SCIENTIFIC TEST & ANALYSIS TECHNIQUES
CENTER OF EXCELLENCE

Continuous Machine Learning for Test and Evaluation

February 2025

Corinne Stafford, Ctr
Joseph Lazarus, Ctr
Brittany Fischer, Ctr

DISTRIBUTION STATEMENT A. Approved for public release; distribution unlimited.
Case number: MSC/PA-2025-0158; 88ABW-2025-0396; CLEARED 27 May 2025

To develop and apply independent, tailored Scientific Test & Analysis Techniques solutions to Test and Evaluation that deliver insight to inform better decisions.

About this Publication:
This work was conducted by the Scientific Test & Analysis Techniques Center of Excellence under contract FA8075-18-D-0002, Task FA8075-21-F-0074.

For more information:
Visit, www.AFIT.edu/STAT
Email, AFIT.ENS.STATCOE@us.af.mil
Call, 937-255-3636 x4736

Technical Reviewer:
Steve Oimoen

Copyright Notice: No Rights Reserved
Scientific Test & Analysis Techniques Center of Excellence
2950 Hobson Way
Wright-Patterson Air Force Base, Ohio

The views expressed are those of the author(s) and do not necessarily reflect the official policy or position of the Department of the Air Force, the Department of Defense, or the U.S. government.

Version: 1, FY25

Abstract

Machine learning (ML) is a powerful tool that can enhance test & evaluation (T&E) by enabling continuous integration of data into predictive models. This paper provides an overview of opportunities for applying ML throughout a capability lifecycle, from mission engineering through sustainment, and details five phase of an ML model lifecycle: plan, data, model, deploy, monitor. Together these five phases guide how a ML model should be initially developed, as well as how the model can be continuously updated to meet T&E needs. A case study is presented that walks through activities during each ML lifecycle phase.

Keywords: machine learning, test and evaluation, developmental test & evaluation as a continuum, modeling and simulation, validation

Table of Contents

Introduction

Machine learning (ML) is a powerful, ever-evolving tool that can learn patterns in data. ML is used across a wide range of applications from generative chatbots to within Department of Defense (DOD) systems themselves. Another area of application for ML is during test and evaluation (T&E), where ML models can be constructed to understand and predict behavior of DOD systems, serving in a modeling and simulation (M&S) role. Currently, ML models are underutilized in T&E primarily due to the additional expertise required for their construction. Even when ML models are constructed, they may not be well-understood by test teams to see continued use throughout a program. ML models learn from data and therefore must be continually updated as new information is gained. In the shift from T&E as a serial set of activities toward developmental test and evaluation as a continuum (dTEaaC), an integrative framework focused on a continuum of activities (Collins & Senechal, 2023), ML models support continuous integration of data into predictive models. Continuous model use and training enables earlier and better-informed decision making, helping to deliver capabilities at the speed of relevance. Additionally, continuous monitoring and retraining enables models to stay relevant into system operations.

While ML resources are vast, there are fewer resources on the use of ML for T&E and M&S. Thus, this paper aims to inform programs on how to effectively use ML for T&E throughout a capability lifecycle. Specifically, this paper focuses on ML models that are used to predict or understand system behavior, not ML-enabled systems.

First, the paper gives background on types of machine learning models that could apply to T&E. Then, a discussion of how ML applies across the capability lifecycle is provided, using specific scenarios to demonstrate various use cases. Next, an overview of the ML lifecycle in T&E is presented, describing the process by which an ML model can be trained and continuously updated to meet decision support needs across the capability lifecycle. The five phases, Plan, Data, Model, Deploy, and Monitor, cover the life of an ML model from its inception through continual updating with new data. Furthermore, a case study is presented where each of these phases is discussed in detail to facilitate more effective use of ML models throughout T&E. A conclusion follows summarizing the paper.

Background

ML is a subset of artificial intelligence (AI) focused on training algorithms to learn patterns and make predictions or decisions based on data. Unlike programs with explicit human-written instructions, ML involves providing data to algorithms and allowing them to discover relationships within the data. This iterative learning process enables the program to improve its performance over time.

