV with an ANN to choose k samples for training and test datasets (Kim & Shin, 2020). The team divided the samples allotting 75% for the training data and 25% for the test data. This team did not use a validation dataset and did not specify k. Another team implements K-FCV as a 10-fold cross-validation (Mowbray, et al., 2020). The process involves the data separated into 10 equal-sized folds or partitions. This study team divides each of the 10 chosen samples into 90% training data and 10% test data according to the K-FCV scheme. Neither of these studies provide a clear sample size to obtain independent of the allowable dataset. Is the provided dataset sufficient? What is the risk of not locating a failure point? These questions are not answered by these methods.

"Systematic Training and Testing for Machine Learning Using Combinatorial Interaction Testing" is an article that does answer the question of sample size and risk determination (Cody, et al., 2022). The study establishes a systematic use of combinatorial coverage techniques for selecting and characterizing test and training sets for ML models. The presented work adapts combinatorial interaction testing, which has been successfully leveraged in identifying faults in software testing, to characterize data used in machine learning. By using combinatorial coverage, the sample size is determined by needed coverage for allowable risk.

2.3 Machine Learning Algorithms

During the literature review, many reviewed articles discuss the applied ML algorithms that determine the process by which the system makes predictions under different situations or feature values. From section 1.2, Figure 6 categorizes the many different ML methods. Every ML system uses algorithms and methods that fall under at least one of these ML subsets. There are diagrams that include algorithms listed under the categories; however, these diagrams are not inclusive. There are over a hundred different algorithms today and engineers are consistently developing new algorithms to accommodate modern ML situations.

ML algorithms automate the process of the system learning a model and determine the relationship between the descriptive features and the resulting target feature in a dataset. The descriptive features are the independent variables and the target feature is the dependent variable of the model. A reliable model will capture the underlying relationship and will generalize appropriately. Many articles were reviewed to gather common and applicable algorithm types and uses. Appendix A shows the review of each reviewed algorithm.

2.4 Hyperparameter Optimization

The reviewed articles use several methods to optimize the hyperparameters of a ML model. One article uses a Grid Search (GS) and Random Search (RS) to optimize the prediction model for the shale barrier size and spatial location (Kim & Shin, 2020). The GS and RS optimize the model for the best accuracy and for parameter ranking processes. Another article uses Gradient Descent (GD) towards shallow and deep models to find the

optimal hyperparameters (Saikia, et al., 2020). GD uses calculus to minimize the objective functions. Both methods produce adequate results when employing a hyperparameter for optimization.

2.5 Verification and Validation

For the purposes of this dissertation, the researcher is using the following V&V definitions:

- The verification of the system ensures that the system adheres to the requirements and specifications developed and given by the major stakeholders.
- The validation of the system confirms that the system properly performs with representative users in the intended environment. (Kelleher, et al., 2020)

2.5.1 Verification and Validation of Machine Learning Models

The developer creates the model through ML processes, outlined in sections 2.1-2.4. During the V&V by an independent testing agency, the system model is tested for accuracy. In the following paragraphs, documented articles show the utility of selected V&V techniques for testing a model.

In one reviewed article, the authors submit that the V&V of ML models include three objectives: enrichment of information about model performance; identification of outliers and influential as well as abnormal observations; and examination of other problems relating to a model by analyzing distributions of residuals such as problems with bias, heteroscedasticity of variance, and autocorrelation of residuals (Gosiewska & Biecek, 2019). To meet these objectives, the study team uses an R software auditor

package which is a tool specifically for model diagnostics and visual verification. The tool looks at the model accuracy and reliability from many perspectives while using many different methods of analysis, not just from R-Square (R2) such as the case with only critiquing based on a linear regression model. The statistical software package R (auditor) is very extensive and commonly used in the V&V of a model.

One common basic method for model analysis is the Area Under Receiver Operating Characteristic Curve (AUC). Several reviewed articles use the AUC method. Specifically, one article uses the AUC method to identify the best fit model to predict hospital admission for older emergency room patients (Mowbray, et al., 2020). AUC is a very clear and useful method for model comparison and overall accuracy. Another article discusses the use of AUC to verify a DL algorithm that evaluates the small-bowel preparation quality in a medical trial (Nam, et al., 2021). AUC helped to determine that the algorithm provides an objective and automated cleansing score for evaluating the small bowel preparation.

R2 is also a common method for model analysis. For example, one article uses R2 to compare models for accuracy in the prediction of shale barrier size and spatial location (Kim & Shin, 2020). R2 performs good for comparisons and accuracy probability. The higher R2, the better the model. In another article, chemists use ML algorithms to predict yield during chemical reactions (Li, Zhang, Liu, Wang, et al., 2021). Chemists optimize the resulting model R2 to produce even more yield by a process known as hyper-parameter tuning. The process achieves an optimal model.

A paper written on the framework for formal verification of ML based complex system-of-systems believes that traditional V&V approaches are often inadequate to

bring in the nuances of potential emergent behavior in a system-of-systems (Raman, et al., 2021). Emergent behavior refers to the ability of a system to produce a highly-structured collective behavior over time, from the interaction of individual subsystems. The paper describes a novel approach towards application of machine learning based algorithms and formal methods for analyzing and evaluating emergent behavior of complex system-of-systems. The proposed approach involves developing a machine learning model that learns on potential negative and positive emergent behaviors, and predicts the behavior exhibited. The analyst then develops a formal verification model to assert negative emergent behavior. According to the authors, the approach is effective.

The V&V of recent ML medical products is of great importance most especially due to safety concerns. The methods used in medical V&V must be acceptable, "qualified," for regulatory submission. One paper describes a methodological framework for the credibility assessment of computational biophysical models built using mechanistic knowledge of physical and chemical phenomena (Viceconti, et al., 2021). The study team purposes to help researchers understand the required scrutiny to reach a qualified status during credibility assessment by providing a comprehensive verification, validation, and uncertainty quantification process.

To help address the quality of bioinformatic output, in a recent article, engineers present a technique for testing the implementations of machine learning classification algorithms which support such applications (Xie, et al., 2011). The approach is a technique referred to as "metamorphic testing" which shows to have effective results. This paper further confirms the effectiveness of metamorphic testing by the detection of real faults in a popular open-source classification program.

Many of the same non-ML model V&V processes work for ML model V&V. In the paper entitled, “Intelligent Black Box Verification, Validation, and Accreditation for Rotocraft Performance Modeling,” the authors provide several useful processes and tools in support of qualification assessments of performance models (McCandless, Jr., et al., 2021). This article provides details regarding developing enablers for ML application. Some of the processes present forward propagation of input uncertainty, quantification of output uncertainties, and statistical analysis of correlations among variables. These methods enable identification of key inputs and help describe the necessary resolution to understand the output dataset. The transition of the information enables system engineers and analysts to communicate these effects to developers and other stakeholders.

Model sensitivity analysis is the study of uncertainty in independent variables of a model and how these variables contribute to the model’s total uncertainty. One reviewed article implies that sensitivity analysis is becoming an integral part of modeling (Razavi, et al., 2021). The authors suggest that the scientific community does not yet realize the tremendous potential benefits of sensitivity analysis. In this article, a multidisciplinary group of researchers and practitioners revisit the status of sensitivity analysis, and outline research challenges regarding both theoretical frameworks and their applications to solve real-world problems.

2.5.2 Verification and Validation of Machine Learning Systems

For the integration and testing of the full ML system, final system performance testing must be accomplished. The testing of the system includes two types of testing: developmental and operational. Developmental testing tends to focus on verification of the system while operational testing concentrates on the validation of the system.

Verification of the system ensures that the system adheres to the requirements and specifications developed and given by the major stakeholders, while validation confirms that the system properly performs with representative users in the intended environment. Together the V&V ensures traceability to the stakeholder requirements and full system functionality in the intended environment.

As seen in the last section, some recent papers exist outlining methods and framework for the ML model V&V. However, very little exists outlining the V&V of the full ML system. In fact, this literature review finds no resource or reference that includes both white-box and black-box methods for a full V&V of a ML system. Most articles demonstrate the V&V of parts of white-box testing or the V&V of only black-box testing.

CHAPTER III

THEORY AND RESEARCH METHODOLOGY

3.1 Context and Setting

This research applies grounded theory to identify V&V methods that are most effective for testing ML systems. The researcher examines white-box and black-box methods contained in recent articles. These methods are categorized and organized to satisfy and test the two research objectives. The objectives, previously given in section 1.4 Research Objectives, are as follows:

1. To identify a general ML process from the systems engineer's perspective.
2. To generate an effective framework for the white-box and black-box V&V of a ML system by ML model type (Figure 6) for each phase of the ML process.

3.2 Research Objective 1

The first objective is for the researcher to identify a general ML process from the systems engineer's perspective. This is an important first step since a general ML process is necessary to follow-on with objective 2 of identifying a framework for testing ML systems. A tester cannot begin to test a ML system if one does not know the parts of the process to be tested. Several versions of general ML processes exist. The researcher examines each and develops the most appropriate process for the systems engineer.

3.3 Research Objective 2

Research Objective 2 is for the researcher to identify and present an effective framework for the white-box and black-box V&V of a ML system for each ML model type given in Figure 6. This research involves the review of the most recent articles concerning ML system V&V. The researcher determines the most effective white-box and black-box V&V methods by ML model type and organized the methods into a ML methods table. The effectiveness of the developed framework is validated with a case study experiment.

3.4 Case Study Experiment

The researcher uses a large open-source dataset. After acquiring the data, one begins by examining the data characteristics and ensuring it's quality. After data inspection, one uses the data to train a model while withholding 20% of the data for V&V testing. The researcher then applies a k-fold cross validation by dividing the data into k equal-sized folds, and then performs k separate evaluation experiments. The resulting model represents a fictitious ML system that requires V&V methods. The researcher then uses the developed ML V&V Framework to V&V the ML system. In conclusion, the researcher records the value of the framework application.

3.4.1 Study Framework

The chosen dataset contains 9568 records collected from a Combined Cycle Power Plant set to work with a full load for the years 2006-2011 (Tufekci, 2014). The

researcher uses the software package R to examine the data characteristics including: dimensions, dependent and independent variable data types, feature distributions, and target distribution.

After ensuring a complete and quality dataset, the researcher arbitrarily chooses an SVM algorithm to train the fictitious system model for this experiment. The algorithm chosen to develop the model is not important at this stage, since it is only used to produce a model for testing. The resulting model has four feature variables and one target variable. R holds the tuning parameter ‘sigma’ constant at 0.3689378 and uses the smallest Root Mean Square Error (RMSE) to select the optimal model out of the 10-fold resampling.

The produced model represents a fictitious ML system that predicts the target variable. The researcher then applies the developed ML V&V Framework to the V&V of the fictitious ML system. The researcher performs and analyzes all applicable methods according to the framework. As the framework explains, for white-box and black-box testing, one applies a single method of choosing for each ML process phase depending on the ML type. The most common and effective methods are highlighted in a quick-reference table to assist in choosing a method. Also, the framework explains that when possible and in ML process phases of concern (such as with algorithm choice), one should perform and compare multiple methods for an overall result.

The researcher formulates conclusions as to the validity of the ML V&V Framework. One summarizes the major points and the research findings, significance, and implications. As well as, one discusses the research shortcomings and needed future research ideas.

3.4.2 Variables

The researcher obtained the case study data from the Institute of Electrical and Electronics Engineers (IEEE) Data Port (Tufekci, 2014). The dataset contains 9568 records collected from a Combined Cycle Power Plant set to work with a full load for the years 2006-2011 (Tufekci, 2014).

Table 1. Case Study: Data Source Information.

Data Set Characteristics	Multivariate	**Number of Attributes**	4
Attribute Characteristics	Real	**Number of Target Variables**	1
Number of Instances	9568	**Missing Values**	None

Table 2 below shows the independent (features) and dependent (target) variables. The features consist of hourly average ambient variables including: temperature (Temp), Pressure, Relative Humidity (Rel_Humidity), and Exhaust Vacuum (Vacuum). The target variable is electrical energy output (Energy_Output). Averages originate from various sensors located around the plant that record the ambient variables every second. The variables are given without normalization.

Table 2. Case Study: Independent and Dependent Variables for the SVM Model.

Variable	Variable Type	Minimum	Maximum
Temperature (Temp)	Feature (independent) (numeric)	1.81	37.11°C
Pressure	Feature (independent) (numeric)	992.89 millibars	1033.30 millibars
Relative Humidity (Rel_Humidity)	Feature (independent) (numeric)	25.56%	100.16%
Exhaust Vacuum (Vacuum)	Feature (independent) (numeric)	25.36 cm Hg	81.56 cm Hg
Electrical Energy Output (Energy_Output)	Target (dependent) (numeric)	420.26 MW	495.76 MW

3.4.3 Constraints / Assumptions

The data is open source and assumed to be accurate. The provided features are assumed to be good predictors of the target variable. Other features are not available with the allotted data.

3.4.4 Data Validation

The chosen dataset represents a combined cycle power plant (CCPP) composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity generates by gas and steam turbines, which combine in one cycle, and transfer from one turbine to another. While the vacuum collects from and has effect on the ST, the other three of the ambient variables effect the GT performance as well. Sensors receive the data from the CCPP areas of interest. To choose the most accurate

model and to account for variation in the data by minimizing sample bias, R software uses the smallest RMSE out of a 10-fold resampling.

3.4.5 ML System Concept

The chosen dataset contains 9568 records collected from a Combined Cycle Power Plant set to work with a full load for the years 2006-2011 (Tufekci, 2014). The researcher partitions the data into 80% for model training and 20% for the V&V application providing a sample of 7656 trials for training and 1912 trials for V&V. To develop a sufficient model for testing, the researcher arbitrarily chose the SVM algorithm to train the data. One then uses a 10-fold cross validation method to attain the most accurate model from the data.

The resulting model has four feature variables: Temperature (Temp), Pressure, Relative Humidity (Rel_Humidity), and Exhaust Vacuum (Vacuum); and one target variable: Energy Output (Energy_Output). The produced model represents a fictitious ML system that predicts the target variable. The researcher subjects the ML system to V&V methods supported by the developed ML V&V Framework. One performs and analyzes all applicable methods according to the framework. The framework explains, for white-box and black-box testing, one applies at least one single method of choosing for each ML process phase depending on the ML type. The most common and effective methods are highlighted in a quick-reference table to assist in the choosing a method. Also, the framework explains that when possible and in ML process phases of concern (such as with algorithm choice), one should perform and compare multiple methods for an overall optimal result.

The researcher formulates conclusions as to the validity of the ML V&V Framework. One summarizes the major points and the research findings, significance, and implications. Additionally, one discusses the research shortcomings and needed future research ideas.

CHAPTER IV

RESULTS AND ANALYSIS

4.1 Research Objective 1

It is important to understand that the ML process presented here is the process from the systems engineer and analyst's perspective. There are many more steps involved for the model developer, such as one's own validation techniques executed during model development. The relevant steps for the testers are the ones that require the application of V&V methods. From study and review, the researcher devised a ML process (from the systems engineer and analyst perspective) as follows:

1. Choose possible model features and levels.
2. Apply a sampling method to the large database.
3. Use fixed percentage amounts of sampling for the training, validation, and test datasets.
4. Determine data properties: Amount, type and format, linearity, training difficulty, number of features, and interpretability.
5. Choose and apply the ML algorithms that fit the data properties.
6. Learn the training dataset with each ML algorithm.
7. Select and rank model features.
8. Optimize any hyperparameters of the model.

9. Analyze the model performance.
10. Analyze the system performance.
11. Explain the model.

The researcher summarized, combined, and included these steps in the ML V&V Framework.

4.2 Research Objective 2

After fulfilling Research Objective 1 above by determining a general ML process, the researcher summarized the process and applied it to a ML Methods Table (**Error! Reference source not found.**). The ML Methods table contains the summarized ML process steps as "ML Phases" and crosses these steps with "ML Types" from Figure 6. A ML V&V process, discussed in the next section, incorporates the ML Methods Table as illustrated by the ML V&V Framework shown in section 4.3.7. The following sections describe the aggregated product.

4.3 ML V and V Process

The researcher synthesized a ML V&V process from the literature review and the researcher's personal knowledge and experience with the testing process. The ML V&V process involves 4 main steps as follows:

- Step 1: Design and test the system's adherence to the requirements through black-box experimentation.
- Step 2: Use white-box methods to determine possible cause and effect of Step 1 failures and to enhance the system performance through examining and improving the internal structure.

- Step 3: Perform black-box system validation through functional testing.
- Step 4: Report results to the developer and other stakeholders.

Sections 4.3.1 through 4.3.4 contain activity diagrams for each of the four steps of the ML V&V process. Appendix B: Detailed ML V and V Processincludes a detailed explanation of the ML V&V process.

4.3.1 ML V and V Process: Step 1

Step 1 (Figure 7) contains the black-box verification portion of the ML V&V process. Black-box verification for a ML system is exactly as it is for a traditional software system. It begins by forming test objectives from the system requirements and applying these objectives to a verification test. The process results in identifying system failures, accuracy, and comparison metrics. Figure 7 below provides the activity diagram for Step 1.

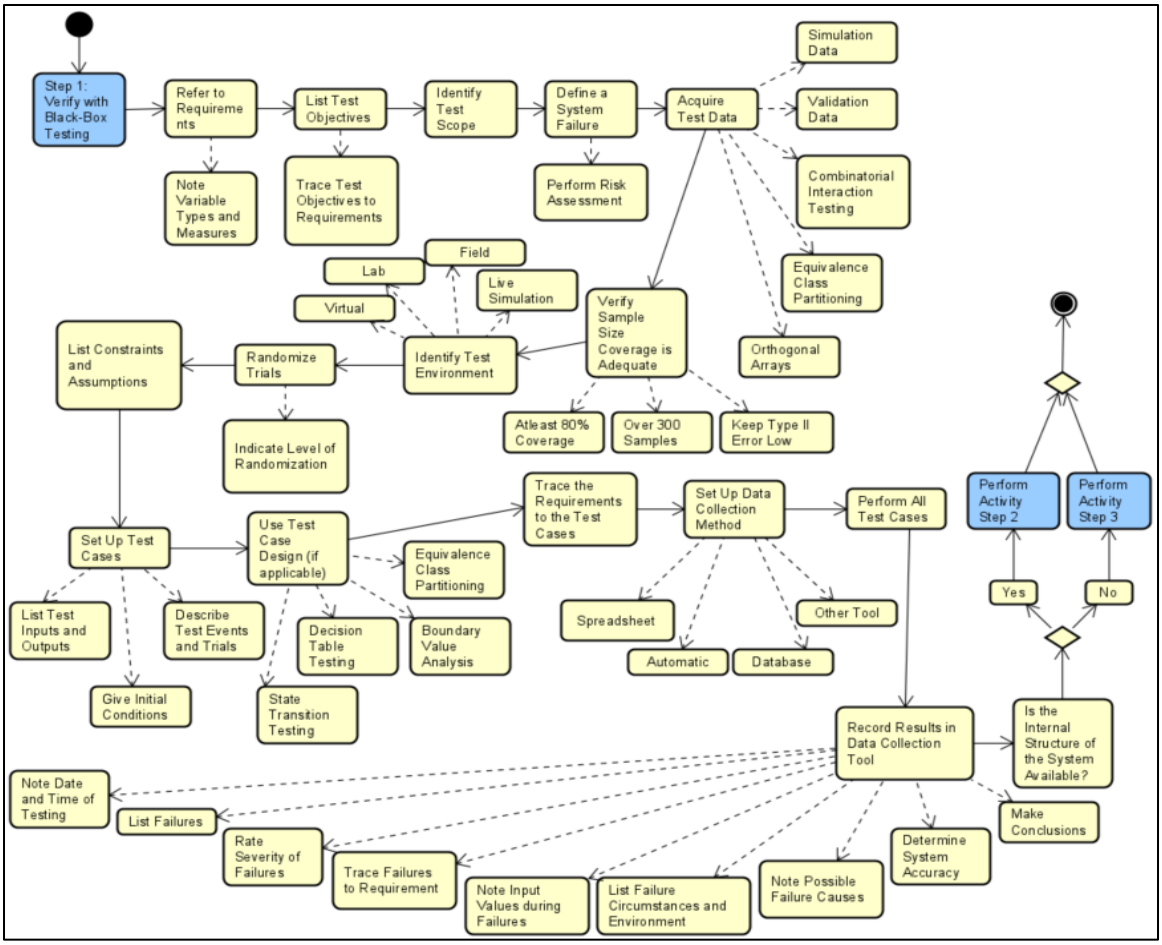

Figure 7. Activity Diagram of the ML V&V Process, Step 1.

4.3.2 ML V and V Process: Step 2

Step 2 (Figure 8) involves white-box testing for the ML system V&V. As the final conditional node depicts in Figure 7, the tester only utilizes Step 2 when having access to the internal structure of the system. Step 2 helps in the determination of cause and effect of the failures from Step 1 and works to enhance the system performance through examining and improving the internal structure. This results in a best system model from the analysis of different model types.

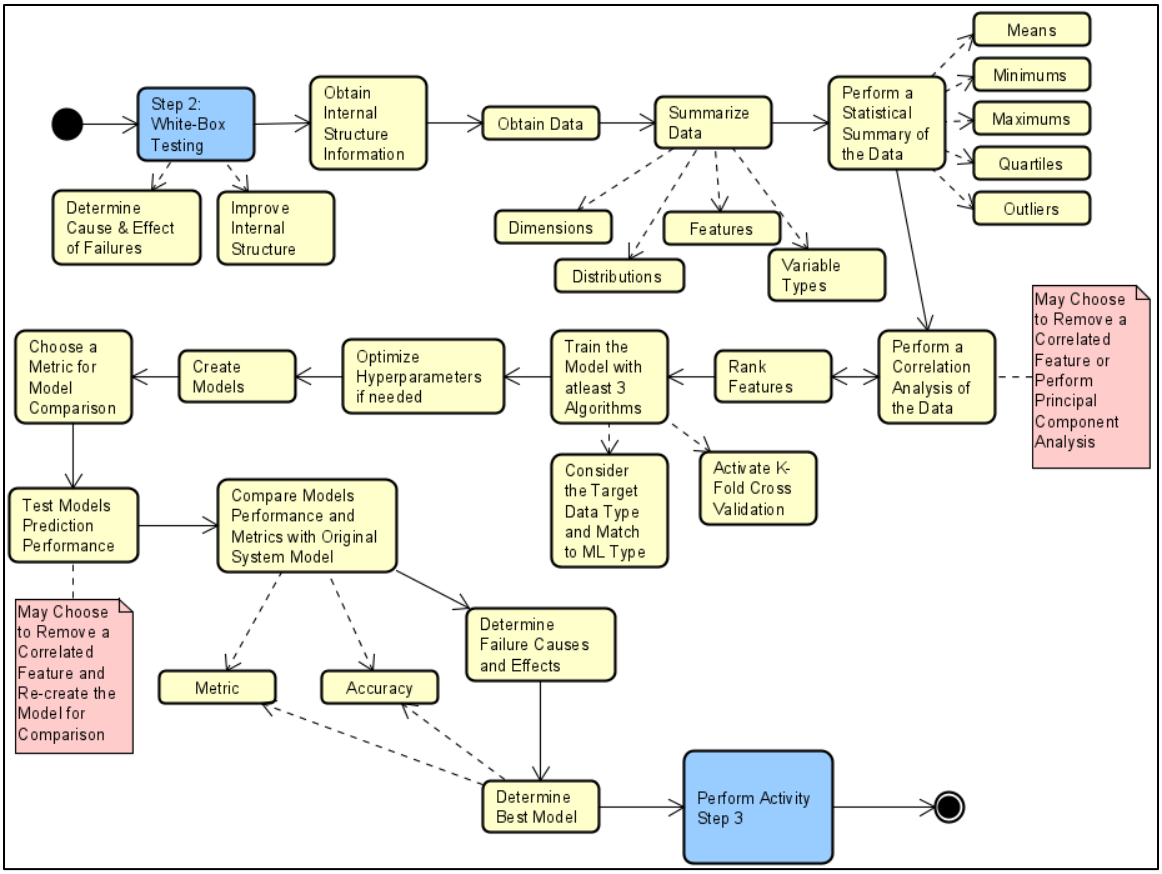

Figure 8. Activity Diagram of the ML V&V Process, Step 2.

4.3.3 ML V and V Process: Step 3

Step 3 (Figure 9) involves functional black-box validation techniques performed in the systems intended environment by representative users. Functional testing validates the ML system against the functional requirements and specifications. The tester records any failures and proceeds to Step 4.

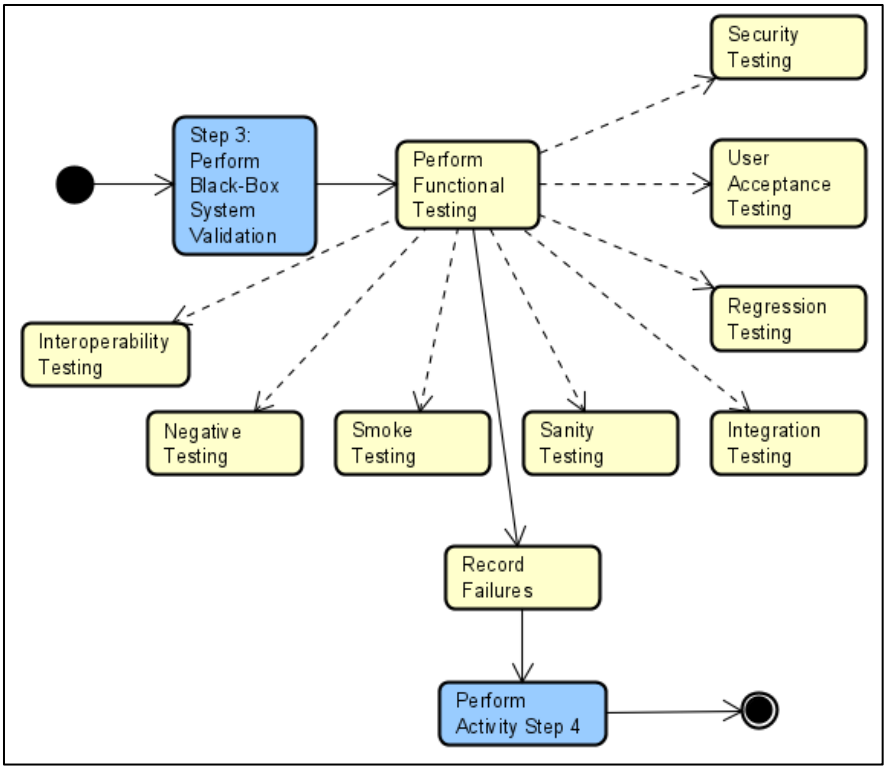

Figure 9. Activity Diagram of the ML V&V Process, Step 3

4.3.4 ML V and V Process: Step 4

Step 4 (Figure 10) represents the information that should be reported based on the results. In other words, submit Step 1 and Step 3 results if the tester determined that the developer's original model was the best model according to the comparison metric defined in the ML V&V Process. Conversely, if a better model was found, submit Steps 1-3 results.

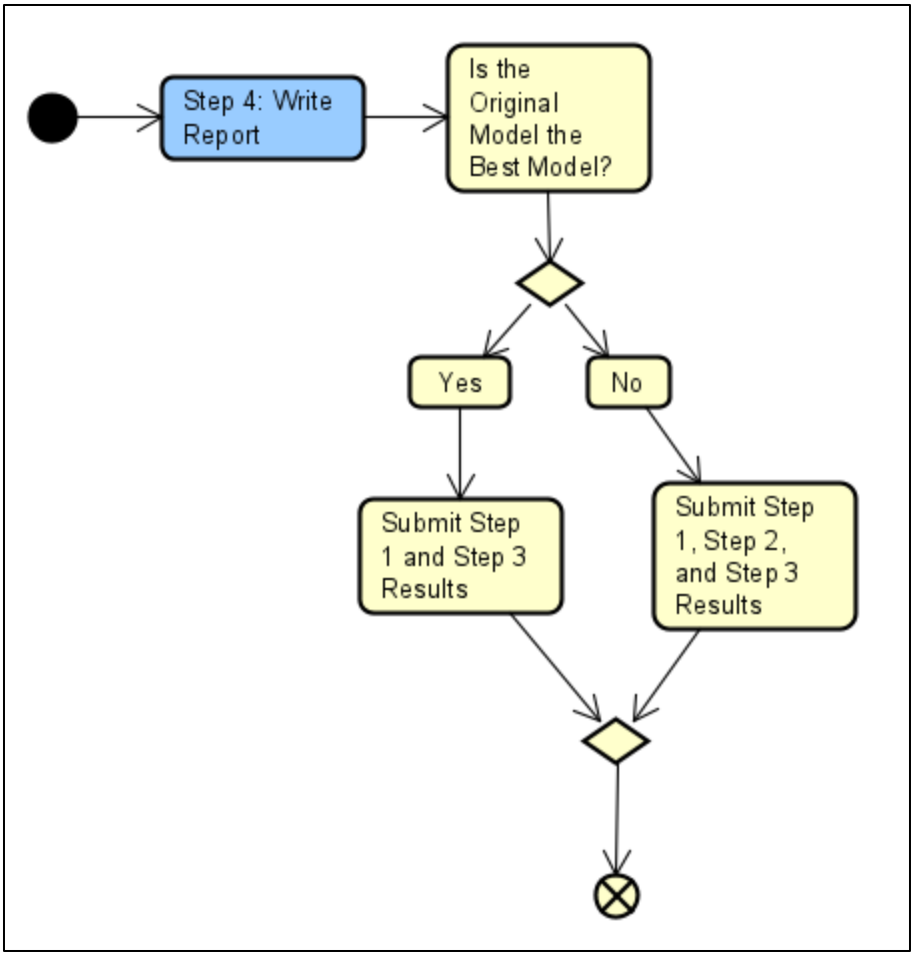

Figure 10. Activity Diagram of the ML V&V Process, Step 4

4.3.5 ML Methods Table

The ML V&V Process discussed above in Section 4.3, interacts with the ML Methods Table shown in **Error! Reference source not found.** below. The process r equires the application of ML methods found in the table. Notice that the table is presented via “ML Phase” and “ML Type.” If one can only test black-box methods, then refer to the next section 4.3.6.

Table 3. White-box and Black-box ML Methods Table.

		White-Box Testing					**Black-Box Testing**	
	ML PHASE	**Sampling Algorithms/ Risk**	**ML Algorithm**	**Feature Selection Algorithms/ Ranking**	**Hyper-parameter Optimization**	**Model Validation**	**System Verification (Performance)**	**System Validation**
ML TYPE	**SUB-TYPE**							
Inductive Learning	**Decision Tree**	K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**C5.0, ID3**, **RF, GBM**, **CART**, **CHAID,** M5, RLT, Decision Stump, Conditional Inference Tree	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

Table 3 Continued.

Inductive Learning	**Rule**	K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**CN2**, Charade, Rulex, Progol,	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing
Instance-Based		K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**KNN, LVQ, SOM, SVM,** LWL, RBF	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

Table 3 Continued.

Genetic Algorithm		K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**GA**	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing
Neural Networks	**Supervised**	K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**SLP, MLP, BP, GD, OSLR, SLR, MLR, MR, LR,** FFNN, ELM, EM, FNI, MNN, PNN, EO	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

Table 3 Continued.

Neural Networks	**Unsupervised**	K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**CNN, RNN, LSTM, DBM, DBN, R-CNN,** SPM, MRH, k-Means Clustering	Drop incomplete features, features with high multicollinearity, or features with (near-)zero variance, JMV, MUFS	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing
Neural Networks	**Reinforced**	K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**PG, HER, Q-learning, A3C, DQN, TRPO, C51, PPO, I2A,** SARSA, DDPG	MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

Table 3 Continued.