There are many different algorithms or methods for machine learning, but they typically fall into one of or a combination of three types:

- **Supervised learning:** uses labeled training data to develop a model based on a “ground truth” reference; examples include regression, decision trees, and gradient-boosting
- **Unsupervised learning**: analyzes data without labels to find patterns or groupings; examples include clustering, anomaly detection, and dimensionality reduction
- **Reinforcement learning**: emulates a trial-and-error approach where an agent makes a series of decisions in a specific environment and is rewarded or punished according to its decisions; examples include autonomous driving, learning-based robots, and natural

language processing

Most often with M&S, the goal is to replicate real-life phenomena; therefore M&S is more likely to leverage supervised learning. Supervised ML models may learn from test data to make predictions about system performance in the future.

Supervised ML models typically serve one of two purposes: numeric prediction or classification. Numeric prediction models output quantitative predictions such as temperature or a flight path. On the other hand, classification models categorize data into discrete classes or labels, such as friendly/hostile aircraft or detect/non-detect. Both types are common in M&S for DOD systems.

The next section discusses how machine learning may be leveraged across the capability lifecycle, as demonstrated with specific scenarios describing applications of ML to T&E.

Machine Learning Across the Capability Lifecycle

Machine learning helps to enable continuous development and T&E of new capabilities. Figure 1 depicts the interplay between test, data, models, and decisions across the capability lifecycle. Models can be used continuously over the lifecycle, from early descriptive models used within model-based systems engineering (MBSE) through predictive subcomponent, component, sub-system, system, and system of systems models. Many models will be built and integrated together into higher level models (e.g. system or system of system) to make operational predictions. Models exist in a feedback loop with data – data helps build, validate, and update models, which can be used to inform test to collect new data. Machine learning models are especially adept at adapting to new data because they are explicitly trained on it. In other words, ML models are dynamic and should be continuously trained and updated over the capability lifecycle. Additionally, ML models can be trained from models themselves. For example, long-runtime physics-based models can be used to train surrogate ML models with much faster execution time, enabling fast prediction at the system or system of systems levels. Both models and data support informed decision making, where the Integrated Decision Support Key (ISDK) provides the traceability between which models and data sources inform which decisions.

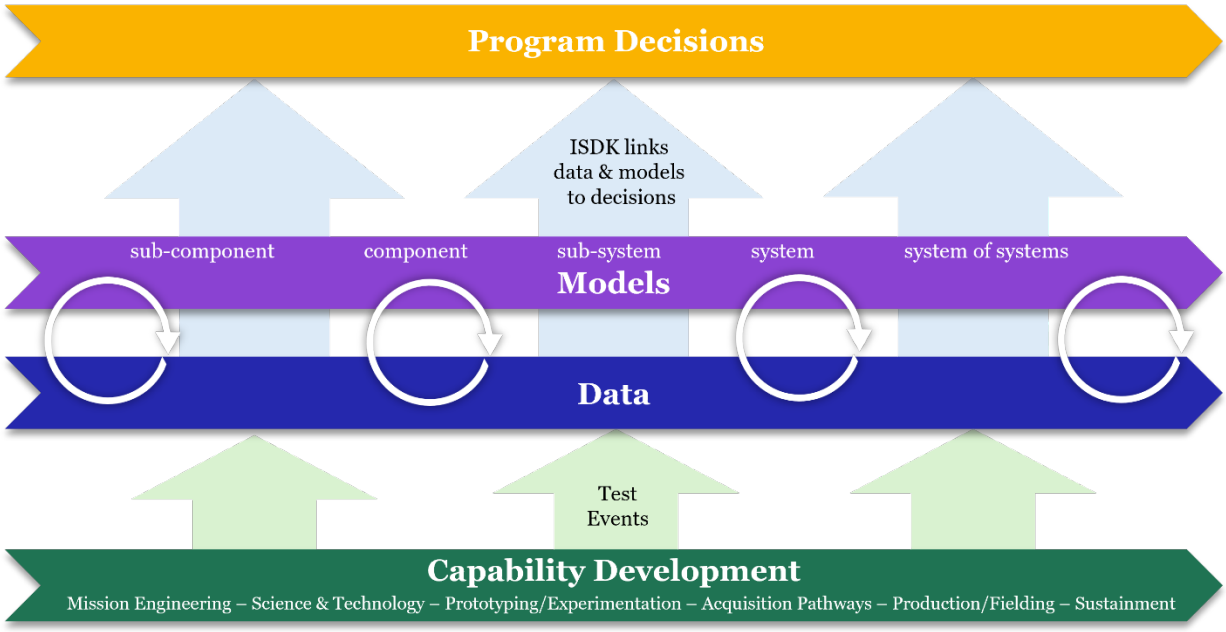

Figure 1
Use of Models and Data to Inform Decisions Across the Capabilty Lifecycle

The scenarios below provide snapshots of how ML might be applied at different points in the capability lifecycle to support informed decision making. These scenarios are based on consultation the Scientific Test & Analysis Techniques Center of Excellence (STAT COE) has performed with DOD programs.

Scenario 1: A research effort exploring the fundamental science of hypersonics wanted to understand what factors influenced a binary response critical to hypersonic flight performance to be able to predict what factor combinations would result in different outcomes. Additionally, the team wanted to validate a long run-time computational fluid dynamics (CFD) model. The team conducted wind tunnel testing, from which logistic regression and decision tree models were built to predict the binary response. The models were used as an alternative to the CFD model due to their fast runtime. This research aimed to influence the design of future aeronautical systems.

Scenario 2: A program encountered an issue during a developmental test event that prompted a new line of materials testing. A neural network and regression model were created from test data to model material behavior. Additionally, a physics informed model was created, though it was not able to interpolate between all factor levels due to physics assumptions. The machine learning models had a more flexible mathematical form which gave the ability to predict at any factor level, and they were able to make predictions much more quickly. The regression model was used to predict material behavior at untested conditions and was integrated with aerodynamic and thermal models to estimate impacts to system performance.

Scenario 3: A program using modeling & simulation (M&S) wanted to use a physics-based simulation to understand system behavior. The simulation had a long runtime and could only be tested at a limited number of input combinations. Machine learning (a Gaussian process model) was used to abstract from the limited number of runs to create a meta-model. The meta model enabled interpolation, the ability to predict model results at new input combinations,

significantly reducing runtime and providing insight into trends in system behavior.

Scenario 4: A deployed system was experiencing anomalies in breathing patterns. Time series anomaly detection was employed to analyze periodicity in breathing patterns and detect when an anomaly occurred, enabling identification and resolution of anomalies.

These scenarios show the breadth of ML application to T&E as well as some of the commonalities. ML can be applied across the entire lifecycle, from research and material-level modeling to understanding anomalies in operations. ML models can be trained from either physical data or simulated data to make new predictions. There are also applications outside of prediction, including anomaly detection and clustering.

In common among the scenarios, ML provides insight into some aspect of system behavior and it makes predictions quickly. A common insight is characterizing the effects of different factors. However, ML models vary in their ease of interpretation (and therefore insight) since ML models often place more emphasis on prediction quality than interpretability. For instance, regression models or decision trees tend to be more interpretable than a neural net, although a combination of these methods can provide a more comprehensive approach. Making predictions quickly is especially beneficial for modeling and simulation, where computationally intensive models can be substituted for a surrogate model or meta-model that mimics the behavior of the time-intensive model.

The next section describes the ML lifecycle, the process by which an ML model can be trained and continuously updated to meet decision support needs across the capability lifecycle.

Machine Learning Lifecycle

The lifecycle of a machine learning model can be broken into five phases, as shown in Figure 2. These phases span the ML modeling process, from determining if ML is an appropriate tool to ensuring the model continues to perform well after deployment.