Bayesian Approach		K-FCV, Hold Out, LOCV, ECP, BVA, Boot-strapping, Out of Time, CIT, Orthogonal Arrays	**BN, Naïve Bayes, Gaussian Naïve Bayes, BBN,** HMM, CRF	BMS, MDI, MDA, AHP, Fuzzy Sets, PROMETHEE, Chi-Square Contingency Tables, Fischer's Score, Correlation Coefficient, Variance Threshold, MAD, Dispersion Ratio	GD, GS, RS	R2, RMSE, AUC, MSA, Invariant Test, Min Functionality Test, Directional Test, PCA	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

4.3.6 Black-box ML Methods Table

When the ML V&V involves only black-box testing because of the lack of system internal structure knowledge, the researcher can reduce **Error! Reference source not found.** above and produce **Error! Reference source not found.** shown below. As described by the ML V&V Process Activity Diagrams, black-box testing requires only Steps 1, 3, and 4. These steps require the application of ML methods given in **Error! Reference source not found.**.

Table 4. Black-box ML Methods Table.

	ML Phase	
ML Type	**System Verification (Performance)**	**System Validation**
All ML Types	ECP, BVA, Requirements Traceability, Decision Table Testing, State Transition Testing	Functional Testing (unit, component, negative, smoke, sanity, integration, regression), Human Factors Testing, UAT, Security Testing

4.3.7 ML V and V Framework

The ML V&V Framework shown in Figure 11 below, illustrates the relationship between the ML V&V Process (Section 4.3 ML V and V Process), the Process Steps (Sections 4.3.1 - 4.3.4), and the ML Methods Table (Table 3). The researcher posted a detailed ML V&V Process description, containing ML Methods Table references, Appendix B: Detailed ML V&V Process. The detailed stepwise procedures allow the tester to work through the entire ML Framework with ease.

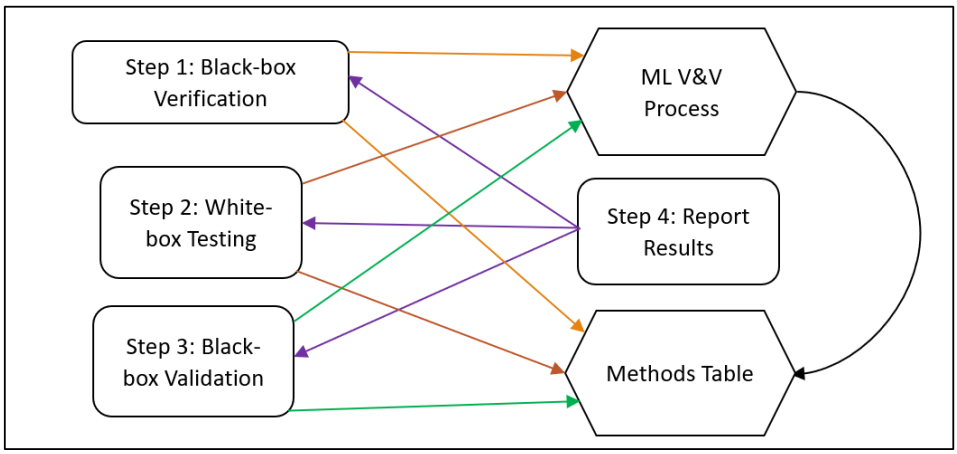

Figure 11. ML V&V Framework.

4.4 Case Study Experiment

From the case study dataset, a resulting SVM model has four feature variables: Temperature (Temp), Pressure, Relative Humidity (Rel_Humidity), and Exhaust Vacuum (Vacuum); and one target variable: Energy Output (Energy_Output) (Table 2). All independent variables are numeric (Table 1). The target variable, energy output, is a continuous numeric variable. The produced model represents a fictitious ML system that predicts energy output. To facilitate the discussion, one names the fictitious system "Sys A." The researcher now enters into the role of a tester and applies Section 4.3.1 to Sys A in order to validate the quality of the developed framework.

4.4.1. Framework Application: Step 1

The first step in the developed framework is to design and test a black-box experiment. The main objective of step one and the V&V in general is to discover and locate failures in the system output. The goal of the internal sub-processes of Step 1 is to

lead the tester to optimal failure discovery by providing the most effective ML V&V methods available. The scope of the black-box testing portion (Step 1) only considers inputs in the given feature level ranges (Table 2). Other testing methods; such as negative, smoke, sanity, integration, and regression are covered in Step 3 of the ML V&V.

Executing Step 1 of the ML V&V Framework, the tester started by considering the requirements of Sys A. The main requirement is that Sys A predict the target variable, energy output, given the 4 feature values in a trial or case. Since Sys A is a fictitious system and a requirement document really does not exist, the researcher defined Sys A as an alarm system that alerts system personnel to an issue that can cause minimal damage in a facility. The alarm system alerts when energy output is at certain values. The alarm alerts about two times a week. The researcher also defines a failure of Sys A. To define the failure, the researcher viewed a preliminary run of the data for accuracy via the root mean square error (RMSE). This gave the researcher an estimate as to how close to allow error of prediction and still get some failures for testing. The researcher defined a failure of Sys A as a prediction of the energy output that has an error greater than 11 units.

Guided by the framework, the tester performed a risk assessment at this point. The assessment considers different types of failures related to the system and their associated risk according to the likelihood of occurrence and the extent of damage or impact. A generic risk matrix is shown in Figure 12. Considering Sys A and the possible failure defined above, the tester placed the risk at a medium level, with high frequency and low severity.

Colored Cells are the Risk Categories	Low Risk	Medium Risk	High Risk

Frequency of Scenario	Severity of Consequences		
	Low Severity	Medium Severity	High Severity
High Frequency	Medium	High	High
Medium Frequency	Low	Medium	High
Low Frequency	Low	Low	Medium

Figure 12. Generic Risk Assessment Matrix. (Vatanpour & Hrudey, 2015)

The dataset of Sys A contained 9568 records collected from a Combined Cycle Power Plant set to work with a full load for the years 2006-2011 (Tufekci, 2014). Before developing the SVM model for Sys A, the tester partitioned the data into 80% for model training and a hold-out test set of 20% for the V&V application, providing a sample of 1912 trials for V&V. In the case study, one cannot determine the sample space coverage, however a sample size of 1912 is typically considered a large sample size for most sample spaces. According to textbook recommendation, a sample size of over 300 is appropriate (Kelleher, et al., 2020).

The testing environment was a computer lab setting with the given data used to develop the fictitious system, Sys A. The tester stored the data in Excel and used the programming statistical language R to develop the model and to perform the V&V of the ML system.

The test data taken from the full dataset was completely random. R software was used to partition the data into a random 80:20 split. There were no missing data and every data record from the test set was used in testing.

The tester assumed that the case study data was accurate. One assumed that the provided features were good predictors of the target variable. The case study is constrained with the given dataset features and target variable.

The tester used R software to run, analyze, display, and record the data. The results were collected in an R output window, then copied and pasted into a Microsoft Word document.

A test case (trial) consisted of one combination of features from the test data. There were 1912 test cases. The tester applied each test case to the Sys A SVM model and recorded each predicted response value using R software.

According to the boxplots in Figure 13 below, the first three features; the tester considered Temp, Vacuum, and Pressure relatively normal in distribution. The fourth feature showed skewness to the right some, but could also pass for relatively normal in distribution. Since relatively normal, the data spread in the sample space was adequately covered throughout with the majority of samples in the center. Since all features were continuous numerical and there was a relatively good spread, then a natural Equivalence Class Partitioning (ECP) occurred. The only difference was that the research tested

multiple samples from each class of the spread. This case study only had the test set of data available. It was not possible for the tester to test additional trials that were not included in the original dataset. Therefore, a full Boundary Value Analysis (BVA) was not possible. However, according to the boxplot of the validation data, some of the boundaries were included in testing. For example, the boundaries were tested where outliers are present, such as with features Pressure and Rel_Humidity. Decision Table Testing was not applicable in this test since there were no actions given for the conditions and the features were not categorical. State Transition Testing was not applicable since the tester did not have system transitions available. A test case (trial) consisted of one combination of features from the test data. There were 1912 test cases. The tester applied each test case to the Sys A SVM model and recorded each predicted response value using R software.

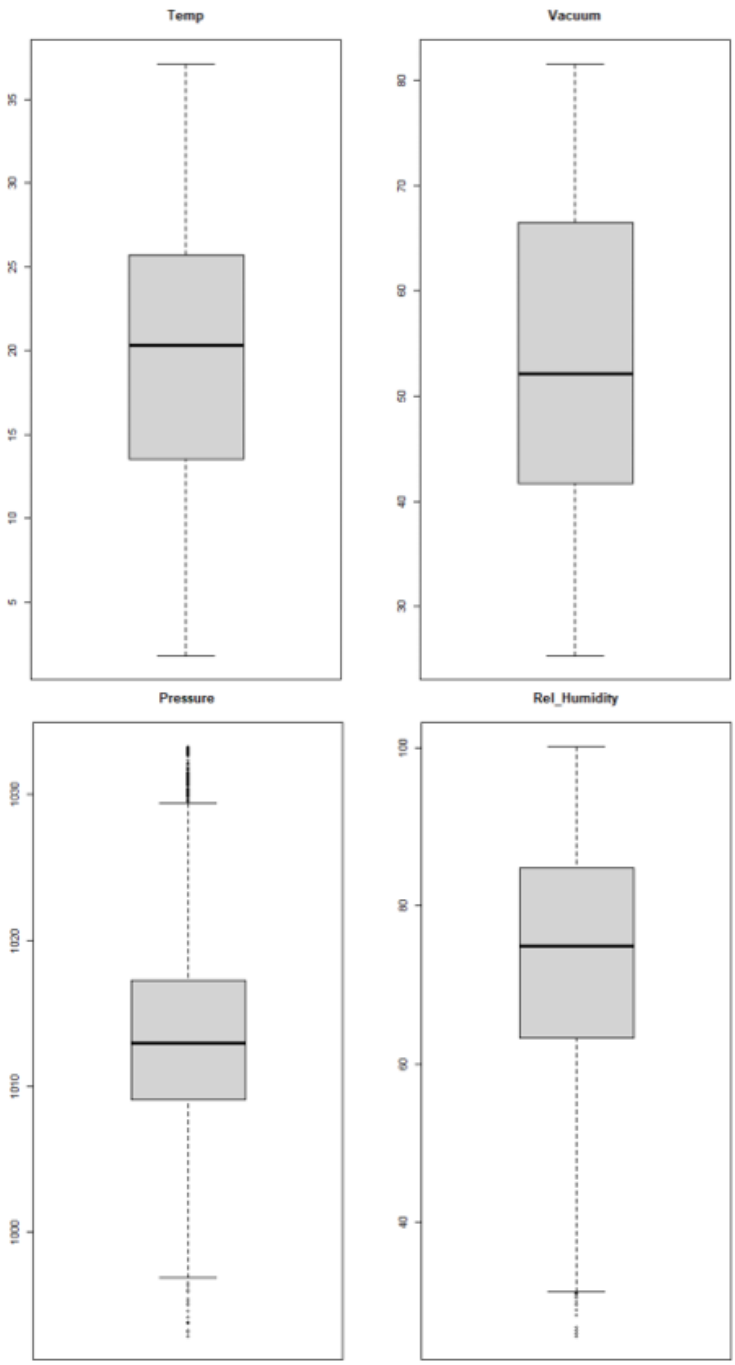

Figure 13. Boxplots of Validation Data Features.

The tester again confirmed that the test cases were traced to the requirement. In this study, the requirement was for Sys A to predict within 11 units of the actual energy

output. The tester resolved that all possible failures from the sample space had an adequate chance of selection for testing since sampling was random and coverage adequate.

Continuing to follow the guidance of the framework, the tester input the validation dataset into the Sys A model using R software. Predicted values were captured and compared to the actual values given in the dataset. The error was calculated as the difference between the actual value and the predicted value. A summary of the results is given in Table 5 below. The RMSE was 4.028321 with an accuracy of 98.43%. The following formulas were used to calculate the RMSE and accuracy of the model predictions:

$$RMSE = \sqrt{mean(error^2)}$$

and

$$\text{accuracy} = \frac{\text{\# of trials} - \text{\#of failures}}{\text{\#of trials}} * 100.$$

The tester used the RMSE and the accuracy to compare to other models developed during the white-box testing to be performed in Step 2 of the ML V&V Framework.

Table 5. Summary of Target Prediction Results.

Actual	**Predicted**	**Error**
Minimum: 425.1	Minimum: 428.8	Minimum: -26.70844
1st Quartile:439.8	1st Quartile:439.9	1st Quartile: -2.75067
Median:451.6	Median:451.7	Median: -0.09926
Mean:454.2	Mean:454.5	Mean: -0.30812
3rd Quartile:468.4	3rd Quartile:468.6	3rd Quartile: 2.12278
Maximum:495.4	Maximum:495.2	Maximum:15.81830

Figure 14 below shows the predicted values versus the actual values for the target variable, Energy_Output. A boxplot showing the prediction error is also shown below in Figure 15. All outliers in the boxplot were counted as failures as well as any trial with an energy output greater than 11 units in error. 30 trials tested as failures.

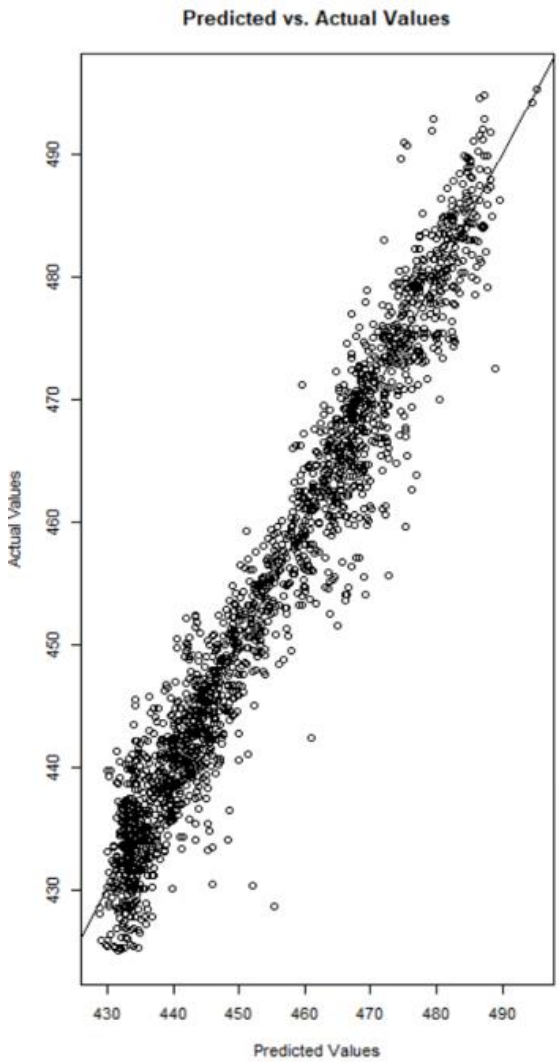

Figure 14. Predicted Values VS. Actual Values for Sys A.

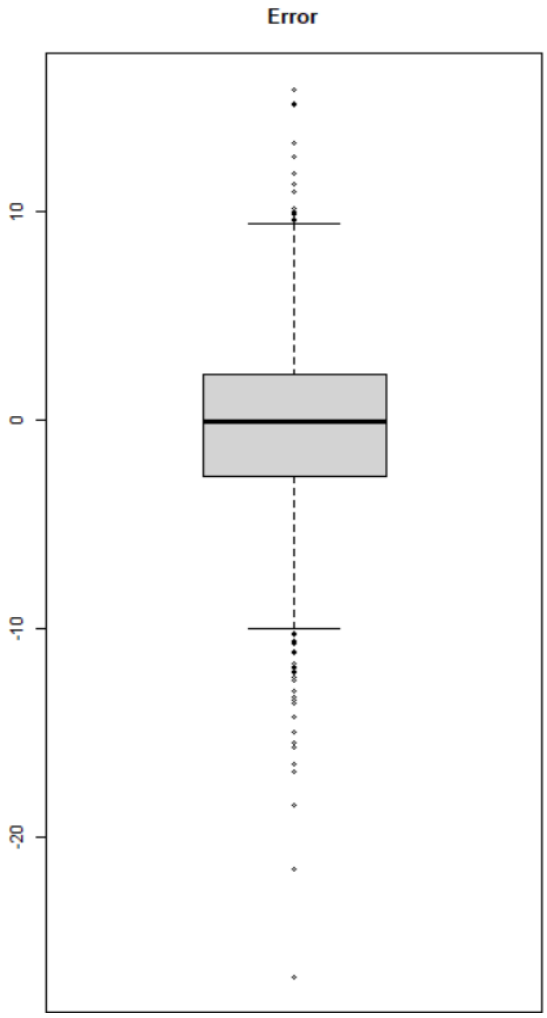

Figure 15. Boxplot of Prediction Error.