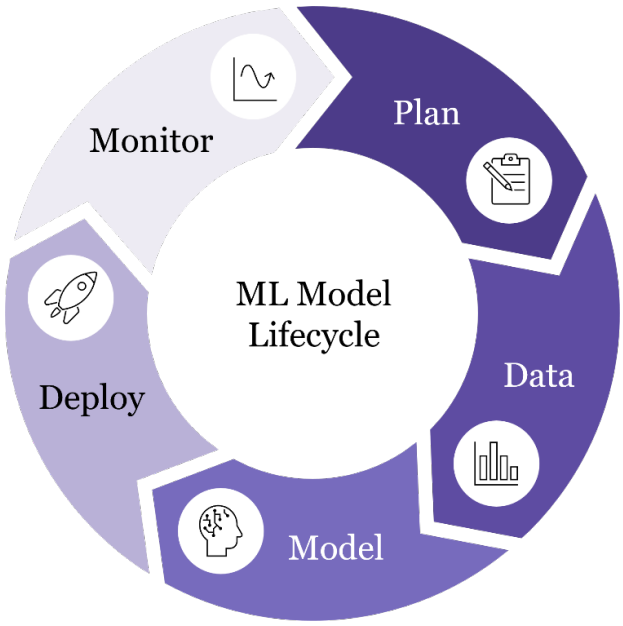

Figure 2
Machine Learning Model Lifecycle

Each phase will be discussed in detail in the subsequent sections. Activities during each phase are summarized in Table 1.

Table 1
Activites During Each ML Lifecycle Phase

Phase	Activities
Plan	• Define the problem • Identify the type of problem • Select appropriate ML algorithms
Data	• Collect data • Clean data • Exploratory data analysis
Model	• Train models with selected algorithms • Compare and tune models using cross-validation to select the best model • Test final model against data not used to train the model
Deploy	• Assign version number • Make available to stakeholders/users in appropriate environments • Use model to inform decision making and/or apply to new data
Monitor	• Monitor model performance • Monitor input and output data • Retrain model to maintain performance

Plan

The ML lifecycle is a carefully structured process that starts with defining a problem. While this may seem obvious, outlining a well-articulated problem statement can be challenging. The problem statement is critical in the plan phase, setting the direction for every subsequent phase of the ML lifecycle. Defining the right problem ensures the ML being developed aligns with the business objectives and focuses efforts on the right model and the right data (Artificial Intelligence +, 2024).

A well-defined problem influences the success metrics for evaluating the model. Success cannot be determined if the problem is not well understood. Additionally, a clear problem statement establishes well-defined goals and scope of the project. This ensures that ML is the right tool for the project. The type of problem dictates the type of algorithm needed. There are several common machine learning algorithms included in Table 2, each suited for specific types of tasks and applications.

Table 2

Machine Learning Algorithms for Different Tasks/Applications

Algorithm \ Task/Application	Numerical Prediction	Classification	Explanatory Modeling	Multi-Class Classification	Interpretability	Dimensionality Reduction	Non-Linear Relationships	Ensemble Learning
Linear Regression	•		•		•	•		
Logistic Regression		•		•	•	•		
Decision Trees	•	•		•	•			
Support Vector Machines (SVM)		•		•			•	
K-Nearest Neighbors (KNN)	•	•		•	•			
Naive Bayes		•		•	•			
Random Forest	•	•		•		•		•
K-Means Clustering						•		
Neural Networks	•	•		•			•	
Gradient Boosting Machines (GBM)	•	•		•			•	•

Table 2 is not a comprehensive list of ML algorithms. There are many other ML techniques to address various types of problems and data. Each algorithm has its strengths and weaknesses, making it important to choose the right one based on the specific application.

The type of ML model subsequently determines the type of data needed. Different stakeholders may have different perspectives on the project, but all should agree on the problem statement prior to moving into the data phase.