Boxplots provide distribution by feature values. Figure 16 below shows the plots for the distributions of the failure cases by feature. Here one determines where failures tend to occur. For example; the Temp tends to have more failures at lower Temp values; whereas, the Rel_Humidity tends to have more failures at higher Rel_Humidity values. At this point, the tester could not specifically determine the cause of the failures. Since the tester has access to the internal structure of Sys A, Step 2 of the ML V&V Framework

provided more insight into cause and effect for the failures and allowed the tester to work towards improving the system model.

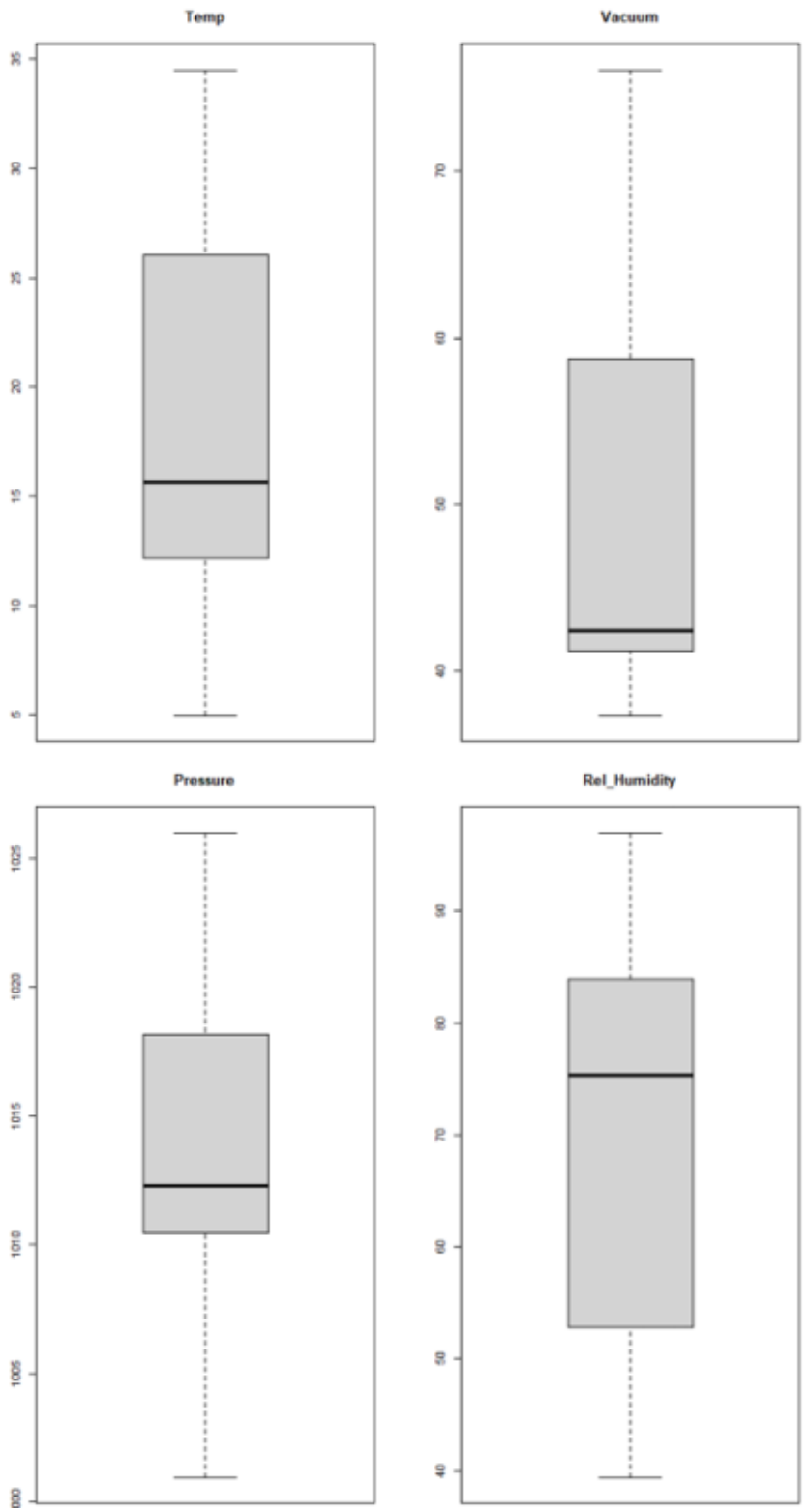

Figure 16. Boxplots of Failure Cases.

4.4.2 Framework Application: Step 2

From the ML V&V Framework, Step 2 involved white-box methods to determine possible cause and effect of the failures found in Step 1. White-box testing also provided the tester an opportunity to improve the system performance with a more accurate model. The following paragraphs explain the methods used by the tester in Step 2.

In this case study, the researcher was the developer by obtaining the data, partitioning the data, arbitrarily applying the data to an SVM algorithm, and training the model with the data to create Sys A. Therefore, since the tester was also the researcher, the internal structure of Sys A was available for white-box testing. The test dataset used in Step 1 was a clean dataset and was used in the white-box testing as well.

The tester tested for correlation between two or more feature variables using the significance level and correlation coefficient method listed in Table 3, column 3. Before testing for correlation, the tester checked the correlation test assumptions. First, each feature was tested for normality using the Shapiro-Wilk Normality Test. The test showed that every feature was approximately normal with a high probability (Appendix C: Case Study R Code). Next, each combination of feature pairs was checked for linearity. A scatterplot matrix, shown in Figure 17 below, revealed that a couple of combinations were not clearly linear (i.e., Pressure vs. Rel_Humidity and Vacuum vs. Rel_Humidity). For the features that did not show clear linearity, the tester applied the Kendall Rank Correlation Test which does not require met assumptions. The tester used Pearson Correlation Test for the other feature pairs that did show clear linearity.

Figure 18 and Table 6 below, show the correlation results for the feature pairs. According to the results, Rel_Humidity showed practically no correlation with Pressure and very

little with the other two features using the Kendall Correlation Test. This finding was consistent with the linearity test results from Figure 17 that showed very little linear relationship between Rel_Humidity and the other features. The Pearson Correlation Test showed that the other three features: Temp, Vacuum, and Pressure; were significantly correlated and did contain redundancies that could affect the model results.

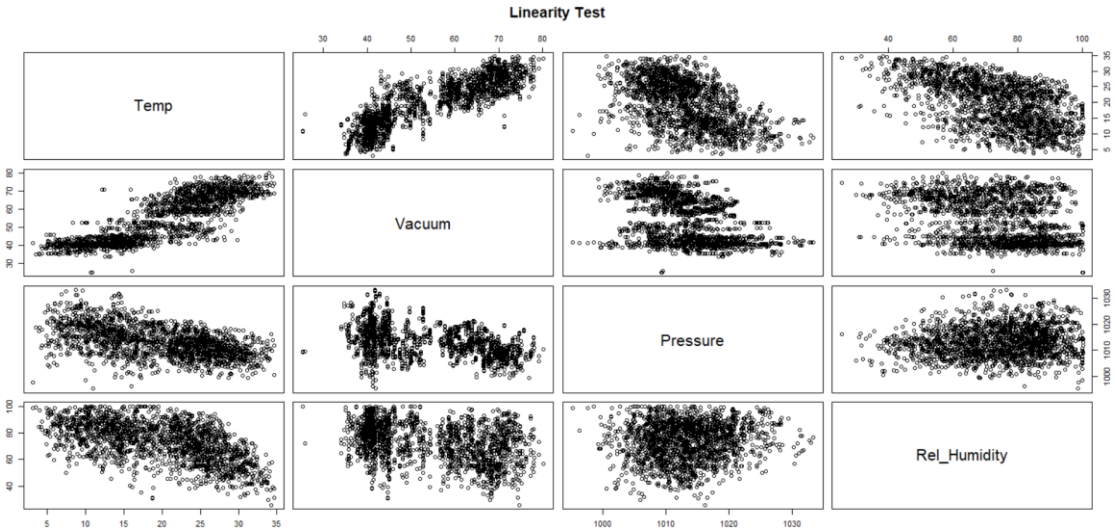

Figure 17. Test for Linearity between Feature Pairs.

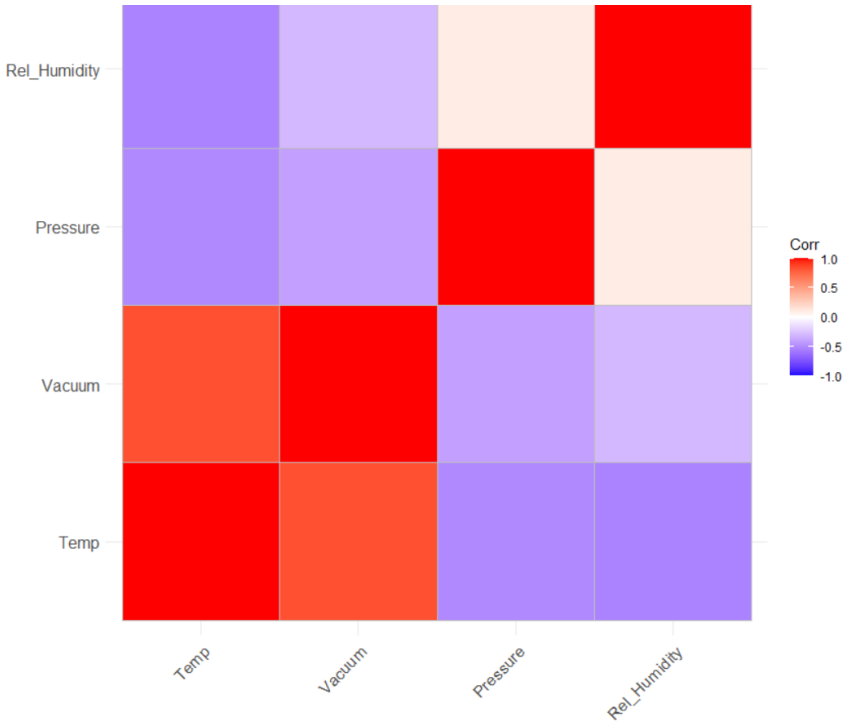

Figure 18. Correlation Matrix of Features.

Table 6. Feature Correlation Pairs.

Rel_Humidity	0.3887	0.2280	-0.0646	
Pressure	0.5123	0.4167		-0.0646
Vacuum	-0.8511		0.4167	0.2280
Temp		-0.8511	0.5123	0.3887
	Temp	**Vacuum**	**Pressure**	**Rel_Humidity**

Even with the given correlations, the tester used all four independent variables to train models and make predictions. After having reviewed the results, the tester revisited the decision of whether to remove any features. Since there were no operative decision makers at this time, MCDA methods were not considered.

Since the target variable is continuous numeric, the tester only trained the model with appropriate algorithms. One applied the following algorithms (Table 3, column 2): MLR and RF with k-fold cross validation, $k = 10$. The tester compared the model results from these two algorithms to the SVM model results from ML V&V Step 1,Table 7.

Table 7. Case Study: Model Comparison.

Model	RMSE	Failures	Accuracy
SVM	4.0283	30	98.43
MLR	4.4363	23	98.80
RF	3.1617	8	99.58

The RF model performs better than SVM or MLR with an accuracy of 99.58%. The tester revisited the feature selection section of the ML V&V framework in order to check the model accuracy without a correlated feature. One removed Vacuum since it showed a high correlation with Temp. The new model did not show improvement; in fact, without the variable Vacuum, the accuracy dropped to 98.95% with an RMSE of 4.20. Since dropping the variable Vacuum did not improve the model, the researcher added it back into the model and retained all of the original four features: Temp, Vacuum, Pressure, and Rel_Humidity. These features utilized in the RF model delivers the best results.

By comparing the results from the SVM model and the new RF model, the tester determined that the failures were not located in the same areas of the sample space. Therefore, the cause of the initial failures was due to a poor fitting model, in this case the SVM. The RF model was superior with an almost perfect fit.

4.4.3. Framework Application: Step 3

This section of testing was not possible to perform with a fictitious system, however one thought through the processes and how they might have been achieved. The tester would have performed this section of testing in an intended operational environment with actual Sys A users. One would utilize the test cases in this situation to determine proper system performance and fit. One performs each test from Table 3, column 7.

4.4.4. Framework Application: Step 4

In this section, the tester creates a report of the test results. In this scenario, one reports to the developer and other stakeholders all results from Steps 1-3. One explains the poor performance of the SVM model and the improved performance of the RF model. One lists all failures in the report along with visual charts and line graphs depicting the results. This provides the developer with an overall picture of the systems development and operational performance.

CHAPTER V

CONCLUSIONS

5.1 Conclusions

The researcher developed a ML process from the viewpoint of the systems engineer and analyst. The process assisted in the further development of a ML V&V Framework. The framework included four general steps for the accomplishment of the complete ML V&V. The first step involved the full black-box performance testing of a system. In the demonstration of the case study, the researcher followed Step 1 for the fictitious system, Sys A.

In Step 1 of the framework, for the case study, the researcher performed the analysis as a tester. The tester followed the guidance of the framework resulting in analysis of the given SVM model provided by the system. The tester located and charted the system failures. The framework, Step 1, was an excellent guiding source for the process. The tester reformed the framework some along the way to make points clearer or to add missing methods, and the process became more robust as a result. By the end of Step 1, the tester was able to find every needed method for the complete black-box analysis of the system performance.

Step 2 of the framework was the white-box testing portion of the ML V&V. The tester followed each suggested method and, by doing so, subjected the data to other

model types, in this case, being MLR and RF models. The tester determined that RF was the best model choice, according to the comparison metric defined in the ML V&V Process, of the three tested models and performed best while keeping all four features given. One also found that the cause of the failures in Step 1 was the inaccurate model results of the SVM model. The system performed much better with the RF model. Again, the framework was very helpful in executing a complete white-box analysis and obtaining model results for the best performance of the system.

Step 3 of the ML V&V framework allowed the tester to look at the operational performance of the system in its intended environment and with expected users of the system. Again, the framework provided a complete analysis guide for results. The fictious system was not exposed to an actual operational test event because of constraints, however, the tester imagined each method applied to the system. This portion of the testing V&V would allow the tester to identify failures such as incompatibility with the system environment. Since the case study system is fictitious, one could not necessarily gauge actual performance, but one could imagine the facilitation of the framework application.

Step 4, the report, was also found to be sufficiently covered by the ML V&V Framework. This step instructed the tester to report the findings in each step according to the performance of the original system model. The guidance of Step 4 in the framework was good direction.

The case study verification found that the developed ML V&V Framework was an effective tool for ML V&V. The methods listed in the associated framework table were easily accessible and applied by online search of websites or library articles. To

help further with this, the researcher also added a ‘List of Terms’ for a quick reference to the method definition. Having the methods broken down by ML type, based on features and target data types, was also valuable in allowing the tester to quickly apply the methods.

5.2 Significance and Implications

The developed ML V&V Framework is the first framework found that contains the guidance through white-box and black-box ML V&V. The framework provides a detailed process guide and a methods table. The ML Methods Table contains significant and effective methods for the following ML phases: feature selection and ranking, sampling algorithms with risk, ML algorithms, hyperparameter optimization, model validation, system performance, and system validation. The table is also broken down by ML type according to the data features and target types. Along with the framework, ML V&V process, and the table; a ‘List of Terms’ is also provided as a quick lookup of the defined methods. With these included elements, this framework provides systems engineers and analysts a complete guide for ML V&V white-box and black-box system testing.

By having clear guidance found in the developed framework, the efficiency and effectiveness of the machine learning system testing improve. The researcher developed the framework based on the most informed methods. Applying the framework ensures reliable and accurate results. This reduces failures in systems and improves system reliability and safety.

5.3 Study Weaknesses

ML is a growing field of study that is changing at a high rate. Despite this, the developed framework does not change so often, however, the methods listed in the supplementary table do change or at least need updating every few years. It would be prudent for the systems engineer or analyst to stay current on methods and to update the table when needed.

There are many methods and algorithms, including hybrid algorithms. There are too many methods and algorithms to list all in the framework table. Specialized algorithms that are not listed in the table may apply and may show superior results.

The case study would have been stronger had someone other than the researcher performed the validation of the framework. This was not possible in this occurrence because of the lack of an available testing expert.

5.4 Future Research

There should be additional research over time for adding to the original guidance and table methods. If available, an experienced tester other than the researcher should perform V&V on the new ML framework. Another option would be to have several participants verify and validate the ML framework. The researcher could administer a survey to the participants to capture their viewpoints on the application worth. The study could include multiple case studies of different ML types. The ML framework could also be validated with an AI natural language processing tool, such as ChatGPT.

Further research could extend the framework application to assist in other ways. For example, the ML V&V Process combined with the ML Methods Table can form as

an indicator to personnel management allowing one to understand what type of skills are required to administer a ML framework. The combination could also contribute to a training development plan for systems engineers and analysts who test ML systems. With additional effort, one could possibly apply the framework to the development of a tool that captures unknown features to predict needed ML methods for a given study.

REFERENCES

Abbasi, M., Shahraki, A., Piran, M. J. & Taherkordi, A., 2021. Deep Reinforcement Learning for QoS provisioning at the MAC layer: A Survey. Engineering Applications of Artificial Intelligence, Volume 102, p. 104234.

Anissa, F., 2017. A multicriteria intelligence aid methodology using MCOA, artificial intelligence and fuzzy sets theory. Mathematical Problems in Engineering, Oct.pp. 1-12.

Anon., 2023. The Institute for Statistics Education. [Online]
Available at: https://www.statistics.com/glossary/chaid/

Bhattacharya, S. & Mishra, S., 2018. Applications of machine learning for facies and fracture prediction using Bayesian Network Theory and Random Forest: Case studies from the Appalachian Basin. Journal of Petroleum Science and Engineering, Volume 170, pp. 1005-1017.

Brownlee, J., 2020. Hyperparameter Optimization with Random Search and Grid Search. [Online]
Available at: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Brownlee, J., 2020. Learning Vector Quantization for Machine Learning. [Online]
Available at: https://machinelearningmastery.com/learning-vector-quantization-for-machine-learning/#:~:text=The%20Learning%20Vector%20Quantization%20algorithm,those%20instances%20should%20look%20like.

Celestial Systems, 2023. Software testing: Boundary value analysis & equivalence partitioning. [Online]
Available at: https://www.celestialsys.com/blog/software-testing-boundary-value-analysis-equivalence-partitioning

Chen, W., Zhao, L., Kang, Q. & Di, F., 2020. Systematizing heteorogeneous expert knowledge, scenarios and goals via a goal-reasoning artificial intelligence agent for democratic urban land use planning. Cities, Mar, Volume 101, pp. 1-15.

Clark, P. & Niblett, T., 1989. The CN2 induction algorithm. Machine Learn, pp. 261-283.

Cody, T., Lanus, E., Doyle, D. D. & Freeman, L., 2022. Systematic training and testing for machine learning using combinatorial interaction testing. Valencia, Spain, IEEE, pp. 102-109.

Datacamp, 2023. Principal Component Analysis in R Tutorial. [Online] Available at: https://www.datacamp.com/tutorial/pca-analysis-r

Dayo-Olupona, O., Gene, B. & Onifade, M., 2020. Technology adoption in mining: A multi-criteria method to select emerging technology in surface mines. Resources Policy, Dec, Volume 69, pp. 1-16.

De-Yu, C., 2021. Deep Q-Network, with PyTorch. Towards Data Science.

Elazar, E., 2023. User acceptance testing (UAT) process explained. [Online] Available at: https://www.panaya.com/blog/testing/what-is-uat-testing/

Fang, S.-G., Huang, D., Wang, C.-D. & Tang, Y., 2022. Joint Multi-view Unsupervised Feature Selection and Graph Learning. Arxiv, 18 Apr.

Fisher, M. S., 2007. Systems V&V. In: Software verification and validation: An engineering and scientific approach. Boston(MA): Springer US, pp. 155-162.

Gajewski, J. & Valis, D., 2020. Verification of the technical equipment degradation method using Hybrid Reinforcement Learning Trees - Artificial Neural Network System. Tribology International, Volume 153, pp. 1-11.

Ganascia, J.-G., 1987. Charade: a rule system learning system. Orsay, France, ResearchGate.

Gautam, A., 2023. SARSA Reinforcement Learning Algorithm: A Guide. [Online] Available at: https://builtin.com/machine-learning/sarsa

Gosiewska, A. & Biecek, P., 2019. An R Package for model-agnostic visual validation and diagnostics. R Journal, Dec, 9(2), pp. 85-98.

Guimaraes, V. & Costa, V. S., 2022. Online learning of logic based neural network structures. Switzerland, Springer, Cham, pp. 140-155.

Gupta, A., 2023. Feature Selection Techniques in Machine Learning. [Online] Available at: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/#:~:text=The%20variance%20threshold%20is%20a,same%20value%20in%20all%20samples.

Hamilton, T., 2023. What is security testing? Example. [Online] Available at: https://www.guru99.com/what-is-security-testing.html

Harper, K., 2016. 5 Considerations for human factors testing. [Online] Available at: https://www.mpo-mag.com/contents/view_online-exclusives/2016-09-13/5-considerations-for-human-factors-testing/#:~:text=Human%20factors%20testing%20looks%20at,error%20or%20difficulty%20in%20use.

Hart, B., 2018. Some thoughts on requirements. [Online] Available at: https://psyche.asu.edu/2018/06/06/some-thoughts-on-requirements/

Hoang, N.-D. & Bui, D. T., 2017. Chapter 18: Slope stability evaluation using radial basis function neural network, least squares support vector machines, and extreme learning machine. In: Handbook of neural computation. Bihar, India: Academic Press, pp. 333-344.

Hothorn, T., Hornik, K. & Zeileis, A., 2012. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics, 01 Jan, 15(3), pp. 651-674.

Huang, Y. et al., 2022. Complementary and Consensus Learning-based Incomplete Multi-view Unsupervised Feature Selection. arXiv, 20 Aug.

Jarosz, W. et al., 2019. Orthogonal Array Sampling for Monte Carlo Rendering. Computer Graphics Forum, Volume 38, pp. 135-147.

Jordan, J., 2020. Effective testing for machine learning systems. [Online] Available at: https://www.jeremyjordan.me/testing-ml/

Joseph, T., 2021. A complete guide to negative testing in software testing. [Online] Available at: https://blog.qasource.com/a-complete-guide-to-negative-testing-in-software-testing

Karaca, Y., Baleanu, D. & Karabudak, R., 2022. Hidden Markov Model and multifractal method-based predictive quantization complexity models vis-á-vis the differential

prognosis and differentiation of Multiple Sclerosis' subgroups. Knowledge-Based Systems, Volume 246, p. 108694.

KARDOME, 2022. Voice Recognition. [Art] (KARDOME Technology LTD).

Katalon, 2023. What is functional testing? Definition, types & examples. [Online] Available at: https://katalon.com/resources-center/blog/functional-testing

Kelleher, J. D., Namee, B. M. & D'Arcy, A., 2020. Machine learning for predictive data analytics: algorithms, worked examples, and case studies. 2nd ed. Cambridge(Massachusetts): The MIT Press.

Kim, M. & Shin, H., 2020. Machine learning-based prediction of the shale barrier size and spatial location using key features of SAGD production curves. Journal of Petroleum Science and Engineering, Volume 191, pp. 1-20.

Kreimel, P. et al., 2020. Anomaly detection in substation networks. Journal of Information Security and Applications, Volume 54, pp. 1-11.

Kumar, P., 2021. Decision coverage testing and decision table testing. [Online] Available at: https://www.h2kinfosys.com/blog/decision-coverage-testing-and-decision-table-testing/

Ladds, M. A. et al., 2017. Super machine learning: Improving accuracy and reducing variance of behavior classification from accelerometry. Animal Biotelemetry, Volume 5, pp. 1-9.

Lau, G., 2022. Machine learning testing for beginners - All in one guide. [Online] Available at: https://blog.testproject.io/2022/01/17/machine-learning-testing-for-beginners-the-all-in-one-guide/

LeCun, Y., 2018. The power and limits of deep learning. Research Technology Management, 61(6), pp. 22-27.

LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning. Nature, Volume 521, pp. 436-444.

Lee, S.-W.et al., 2021. Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review. Journal of Network and Computer Applications, Volume 187, p. 103111.

Li, A. C., Pinto, L. & Abbeel, P., 2020. Generalized Hindsight for reinforcement learning. Vancouver, Canada, arXiv, pp. 11-14.

Liao, Z., Zhang, P. & Chen, M., 2022. ML4ML: Automated invariance testing for machine learning models. Newark, CA, IEEE, pp. 34-41.

Li, D. et al., 2021. Battery fault diagnosis for electric vehicles based on voltage abnormality by combining the Long Short-Term Memory neural network and the Equivalent Circuit Model. 36(2), pp. 1303-1325.

Li, G. et al., 2020. Machine learning enabled high-throughput screening of hydrocarbon molecules for the design of next generation fuels. Fuel, Volume 265, pp. 1-7.

Li, J. et al., 2021. Machine learning aided bio-oil production with high energy recovery ad low nitrogen content from hydrothermal liquefaction of biomass with experiment verification. Chemical Engineering Journal, Dec, Volume 425, pp. 1-12.

Li, M. & Wang, J., 2019. An emperical comparison of multiple linear regression and artificial neural network for concrete dam deformation modelling. Mathematical Problems in Engineering, pp. 1-13.

Mamun, O., 2019. Bayesian model selection: As a feature reduction technique. [Online].

Martinez, D. et al., 2020. Rotorcraft virtual sensors via deep regression. Journal of Parallel Distributed Computing, Volume 135, pp. 114-126.

MathWorks, 2022. Deep Deterministic Policy Gradient (DDPG) Agents. [Online] Available at: https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html

McCaffrey, J., 2012. Test run - Evolutionary Optimization Algorithms. MSDN Magazine, 27(6).

McCandless, Jr., W., Dettwiller, I. & George, G., 2021. Intelligent black box verification, validation, and accredidation for rotorcraft performance modeling. Journal of the American Helicopter Society, Jul, 66(3), pp. 1-14.

Mihalic, F., Truntic, M. & Hren, A., 2022. Hardware-in-the-loop simulations: a historical overview of engineering challenges. Electronics, 8 Aug, 11(15), p. 2462.

Mnih, V. et al., 2016. Asynchronous Methods for Deep Reinforcement Learning. New York, NY, arXiv, pp. 1-19.

Mohebali, B., Tahmassebi, A., Meyer-Baese, A. & Gandomi, A. H., 2020. Chapter 14: Probabilistic neural networks: a brief overview of theory, implementation, and application. In: Handbook of Probabilistic Models. s.l.:Science Direct, pp. 347-367.

Mowbray, F. et al., 2020. Predicting hospital admission for older emergency department patients: Insights from machine learning. International Journal of Medical Informatics, Volume 140, pp. 1-8.

Muggleton, S., 1995. Inverse entailment and Progol. s.l., NGCO, pp. 245-286.

Nam, J. et al., 2021. Development and verification of a deep learning algorithm to evealuate small-bowel preparation quality. Diagnostics, Jun, Volume 11, pp. 1-10.

Pham, B. T., et al., 2020. Flood risk assessment using hybrid artificial intelligence models integrated with multi-criteria decision analysis in Quang Nam Province, Vietnam. Journal of Hydrology, Nov, Volume 592, pp. 1-15.

Raman, R., Gupta, N. & Jeppu, Y., 2021. Framework for formal verification of machine learning based complex system-of-system. s.l., s.n., pp. 310-326.

Razavi, S. et al., 2021. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. Environmental Modelling and Software, Mar, Volume 137, pp. 1-22.

Ren, S., He, K., Girshick, R. & Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. China, Microsoft.

Ribeiro, M. T., Wu, T., Guestrin, C. & Singh, S., 2020. Beyond Accuracy: Behavioral testing of NLP models with CheckList. ACL, Association for Computational Linguistics, pp. 4902-4912.

Rowland, M. et al., 2018. An analysis of categorical distributional reinforcement learning. s.l., PMLR, pp. 29-37.

Saikia, P., Baruah, R. D., Singh, S. K. & Chaudhuri, P. K., 2020. Artificial neural networks in the domain of reservoir characterization: A review from shallow to deep models. Computers and Geosciences, Volume 135, pp. 1-13.

Salciccioli, J., Crutain, Y., Komorowski, M. & Marshall, D., 2016. Sensitivity Analysis and Model Validation. In: Secondary Analysis of Electronic Health Records. Cambridge(MA): Springer, Cham, pp. 263-271.

Sammut, C. & Webb, G., 2011. Decision stump. Encyclopedia of Machine Learning.

Schulman, J. et al., 2017. Proximal Policy Optimization Algorithms. Research Gate.

Schulte, R. V., Prinsen, E. C., Hermens, H. J. & Buurke, J. H., 2021. Genetic algorithm for feature selection in lower limb pattern recognition. Frontiers in Robotics and AI, 25 Oct.Volume 8.

Selig, J., 2022. What Is Machine Learning? A Definition.. [Online]
Available at: https://www.expert.ai/blog/machine-learning-definition/

Shani, L., Efroni, Y. & Mannor, S., 2020. Adaptive Trust Region Policy Optimization: Global convergence and faster rates for regularized MDPs. s.l., AAAI Organization, pp. 5668-5675.

Shen, G., 2020. Image understanding via learning weakly-supervised cross-modal semantic translation. Journal of Visual Communication and Image Representation, Volume 71, pp. 1-7.

Shukla, A., Tiwari, R. & Kala, R., 2010. Modular Neural Networks. Towards Hybrid and Adaptive Computing, pp. 307-335.

Sihag, P., Karimi, S. M. & Angelaki, A., 2019. Random forest, M5P and regression analysis to estimate the field unsaturated hydraulic conductivity. Applied Water Science, 04 Jul.9(129).

Sim, H., Oh, J.-C. & Lee, H.-O., 2010. Multiple Reduced Hypercube(MRH): A new interconnection network reducing both diameter and edge of hypercube. International Journal of Grid and Distributed Computing, pp. 1-12.

Simplilearn, 2023. Intro to Deep Belief Network (DBN) in deep learning. [Online]
Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-belief-network#:~:text=Deep%20Belief%20Networks%20(DBN)%20is,a%20multi%2Dlayer%20belief%20network.

Sinaga, K. P. & Yang, M.-S., 2020. Unsupervised K-Means Clustering algorithm. IEEE Access, Volume 8, pp. 80716-80727.

SkyQuest Technology, 2022. Machine Learning Market to hit sales of $164.05 billion | Machine Learning has Become Essential for Decision Making and Data Pre-processing, USA: GlobeNewswire, Inc..

Sogeti, 2023. State transition testing. [Online]
Available at: https://www.tmap.net/wiki/state-transition-testing#:~:text=State%20Transition%20testing%20is%20a,valid%20and%20invalid%20state%20transitions.

Song, Z. & Xia, Z., 2022. Carbon emission reduction of tunnel construction machinery system based on self-organizing map-Global particle swarm optimization with multiple weight varying models. IEEE Access, Volume 10, pp. 50195-50217.