Data

Data serves as the foundation for any machine learning project, and its quality directly impacts the performance of the model. The first step in building this foundation is planning data collection. This involves identifying the type of data needed, determining how it will be sourced, and ensuring it aligns with objectives of the model. The STAT Process (Adams et al, 2022) can guide this planning phase to ensure effective test design and data collection.

The STAT Process provides a structures framework that includes planning steps such as understanding requirements, decomposing the mission and system, determining objectives, defining response, selecting factors and or levels, and identifying constraints. The STAT Process helps organizations ensure their data collections efforts align with test objectives and mitigates risk such as incomplete data or sampling biases.

In the context of T&E, data sources may include legacy system data, results from previously completed testing such as contractor or component test data, or data obtained from planned future tests. Effective data planning should also address potential limitations such as incomplete datasets, biases in sampling, or logistical challenges in accessing the required data. By addressing these considerations early, organizations can create a robust framework for acquiring data that meets both quantity and quality requirements.

Raw data is rarely ready for direct use in machine learning models, which makes data cleaning and processing a critical step. This involves addressing missing values, correcting inconsistencies, and transforming data into a format suitable for analysis. Techniques such as scaling, encoding categorical variables, and normalizing numerical data help standardize inputs and reduce the likelihood of errors during modeling. Proper data cleaning ensures that the model is trained on reliable and meaningful information, reducing noise and improving overall performance.

Exploratory data analysis (EDA) is an important step paired with data collection and data cleaning to understand the characteristics of the dataset and uncover insights that can guide feature engineering and model development. By visualizing distributions, correlations, and outliers, EDA helps identify patterns and potential issues, such as imbalanced classes or hidden biases, that could impact model performance. EDA also provides a deeper understanding of the problem space, ensuring that the data aligns with the goals of the project and any necessary adjustments can be made early in the process.

Model

The modeling phase of the ML lifecycle involves three primary steps: model training, model tuning and selection, and model testing. Despite the variety of ML models available, these steps are common across most modeling processes.

During model training, it's often advantageous to train a multitude of models using different algorithms. For example, logistic regression and decision trees are two algorithms frequently used to predict a binary response. Performance of different models can be compared to select the model best suited for a particular use case. The choice of model dictates how the model processes data during training.

Many ML algorithms require hyperparameters, settings that are defined by the modeler that influence the training process but are not derived from the data itself. Examples include the degree of polynomial in regression models or the learning rate in gradient descent for neural networks. Different hyperparameter configurations yield different model performances, emphasizing the importance of careful selection and fine tuning.

To select the best model from among different algorithms and hyperparameter choices, model performance is evaluated against a validation set. Importantly, models are not trained on all available data; instead, data is partitioned into subsets to facilitate evaluation against unseen information. The validation set can be used to tweak the model to perform better on unseen

data and is used to tune hyperparameters and select the best performing model. Cross-validation (Figure 3) is a robust technique for model validation. The data is split in many possible ways into a training and validation set, training a different model with each different training set. This procedure enables calculation of average performance metrics across iterations, ultimately ensuring the selected model generalizes well to new data. Since the training data changes depending on how the data was split, cross-validation provides a more robust evaluation of a model's performance on unseen data than a single split. Once the best algorithm and hyperparameters are selected based on average performance metrics, the model is retrained on the combined training and validation datasets to produce the final model. Stafford and Pai (2025) further discuss cross-validation along with common metrics for evaluating performance. It's important to note that the best metrics for evaluating model performance depend on the type of model (e.g., classifier versus regressor), as well as what risks are important to reduce for a particular use case.

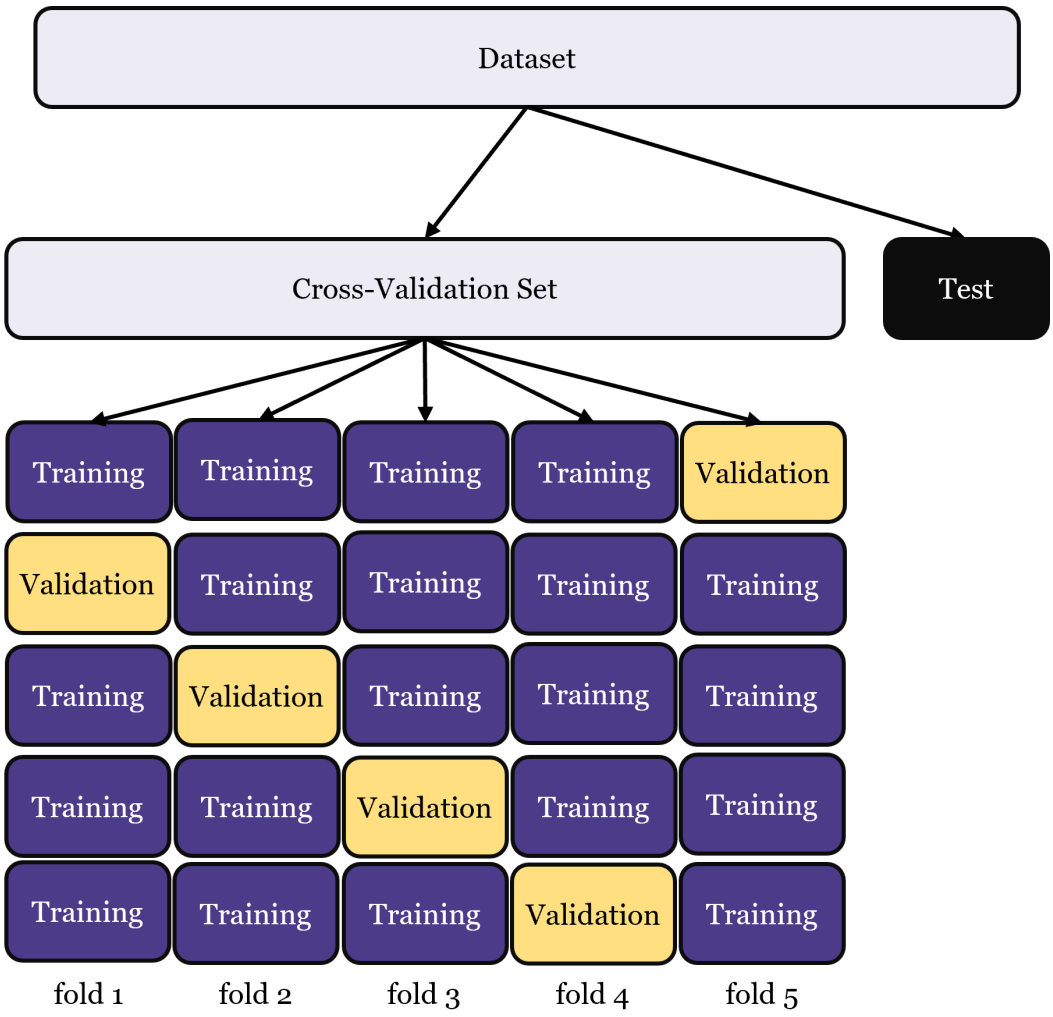

Figure 3
Five-fold (k=5) Cross-Validation

Final model evaluation is conducted using a test set, data not used during training or model selection. This independent assessment validates the model's performance and provides an evaluation of the final model's performance and generalization to unseen data. This evaluation

can support accreditation decisions. The test set is commonly held out from the initial data set and is not seen by the modeler or model until model selection and training is complete. Alternatively, the test set may come from a different data set entirely – this enables a truly independent evaluation of the model against data which may not be subject to the same data collection errors, operator bias, etc. Testing on an independent dataset may reveal new behaviors, such as operational failure modes, absent in the training data. When these behaviors arise, retraining the model may be necessary, effectively restarting the ML lifecycle.