Sung, J. M. et al., 2019. Development and verification of prediction models for preventing cardiovascular diseases. Public Library of Science (PLOS) ONE, 14(9), pp. 1-12.

Sutton, C. & McCallum, A., 2012. An introduction to Conditional Random Fields for relational learning. Foundations and Trends in Machine Learning, 4(4), pp. 267-373.

Touzani, S., Granderson, J. & Fernandes, S., 2018. Gradient boosting machine for modeling the energy consumption of commercial buildings. Energy and Buildings, Volume 158, pp. 1533-1543.

Tufekci, P., 2014. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, Sept, Volume 60, pp. 126-140.

Umanoide, 2021. Unsplash. [Online]
Available at: https://unsplash.com/photos/tHS9j3HWT1s

Vatanpour, S. & Hrudey, S., 2015. Can public health risk assessment using risk matrices be misleading?. International Journal of Environmental Research and Public Health, 12(8), pp. 9575-9588.

Vergara, J. R. & Estevez, P. A., 2013. A review of feature selection methods based on mutual information. Neural Computing and Applications, Volume 24, pp. 175-186.

Viceconti, M. et al., 2021. In silico trials: Verification, validation, and uncertainty quantification of predictive models used in the regulatory evaluation of biomedical products. Methods, Jan, Volume 185, pp. 120-127.

Wang, J., Lu, S., Wang, S.-H. & Zhang, Y.-D., 2022. A review on extreme learning machine. Multimedia Tools and Applications, pp. 41611-41660.

Wei, H., Jai, H., Li, Y. & Xu, Y., 2020. Verify and measure the quality of rule based machine learning. Knowledge-Based Systems, Volume 205, pp. 1-14.

Williams, R. J., 1992. Simple statistical gradient-following algorithms for Connectionist Reinforcement Learning. Machine Learning, 8(3-4), pp. 229-256.

Wuest, T., Weimer, D., Irgens, C. & Thoben, K.-D., 2016. Machine learning in manufacturing: advantages, challenges, and applications. Production & Manufacturing Research: An Open Access Journal, 4(1), pp. 23-45.

Xie, L. et al., 2018. Improved spatial pyramid matching for scene recognition. Pattern Recognition, Volume 82, pp. 118-129.

Xie, X. et al., 2011. Testing and validating machine learning classifiers by metamorphic testing. Journal of Systems and Software, Apr, 84(4), pp. 544-558.

Yen, H. P. H. et al., 2021. Locally weighted learning based hybrid intelligence models for groundwater potential mapping and modeling: A case study at Gia Lai province, Vietnam. Geoscience Frontiers, 12(5).

Yu, H., Chen, G. & Gu, H., 2020. A machine learning methodology for multivariate pore-pressure prediction. Computers and Geosciences, Volume 143, pp. 1-18.

Zanin, M., 2021. Simplifying functional network representation and interpretation through causality clustering. Scientific Reports, p. 15378.

Zhao, Z. et al., 2013. Ship classification with high resolution TerraSAR-X imagery based on analytic hierarchy process. International Journal of Antennas and Propagation, pp. 1-13.

APPENDICES

Appendix A: Reviewed Algorithms

The Single Layer Perceptron (SLP) was the first supervised neural network (NN) algorithm (Li, et al., 2020). Supervised NNs train by exposure to a set of data that consists of examples that contain desired outputs from given inputs. The SLP is known for having simple application with understandable results. Since SLP only has one layer of perceptron networks, it only works for linear functions. The authors find SLP to have high accuracy at 99.99% for screening hydrocarbon molecules in fuel development. In the study, SLP identifies 28 molecules, all of which are new hydrocarbon structures.

Multilayer Perceptron (MLP) is a supervised ML algorithm that has at least one extra hidden layer of perceptron network allowing for learning of non-linear functions. Two articles apply MLP algorithms against other algorithms to test for accuracy. The first article team compares MLP with three decision tree induction algorithms: Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting Machine (GBM) (Yu, et al., 2020). The goal is to find the best predictor for multivariate pore-pressure. The second article team compares MLP against a classical regression model to detect anomalies in substation networks (Kreimel, et al., 2020). In the first article, the team builds all ML models with optimized hyperparameters and uses the mean cross-validation score under

variations of different hyperparameters to determine accuracy level. The team finds MLP to be approximately 72% accurate, SVM 79%, GBM 84%, and RF the highest accuracy at 87%. Therefore, RF achieves better performance regarding pore-pressure prediction accuracy. From the second article, the team compares the MLP model against the formal classical regression model, finding the MLP model to have an accuracy of 91% versus the accuracy of the regression model at 99%. It is unusual to have a regression model perform better than a ML model. There are a couple of issues with the study that could cause error in the ML model predictions. One issue is the use of a small sample size, and the other issue is the author indicated data extraction problems. Overall, MLP shows to be an acceptable predictor for multi-layer perceptron networks with a large sample size and appropriate training.

Another article uses a hybrid of the supervised Genetic algorithm (GA) combined with the NN algorithm Radial Basis Function (RBF) in conjunction with a Reinforced Learning Tree (RLT) algorithm (Gajewski & Valis, 2020). The study aims to extend engine operation time by postponing oil changes in systems subject to discontinuous work. The hybrid model has high accuracy on the technical system conditions and reliability assessment. In this article, the model fit over operating time determines the accuracy. A second article also uses many different combinations (hybrids) of algorithms including the RBF (Saikia, et al., 2020). The article team compares performance and characteristics of each algorithm to characterize reservoirs. The tested algorithms are Feedforward NN (FFNN), Ensemble Model (EM), Functional Network (FN), Modular NN (MNN), Radial Basis Function (RBF) NN, Probabilistic NN (PNN), and GA. **Error! Reference source n ot found.**Table 8 below gives a comparison between the characteristics of these different

models. As shown in the table, the performance of the algorithms at model selection depends on the data type.

Table 8. Article Algorithm Characteristics.

Algorithm	Description
FFNN	simplest NN; use for complex functions
EM	best model selection from multiple separate NN models; performance depends on the decision-making criteria
FN	provides better generalization capability in reduced time; activation function learns during the training and no need of weight initialization; learned function is interpretable; more like problem-driven model than data-driven model; changing a part of the network requires to change the whole mathematical equations for training
MNN	better than basic NN model in learning capability, computational time, and accuracy; easier to handle; not automatic in terms of selection of the number of networks and architecture
RBF	generalizes well; avoids overfitting; complex to use
PNN	works well even for complex functions; training is easy and fast; comparatively insensitive to outliers; inefficient in terms of memory

Table 8 Continued.	
GA	combines the strength of both NN and GA; helps in optimizing the parameters of ANN to obtain better model prediction; high computational cost

Semi-supervised learning algorithms combine a small amount of labeled data with a large amount of unlabeled data during training while unsupervised learning algorithms do not provide any labeled data examples during training of the model. Unsupervised learning algorithms must discover patterns in the training data and independently determine the best choice for the output. Two reviewed articles use or discuss the supervised Backpropagation (BP) algorithm in their pursuit for prediction accuracy. The first article discusses the benefits of using BP and other DL algorithms for prediction (LeCun, et al., 2015). The article emphasizes the advantages that DL algorithms have for nonlinear model prediction accuracy. This article did not perform any experimental tests. The second article team uses the BP algorithm and compares it to Multiple Linear Regression (MLR), a Step-wise Regression (SR), and a supervised Extreme Learning Machine (ELM) (Li & Wang, 2019). The article team wants to determine which of these four algorithms would perform best at modelling dam deformation. The team finds BP and ELM are more suitable for reproducing nonlinear effects and complex interactions between dam input variables and responses.

Several articles discuss the unsupervised Convolutional NN (CNN) algorithm. CNN is a DL algorithm that uses BP to utilize multiple layers of simple nonlinear processing modules for feature extraction and transformation. CNN is unsupervised since

there is no direction given to the passing of information from one module to the adjoining module. The first article discusses CNN as a very powerful prediction DL algorithm that produces a successful generalized model (LeCun, 2018). Here the author is referring to a successful generalized model as one that is flexible and allows for a nonlinear error term. The author states, “The best aspect of training a deep convolutional net is that you don’t have to specify what motifs need to be detected at each layer: the learning procedure adjusts the coefficients in all the layers so that the system automatically learns to detect the right features or combinations of features.” Image detection, voice recognition, and AI transportation vehicles are examples of systems that apply CNN algorithms. This article only discusses ML processes and does not discuss any experiments. This author, LeCun, is also the first listed author for the second reviewed article that discusses CNN algorithms (LeCun, et al., 2015). In this article; LeCun, Bengio, and Hinton reiterate the importance of BP to inform the multiple layers for unsupervised learning to occur. The authors state, “As it turns out, multilayer architectures can be trained by simple stochastic gradient descent. If the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure.” In other words, the CNN algorithm success is dependent on the BP procedure. A third article uses the CNN algorithm, in various forms, to compare to weakly-supervised cross-modal semantic translation in an image understanding model (Shen, 2020). The author defines ‘Semantic Gap’ as an image limitation that occurs because of the distortion in a low-level image. While comparing two images for visual response, the author uses a weakly-supervised cross-modal semantic translation to find that the quality of the two images does not match. This weakly-supervised model is a designed manifold embedding algorithm that

automatically translates image-level text, semantic labels, into several pixel-level image regions. Image classification and retrieval improves with this method. In experiments, the team compares their model to other popular image models such as: Spatial Pyramid Matching-based (SPM) Traditional Image Kernel, Multiresolution Histogram (MRH), Fixed-length Walk Kernel (FLWK), and Super Vector (SV)-based Image Encoding. The team also compares their method with CNN as well as other CNN architecture such as the Region CNN (R-CNN), faster R-CNN, and Multi-column CNN (M-CNN). Experiments show that the author's weakly-supervised model achieves competitive performance. The CNN models have very similar accuracy to the weakly-supervised model, within .01 difference.

Another one of the most popular DL network architectures is the Recurrent Neural Network (RNN). CNN is appropriate for processing data that have a fixed-size grid-like structure; however, the data may have more of a sequential varying-length structure. RNN is an adequate algorithm for this type of data because it has the capacity to remember relevant information from earlier in the sequence. One reviewed journal article uses RNN algorithms to predict Cardiovascular Diseases (CVD) (Sung, et al., 2019). The authors compare RNN to a Cox Regression model. The study team shows that RNN significantly improves the predictive power of CVD.

Many scientists combine algorithms with methods in field to develop successful models. The Long Short-Term Memory (LSTM) algorithm combined with the Equivalent Circuit Model (ECM) is one method that an article team uses to diagnose battery faults for electric vehicles (Li, et al., 2021). LSTM is a DL model with a type of RNN architecture. The team combines RNN with the ECM and produces accurate fault diagnosis. A second

article uses a GA with DL Evolutionary Optimization (DLEO) to develop a robust and accurate model that supports migrating from fixed-based to condition-based maintenance for Rotocraft virtual sensor components (Martinez, et al., 2020). The GA DLEO algorithm trains and models in parallel while diverging NN architectures. This adds three hidden layers and drops out rates strengthening the model. The method is ideal for 'Big Data.' The authors find GA DLEO excellent at exposing tendencies and patterns and ultimately discovering the best generalized model.

Rule-Based machine learning (RBML) algorithms use logic to handle inconsistencies in the data. One article explains how to improve RBML with the "Logic-based Automatic Reasoning Method" (Wei, et al., 2020). The team discusses the theory and approach to verify and increase the consistency degree of the RBML model. Showing a step-by-step approach, the team transforms the model into a conjunctive normal form and improves the model through a logic-based automated reasoning method.

Three different reviewed articles use the Support Vector Machine (SVM) algorithm. This synthesis reviewed two of the articles in earlier paragraphs looking at other models for comparison. A third article uses SVM to predict hospital admission for older emergency room patients (Mowbray, et al., 2020). The research team compares SVM to Linear Regression (LR), Classification and Regression Tree (CART), RF, and Gradient Boosting Tree (GBT). The Area Under the Receiver Operation Characteristic Curve (AUC) estimates performance accuracies. The team reports accuracy, sensitivity, and specificity of each model. In all three categories, the SVM performs adequate; however, RF outperforms all other compared models in each category.

Gradient Boosting Tree (GBT) or Gradient Boosting Machine (GBM) refers to a powerful ML algorithm that uses regression tree methods. A reviewed article employs GBM to model the energy consumption of commercial buildings (Touzani, et al., 2018). GBM involves the partitioning of the input parameter space into distinct and non-overlapping regions following a set of if-then rules. The splitting rules identify regions that have the most homogeneous response to the predictor. A simple model, such as a constant, fits within each region. The team learns that GBM improves the r-squared prediction accuracy and the Root Mean Square Error (RMSE) in more than 80% of the cases.

One article uses a Bayesian Network (BN) algorithm and compares it to a decision tree induction algorithm, RF (Bhattacharya & Mishra, 2018). This study applies BN and RF algorithms to model sedimentary facies and fractures to make predictions and causal insights. The study team finds RF prediction to be better than BN prediction by 7.03%. RF also has a slightly lower variance and bias. However, the team finds BN to be better than RF at revealing complex relations and inferring the causal relationships derived through data driven modeling.

Another article team uses Super Learning (SL) to improve accuracy and reduce variance when predicting behavior classification from accelerometry (Ladds, et al., 2017). SL combines base learners, also known as weak learners, in an optimal manner to achieve overall improved accuracy. A base learner is a ML algorithm that combines with other algorithms to form an optimal prediction model. This study applies a selection of weighted base learners: RF, GBM, and Logistic Regression (LR). A cross-validation method determines the optimal SL feature combination. The SL algorithm produces a model that has more accuracy and less variance than the base learners.

From the synthesis of the algorithms, the most recent articles tend to prefer the use of inductive learning and NN models with DL tendencies. Some of the ML algorithms that develop these models are: RF, BP, CNN, RNN, and SL, as well as hybrid algorithms. The inductive learning algorithms use decision tree induction or rule induction; whereas, the DL NN algorithms are supervised, unsupervised, or reinforced. DL models are complex; however, with the multiple layers they are very quick and accurate at prediction.

Appendix B: Detailed ML V and V Process

Step 1: Design and test the system's adherence to the requirements through black-box experimentation.

- Refer to requirements document contained in MBSE file
    - Trace all tested requirements to system component target variables (verification)
    - Consider target variables and types (i.e., binary, continuous numeric, categorical)
    - Consider input variables and types (if available)
- List the objectives of your test
- Identify the scope of your test
- Define a system failure
    - Perform a risk assessment
- Acquire data for testing
    - Simulation data from:
        - Bootstrapping (Table 3, Column 1)
    - Validation data (not used in training system model)
        - A random "Hold Out" sample
    - Combinatorial Interaction Testing (CIT) (Table 3, Column 1)
        - See article, "Systematic Training and Testing for Machine Learning Using Combinatorial Interactive Testing" (Cody, et al., 2022)

    - Equivalence Class Partitioning (ECP) Method to improve coverage with less samples (Table 3, Column 6)
    - Orthogonal Arrays to improve coverage with less samples (Table 3, Column 1)
- Verify sample size coverage is adequate with stakeholders
    - Greater than or equal to 80% coverage is usually preferred
    - In a random sample (Hold Out), coverage is unknown
        - Use large sample of over 300 samples (Kelleher, et al., 2020)
        - If possible, determine risk of missing a system failure, type II error (accepting a bad system)
- Identify the testing environment
    - Lab, field, virtual, live-simulation, etc.
- Randomize trials for testing
    - Indicate if “Split-plot” or “No Randomization” is possible
    - Indicate if test is an observational study with no randomization
- List constraints and assumptions for the test
- Set up test cases
    - Use all applicable test case design techniques (Table 3, Column 6)
        - Equivalence Class Partitioning (ECP)
        - Boundary Value Analysis (BVA)
        - Decision Table Testing
        - State Transition Testing

    - Describe test events and trials
    - Give initial conditions
    - Give test inputs and outputs
- Trace the requirements to the test cases
    - Confirm that every requirement is satisfied by a test case
- Set up data collection method
    - Spreadsheet
    - Automatic
    - Database
    - Other tools
- Perform all test cases
    - Run the test data inputs through the ML system to acquire outputs
- Record results for black-box testing
    - Give date and time of testing
    - List any failures
    - Rate severity of failures
    - Trace failures to their requirement
    - Note input values during failures
    - List failure circumstances and environment
    - Note possible failure causes
    - Determine system accuracy
    - Make conclusions

- If the internal structure of the system (i.e., features, levels, algorithm, code) is available for testing and there were failures in Step 1 and the system accuracy is considered defective, then proceed to Step 2, else advance to Step 3

Step 2: Use white-box methods to determine possible cause and effect of Step 1 failures and to enhance the system performance through examining and improving the internal structure.