Deploy

The deploy phase in the ML lifecycle focuses on transitioning models from development to their intended application. Deployment is not a static process, it requires integrating the model into its deployment environment, whether that is a controlled experimental setup, an operational system, or an individual's workstation. A well-planned deployment ensures the model remains, functional, adaptable, and aligned with its intended application.

Deployment typically initiates iteration through the ML lifecycle, where new data collected from the deployment environment is used to refine the model, ensuring the model remains relevant and accurate. For example, predictive outputs can help determine the most informative next steps, improving and optimizing outcomes over time.

The deployment environment must be tailored to the use case, considering factors like accessibility, computation resources, and connectivity. Deployment might involve directly embedding the model into existing systems or providing model outputs in a format that can be easily shared or integrated. By designing deployment strategies that enable flexibility and adaptability, organizations can ensure that machine learning models provide actionable insights while remaining resilient to operational challenges.

Monitor

ML models are often deployed with the expectation that they will perform as well as they did during model evaluation. The reality is that model performance inevitably will degrade over time. Therefore, the development of ML models cannot end after deployment. Although methods for monitoring and retraining are still evolving, having some sort of maintenance plan for an ML model beyond deployment will be crucial to its success.

Well-trained machine learning models can be useful once deployed, but their useful lifespan is typically very short. The model’s predictive performance will degrade over some time from changes in the environment that violate the model’s assumptions. A classic example would be when there is seasonality in the data, like increases in flight demand during holidays. Another example would be the performance of a model that predicts house prices when the COVID-19 pandemic caused the significant and unexpected increase. This degradation of model performance over time is known as drift. Drift can be mitigated by model retraining, but this requires determining when and how to retrain the model effectively.

There are two primary methods available for detecting or inferring when drift occurs: monitoring model performance and monitoring changes in data. Monitoring a model performance metric over time, as shown in Figure 4, allows direct quantification of the amount of degradation. When the metric falls below a certain threshold it would trigger the need to retrain the model. Once retrained, the model performance improves as it adapts to the current data to provide more accurate outputs.

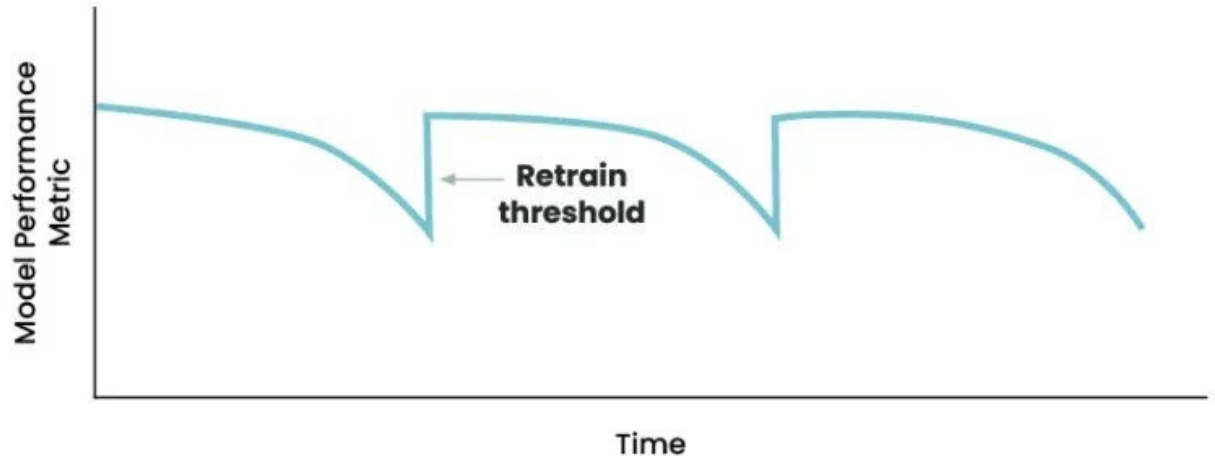

Figure 4
Trigger-Based Retraining at a Given Threshold

Note. Figure from Diliberto, L. (2021, May 6). *When Should You Retrain Machine Learning Models?*. phData. https://www.phdata.io/blog/when-to-retrain-machine-learning-models/

Alternatively, a substantial change in the distribution of targets or input data would imply model drift. Even without checking model performance, models typically degrade when data changes. This is because the model reacts to the input data it receives—changes to those inputs will most likely influence performance. A common approach is to periodically compare the distributions of current target or input data to the distributions of data used in training. When a difference between these distributions is found, the model should be retrained with the new representative data.