- Obtain pertinent information from the developer
    - What are the features and levels? How were they ranked? How were they validated?
    - What target variables were used? What are the measures of the target variables?
    - How was the data obtained? Is the data clean? How was the data sampled? How was the data partitioned? Was k-fold cross validation used?
    - What algorithms were used to train the model? What software tool was used? Was a hyperparameter used?
    - How was the model validated?
- Obtain a clean datafile from the developer
    - Load data into software to create a model (i.e., R or Python)
    - Partition data
        - For large datasets, separate data into a training and validation dataset (recommended 70%:30%)

        - For small datasets, under 300 instances, use bootstrapping method (Table 3, Column 1)
    - Summarize the datasets
        - Dataset sizes
        - Feature levels, feature data types, and distributions
        - Target types and distributions
        - Look at a few rows of the data, if possible
        - Perform a statistical summary of features and target(s) (i.e., means, minimums, maximums, quartiles, outliers)
        - Run a correlation analysis to determine correlation between features
        - Remove one of the correlated features or use principal component analysis (PCA) (Table 3, Column 5)
- Analyze and Rank Features
    - Use PCA or another method (from Table 3, Column 3) to rank and choose the best features for prediction
- Choose at least 3 algorithms (Table 3, Column 2) to train the model for comparison (the most common algorithms are in bold print)
    - Choose one algorithm to be the same algorithm used by the developer
    - Use a mixture of algorithms that are of different ML Types (Table 3)

    - Ensure that an algorithm is chosen in accordance with the target variable data type matched to the ML Type listed in Table 3 (i.e., support vector machine (SVM) can be used for either categorical or numerical target variables but is best suited for categorical, random forest (RF) can be used for either categorical or numerical target variables, simple linear regression (SLR) is used for numerical data)
    - Set up k-fold cross validation (recommend $k \geq 5$)
- If hyperparameters are employed, optimize the hyperparameters
    - Use a method (from Table 3, column 4) (i.e., mean cross validation score (MCVS), gradient descent (GD), or grid search (GS))
- Run algorithms and create models
    - Choose a metric for model comparison (i.e., root mean square error (RMSE))
        - The metric for model comparison depends on the data type
- Choose the best model according to the comparison metric and at least one other method (from Table 3, Column 5) (i.e., r-squared (R2) or area under the curve (AUC))
- Improve model
    - Use directional testing (Table 3, Column 5) with or without features
    - Use a method (from Table 3, Column 3) to determine if any features should be removed

- Test performance of new model (system)
    - Perform Step 1 on the new model
- Compare to black-box testing results of the new model versus the developer's model results from Step 1
    - Do the same failures occur?
    - Perform minimum functionality testing (Table 3, Column 5) to locate cause and effect of failures for both models

Step 3: Perform black-box system validation through operational testing (Table 3, Column 7).

- Note any failures

Step 4: Report findings to developer and other stakeholders.

- If the original developer's model is proven to be the best model according to the comparison metric chosen above, submit Step 1 and Step 3 results. Note in the report that the model is superior against other models of given algorithm types
- If the new model is proven to be the best model according to the comparison metric, submit Step 1, Step 2, and Step 3 results

Appendix C: Case Study R Code

This section shows the Case Study R Code, with its associated output, used to perform the V&V of the developed ML Framework. The code displays the comments in red, the code in blue, and the output in black. The R code and output is as follows:

```
> # Read in the datafile and name columns.
> library(caret)
> original_data <- read.csv("C:\\Documents\\Dissertation\\case_study_data.csv, header=TRUE)
> data <- original_data
> colnames(data) <- c("Temp","Vacuum","Pressure","Rel_Humidity","Energy_Output")
> # Create a random list of 80% of data for training and 20% for validation.
> training_data <- createDataPartition(data$Energy_Output, p=.80, list=FALSE)
> ValidData <- data[-training_data, ]
> # Observe data characteristics.
> # Samples.
> dim(data)

[1]  9568   5

> dim(training_data)

[1]  7656   1

> dim(ValidData)

[1]  1912   5

> # Data variable types, example rows, and summary statistics.
> sapply(data, class)

        Temp        Vacuum      Pressure  Rel_Humidity Energy_Output
   "numeric"     "numeric"     "numeric"     "numeric"     "numeric"

> head(data)
```

```
   Temp Vacuum Pressure Rel_Humidity Energy_Output
1 14.96  41.76  1024.07        73.17        463.26
2 25.18  62.96  1020.04        59.08        444.37
3  5.11  39.40  1012.16        92.14        488.56
4 20.86  57.32  1010.24        76.64        446.48
5 10.82  37.50  1009.23        96.62        473.90
6 26.27  59.44  1012.23        58.77        443.67
```

```
> summary(data)
```

```
     Temp           Vacuum         Pressure       Rel_Humidity    Energy_Output
Min.   : 1.81   Min.   :25.36   Min.   : 992.9   Min.   : 25.56   Min.   :420.3
1st Qu.:13.51   1st Qu.:41.74   1st Qu.:1009.1   1st Qu.: 63.33   1st Qu.:439.8
Median :20.34   Median :52.08   Median :1012.9   Median : 74.97   Median :451.6
Mean   :19.65   Mean   :54.31   Mean   :1013.3   Mean   : 73.31   Mean   :454.4
3rd Qu.:25.72   3rd Qu.:66.54   3rd Qu.:1017.3   3rd Qu.: 84.83   3rd Qu.:468.4
Max.   :37.11   Max.   :81.56   Max.   :1033.3   Max.   :100.16   Max.   :495.8
```

```
> # Look at data distributions of feature and target variables with boxplots.
> x <- data[ , 1:5]
> par(mfrow=c(1,5))
> for (i in 1:5) {boxplot(x[ , i], main=names(data)[i])}
```

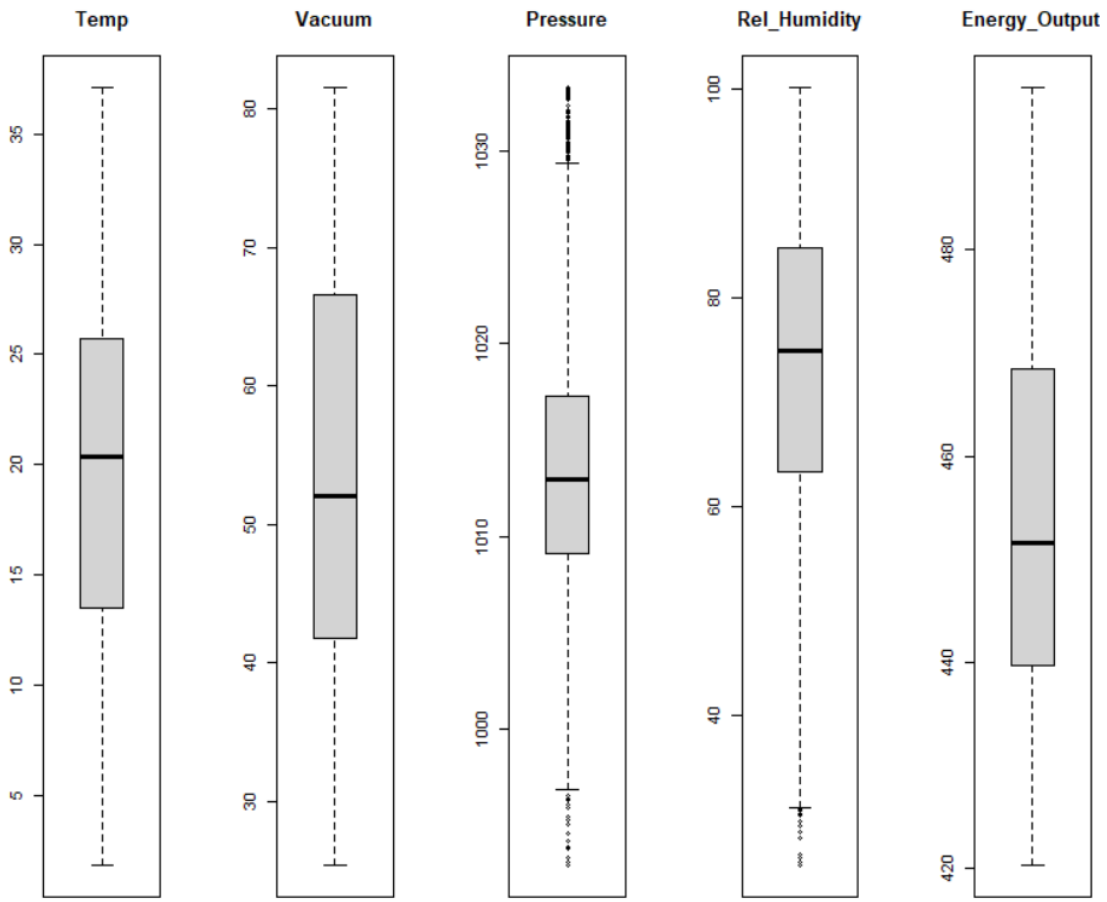

Figure 19. Case Study Boxplots.

```
> # Train system model using 10-fold cross validation.
> # The SVM algorithm was arbitrarily chosen to train the model.
> control <- trainControl(method="cv", number=10)
> metric <- "RMSE"
> set.seed(13)
> fit.svm <- train(Energy_Output~., data=data, method="svmRadial",
+ metric=metric, trControl=control)
> Print(fit.svm)
```

```
Support Vector Machines with Radial Basis Function Kernel

7656 samples
   4 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 6890, 6891, 6892, 6889, 6892, 6891, ...
Resampling results across tuning parameters:

  C     RMSE      Rsquared   MAE
  0.25  4.059537  0.9432381  3.047396
  0.50  4.025685  0.9442409  3.010346
  1.00  4.006370  0.9448187  2.985773

Tuning parameter 'sigma' was held constant at a value of 0.3689378
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.3689378 and C = 1.
```

```
> # Make predictions with the SVM model using the Validation Data.
> predictions <- predict(fit.svm, ValidData)
> # Combine the original data with the predicted values into a new file called Final_Data1.
> # Call the original target values 'Actual' and the predicted values "Predicted".
> # Make the file into a dataframe called Final_Data2.
> Final_Data1 <- cbind(Actual=ValidData$Energy_Output, Predicted=predictions)
> Final_Data2 <- as.data.frame(Final_Data1)
> # Look at a few rows of the new file.
> head(Final_Data2)

  Actual Predicted
1 444.37  444.0758
2 467.35  464.1397
3 453.02  457.1349
4 433.99  435.1278
5 459.85  462.2701
6 437.89  437.7351
```

```
> # Calculate the error and add it as a column into a new file called Final_Data3.
> error <- Final_Data2$Actual – Final_Data2$Predicted)
> Final_Data3 <- cbind(Final_Data2, error)
> head(Final_Data3)

  Actual Predicted      error
1 444.37  444.0758  0.2941989
2 467.35  464.1397  3.2102868
3 453.02  457.1349 -4.1149202
4 433.99  435.1278 -1.1378205
5 459.85  462.2701 -2.4201187
6 437.89  437.7351  0.1549284
```

```
> # Calculate the "RMSE" for model comparison later.
> RMSE1 <- sqrt(mean(Final_Data3$error^2))
> head(RMSE1)

[1] 4.028321

> # Summary of results and model.
> summary(Final_Data3)

    Actual        Predicted         error
Min.   :425.1   Min.   :428.8   Min.   :-26.70844
1st Qu.:439.8   1st Qu.:439.9   1st Qu.: -2.75067
Median :451.6   Median :451.7   Median : -0.09926
Mean   :454.2   Mean   :454.5   Mean   : -0.30812
3rd Qu.:468.4   3rd Qu.:468.6   3rd Qu.:  2.12278
Max.   :495.4   Max.   :495.2   Max.   : 15.81830

> model_summary <- lm(Actual~Predicted, data=Final_data3)
> summary(model_summary)

Call:
lm(formula = Actual ~ Predicted, data = Final_Data3)

Residuals:
     Min       1Q   Median       3Q      Max
-26.3926  -2.4634   0.2419   2.4323  16.3156

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.865550   2.497877   1.548    0.122
Predicted   0.990818   0.005492 180.422   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.016 on 1910 degrees of freedom
Multiple R-squared:  0.9446,    Adjusted R-squared:  0.9445
F-statistic: 3.255e+04 on 1 and 1910 DF,  p-value: < 2.2e-16

> # Plot predicted values versus actual values.
> plot(x=Final_Data3$Predicted, y=Final_Data3$Actual,
+ xlab='Predicted Values', ylab='Actual Values',
+main='Predicted vs. Actual Values')
> abline(a=0, b=1)
```

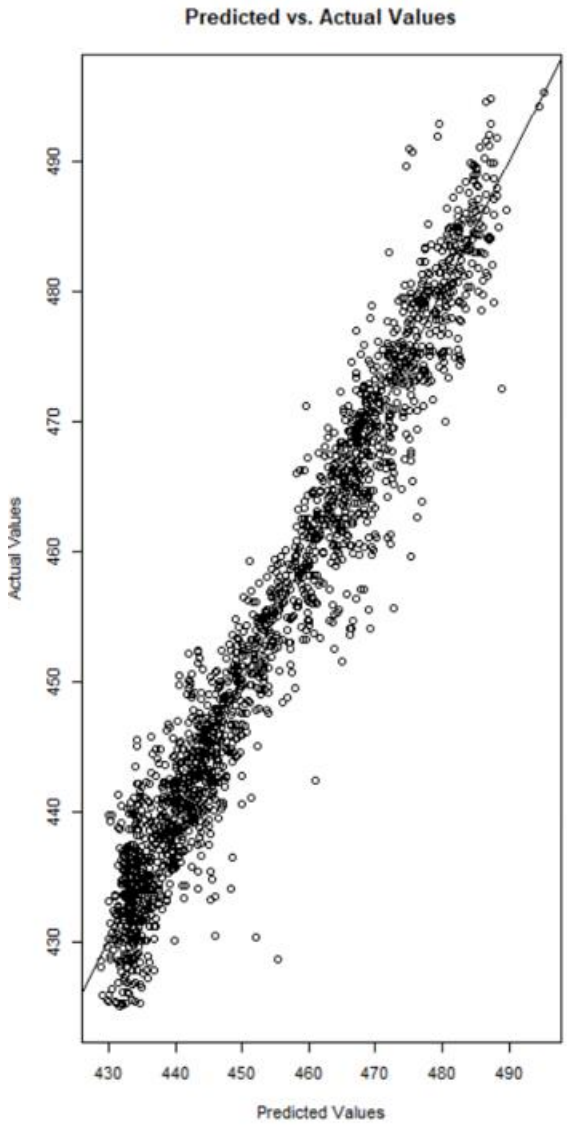

Figure 20. Predicted versus Actual Case Study Values.

```
> # Boxplot of error.
> boxplot(Final_Data3$error, main="error")
```

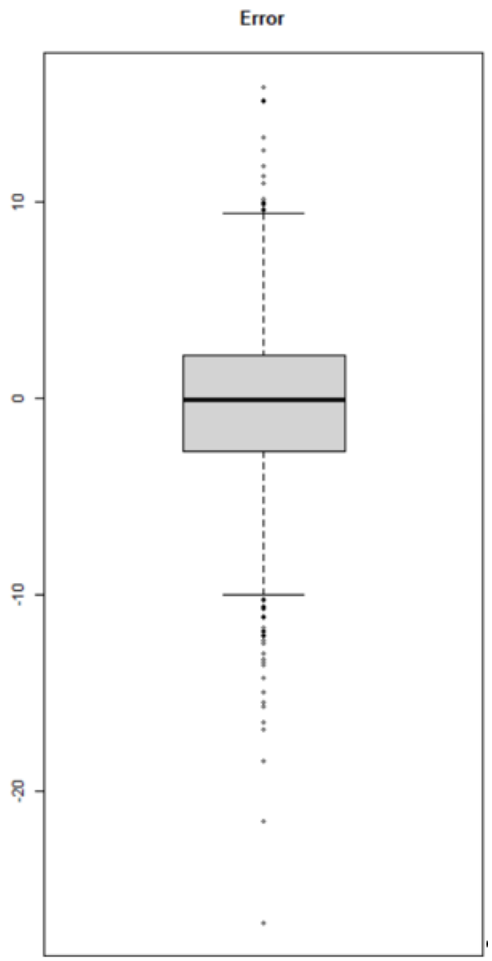

Figure 21. Case Study Error Boxplot.

```
> # 99% confidence interval of error.
> Upper_Limit <- qnorm(0.99, mean=mean(error), sd=sd(error))
> Lower_Limit <- mean(error) – (Upper_Limit – mean(error))
> head(Upper_Limit)

[1] 9.038143

> head(Lower_Limit)

[1] -9.65439

> # Failures are defined as trials where the Predicted value is
> # greater than 11 units from the Actual value.
> # Calculate absolute value of error, the number of errors, and the accuracy of the model.
```

```
> abs_error <- abs(Final_Data3$error)
> Final_Data3a <- (Final_Data3, abs_error)
> length(which(Final_Data3a$abs_error > 11 ))

[1]  30

> # Rename Final_Data3 as Final_Data5. Check correlation test assumptions.
> # Shapiro-Wilk Normality Tests for each feature.
> Final_Data5 <- Final_Data3
> Shapiro.test(Final_Data5$Temp)

        Shapiro-Wilk normality test

data:  Final_Data5$Temp
W = 0.96852, p-value < 2.2e-16

> Shapiro.test(Final_Data5$Vacuum)

        Shapiro-Wilk normality test

data:  Final_Data5$Vacuum
W = 0.90574, p-value < 2.2e-16

> Shapiro.test(Final_Data5$Pressure)

        Shapiro-Wilk normality test

data:  Final_Data5$Pressure
W = 0.99325, p-value = 1.11e-07

> Shapiro.test(Final_Data5$Rel_Humidity)

        Shapiro-Wilk normality test

data:  Final_Data5$Rel_Humidity
W = 0.97879, p-value = 3.084e-16

> # Linearity between each pair of features.
```

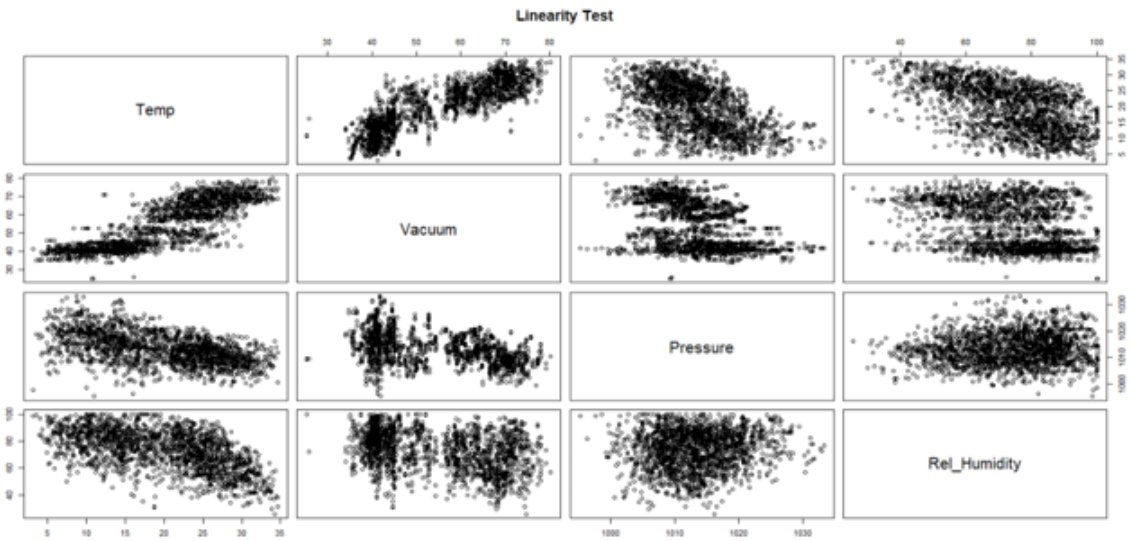

Figure 22. Case Study Linearity Test Results.

```
> # Pearson's Correlation Test between each linear feature pair.
> Cor_Test <- cor.test(Final_Data5$Temp, Final_Data5$Pressure, method="pearson")
> Cor_Test

        Pearson's product-moment correlation

data:  Final_Data5$Temp and Final_Data5$Pressure
t = -26.068, df = 1910, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.5445864 -0.4784219
sample estimates:
       cor
-0.5122638

> Cor_Test <- cor.test(Final_Data5$Temp, Final_Data5$Vacuum, method="pearson")
> Cor_Test

        Pearson's product-moment correlation

data:  Final_Data5$Temp and Final_Data5$Vacuum
t = 70.834, df = 1910, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8382001 0.8629557
sample estimates:
      cor
0.8510501

> Cor_Test <- cor.test(Final_Data5$Vacuum, Final_Data5$Pressure, method="pearson")
```

> Cor_Test

```
        Pearson's product-moment correlation

data:  Final_Data5$Vacuum and Final_Data5$Pressure
t = -20.034, df = 1910, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4530727 -0.3789584
sample estimates:
       cor
-0.4167078
```

> # Kendall's Correlation Test between feature pairs that do not meet the assumptions.
> Cor_Test <- cor.test(Final_Data5$Temp, Final_Data5$Rel_Humidity, method="kendall")
> Cor_Test

```
        Kendall's rank correlation tau

data:  Final_Data5$Temp and Final_Data5$Rel_Humidity
z = -25.463, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
-0.3886903
```

> Cor_Test <- cor.test(Final_Data5$Vacuum, Final_Data5$Rel_Humidity, method="kendall")
> Cor_Test

```
        Kendall's rank correlation tau

data:  Final_Data5$Vacuum and Final_Data5$Rel_Humidity
z = -14.925, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
-0.228021
```

> Cor_Test <- cor.test(Final_Data5$Pressure, Final_Data5$Rel_Humidity, method="kendall")
> Cor_Test

```
        Kendall's rank correlation tau

data:  Final_Data5$Pressure and Final_Data5$Rel_Humidity
z = 4.2286, p-value = 2.351e-05
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
0.06455312
```

```
> # Remove target variable from data to normalize data and plot the correlation matrix.
> drop_target <- subset(ValidData, select = -c(Energy_Output))
> data_normalized <- scale(drop_target)
> corr_matrix <- cor(data_normalized)
> ggcorrplot(corr_matrix)
```

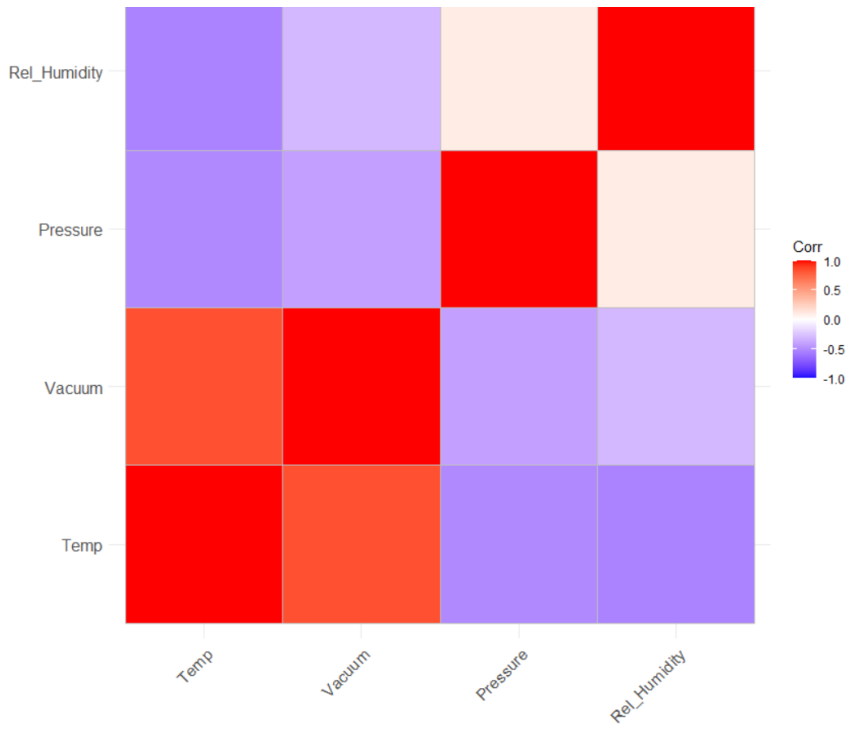

Figure 23. Case Study Correlation Matrix.

```
> # Build MLR and RF models to compare with SVM model.
> control <- trainControl(method="cv", number=10)
> metric <- "RMSE"
> set.seed(13)
> fit.mlr <- lm(Energy_Output~., data=data, metric=metric, trControl=control)
> fit.rf <- randomForest(Energy_Output~., data=data, metric=metric, trControl=control)
> # Make predictions using the MLR and RF models for the validation data.
> # Then calculate error for each model.
> predictions <- predict(fit.mlr, ValidData)
> predict_mlr <- cbind(Actual=ValidData$Energy_Output, Predicted=predictions)
> frame1_mlr <- as.data.frame(predict_mlr)
> error <- frame1_mlr$Actual -frame1_mlr$Predicted
```

```
> frame2_mlr <- cbind(frame1_mlr, error)
> predictions <- predict(fit.rf, ValidData)
> predict_rf <- cbind(Actual=ValidData$Energy_Output, Predicted=predictions)
> frame1_rf <- as.data.frame(predict_rf)
> error <- frame1_rf$Actual -frame1_rf$Predicted
> frame2_rf <- cbind(frame1_rf, error)
> # Compute absolute error, count of errors, and accuracy for MLR and RF.
> abs_error <- abs(frame2_mlr$error)
> frame2a_mlr <- cbind(frame2_mlr, abs_error)
> length(which(frame2a_mlr$abs_error > 11))

[1] 23

> accuracy  = ((dim(frame2a_mlr)-length(which(frame2a_mlr$abs_error >
11)))/dim(frame2a_mlr))*100
> accuracy

[1]   98.79707 -475.00000

> abs_error <- abs(frame2_rf$error)
> frame2a_rf <- cbind(frame2_rf, abs_error)
> length(which(frame2a_rf$abs_error >11))

[1] 8

> accuracy  = ((dim(frame2a_rf)-length(which(frame2a_rf$abs_error > 11)))/dim(frame2a_rf))*100
> accuracy

[1]   99.58159 -100.00000

> # Calculate the RMSE for MLR and RF.
> RMSE_mlr <- sqrt(mean(frame2a_mlr$error^2))
> RMSE_mlr

[1] 4.436288

> RMSE_rf <- sqrt(mean(frame2a_rf$error^2))
> RMSE_rf

[1] 3.161678

> # MLR summary statistics.
> summary(frame2a_mlr)
```

```
    Actual        Predicted         error            abs_error
 Min.   :425.2   Min.   :421.8   Min.   :-23.7167   Min.   : 0.00012
 1st Qu.:439.8   1st Qu.:440.2   1st Qu.: -3.1036   1st Qu.: 1.48483
 Median :451.6   Median :451.9   Median : -0.1052   Median : 3.18655
 Mean   :454.3   Mean   :454.2   Mean   :  0.1253   Mean   : 3.57050
 3rd Qu.:468.4   3rd Qu.:468.7   3rd Qu.:  3.2905   3rd Qu.: 5.06625
 Max.   :494.9   Max.   :488.9   Max.   : 17.5726   Max.   :23.71673
```

```
> model_summary_mlr <- lm(Actual~Predicted, data=frame2a_mlr)
> summary(model_summary_mlr)
```

```
Call:
lm(formula = Actual ~ Predicted, data = frame2a_mlr)

Residuals:
     Min       1Q   Median       3Q      Max 
-23.6565  -3.2147  -0.2617   3.2024  17.2366 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 4.551687   2.796724   1.628    0.104    
Predicted   0.990254   0.006154 160.922   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.434 on 1910 degrees of freedom
Multiple R-squared:  0.9313,	Adjusted R-squared:  0.9313 
F-statistic: 2.59e+04 on 1 and 1910 DF,  p-value: < 2.2e-16
```

```
> # RF summary statistics.
> summary(frame2a_rf)
```

```
    Actual        Predicted         error             abs_error
 Min.   :425.2   Min.   :428.3   Min.   :-26.05430   Min.   : 0.001986
 1st Qu.:439.8   1st Qu.:439.8   1st Qu.: -1.82379   1st Qu.: 0.846041
 Median :451.6   Median :450.8   Median :  0.06213   Median : 1.869967
 Mean   :454.3   Mean   :454.3   Mean   :  0.05187   Mean   : 2.383137
 3rd Qu.:468.4   3rd Qu.:468.1   3rd Qu.:  1.90800   3rd Qu.: 3.403694
 Max.   :494.9   Max.   :490.2   Max.   : 13.68611   Max.   :26.054304
```

```
> model_summary_rf <- lm(Actual~Predicted, data=frame2a_rf)
> summary(model_summary_rf)
```

```
Call:
lm(formula = Actual ~ Predicted, data = frame2a_rf)

Residuals:
     Min       1Q   Median       3Q      Max 
-26.3357  -1.8677   0.0287   1.8363  13.7434 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -4.843395   1.997514  -2.425   0.0154 *  
Predicted    1.010776   0.004394 230.012   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.158 on 1910 degrees of freedom
Multiple R-squared:  0.9652,	Adjusted R-squared:  0.9651 
F-statistic: 5.291e+04 on 1 and 1910 DF,  p-value: < 2.2e-16
```

```
> # Check RF accuracy, failure count, and RMSE without the correlated variable 'Vacuum'.
> drop_vacuum <- subset(data, select = -c(Vacuum))
> fit_reduce_rf <- randomForest(Energy_Output~., data=drop_vacuum,
+ metric=metric, trControl=control)
> reduce_ValidData <- subset(ValidData, select = -c(Vacuum))
> predictions <- predict(fit_reduce_rf, reduce_ValidData)
> predict_reduce_rf <- cbind(Actual=reduce_ValidData$Energy_Output,
+ Predicted=predictions)
> frame_reduce_rf <- as.data.frame(predict_reduce_rf)
> error <- frame_reduce_rf$Actual – frame_reduce_rf$Predicted
> frame_reduce1 <- cbind(frame_reduce_rf, error)
> abs_error <- abs(frame_reduce1$error)
> frame_reduce2 <- cbind(frame_reduce1, abs_error)
> length(which(frame_reduce2$abs_error > 11))
```

```
[1] 20
```

```
> accuracy = ((dim(frame_reduce2)-length(which(frame_reduce2$abs_error >
11)))/dim(frame_reduce2))*100
> accuracy
```

```
[1]    98.95397 -400.00000
```

```
> RMSE_reduce_rf <- sqrt(mean(frame_reduce2$error^2))
> RMSE_reduce_rf
```

```
[1] 4.19913
```

BIOGRAPHICAL SKETCH

Name of Author:
Swala B. Burns

Graduate and Undergraduate Schools Attended:
University of West Florida, Pensacola, Florida
University of South Alabama, Mobile, Alabama

Degrees Awarded:
Bachelor of Science Mathematics, 2001
Master of Science Mathematics, 2004
Doctor of Philosophy Systems Engineering, 2023