Some of the methods for detecting differences in distributions assume that the distribution of the current data is not a function of time. In other words, these methods are for stationary, or batch processing, data, and do not measure patterns over time. Furthermore, the outcomes of these methods could vary if different samples of data are used (i.e., different time windows of data). Detecting a change in data trends over time requires something more like statistical process control (SPC). SPC should be used for monitoring data drift when the data is continuously streamed to the model over time. If the streaming data is aggregated such that the timestamps are removed, then it's likely the most important component of drift has been removed. Statistical process control is one possible solution for monitoring changes in data, and its application to machine learning drift detection is an ongoing area of study (see Zamzmi et al., 2024).

When drift is detected through performance monitoring or changes in distributions, a good first step is to consider if there has been a change in the data quality (i.e., how the data is collected). If the data collection and processing is acceptable, then retraining the model can mitigate the drift. Determining the new training data depends on the ML problem. If a data stream is dynamic, then the model should be retrained frequently, replacing older data with the new data as it becomes available. If an old dataset is simply not representative of the new environment, then the entire dataset should be replaced. However, sufficient data must be available for retraining, otherwise there may be little to no effect on model performance. An additional

consideration for retraining is the cost to retrain.

The development and deployment of ML models should be regarded as a continuous process that includes model retraining so the model continues to perform as it was intended. To illustrate this process, the next section provides a case study describing activities that might occur during each ML phase for a particular modeling effort.

Case Study

The hypersonic international flight research experimentation (HIFiRE) program provides a practical case study for the integration of machine learning in a rigorous, test-driven environment (Natoli et al., 2020). The objective for using ML was to predict and characterize the start/unstart boundary of a hypersonic inlet through hypersonic wind tunnel testing. The HIFiRE team and the STAT COE leveraged ML methods to support test planning, experimental execution, and continuous evaluation. Framing this process within the ML lifecycle; Plan, Data, Model, Deploy and Monitor, highlights its relevance to a continuous ML model evaluation strategy.

Plan

The HIFiRE effort began with a well-defined problem; to accurately identify the inlet's start/unstart boundary during hypersonic wind tunnel tests and validate a computational fluid dynamics (CFD) model. Historical data analysis identified that inlet pressure was closely related to the probability of start or unstart. Thus, the choice was made to model both start/unstart (binary response) and inlet pressure (continuous response). Specifically, a decision tree and logistic regression were chosen to model start/unstart while a decision tree and linear regression were chosen to model the inlet pressure. A binary decision tree enables identification of cut-off points between start/unstart, while a logistic regression model can predict the probability of start/unstart for any given factor settings. Further, for the inlet pressure, the decision tree could be compared against the start/unstart decision tree to see if there were differences in factor impacts. Since SME input would be used to categorize the conditions into started and unstarted, the models could potentially have slightly different outputs. Additionally, the linear regression allowed smooth modeling of the continuous pressure, which could be used to identify the pressure at the cut off between start and unstart. All these models could be compared to choose the best predictive model.

Data

A key challenge was to design a testing strategy that would efficiently gather data to train the models and meet testing objectives. The team used a scaled wind tunnel model tested at NASA Langley's Mach 6 Wind Tunnel, where various factors such as, angle of attack (AoA), the Reynolds number (a non-dimensional flow parameter which could be controlled by varying the wind tunnel pressure), angle of sideslip (AoS), and sweep direction, were manipulated and studied. Due to the constrained wind tunnel environment with hard-to-change factors such as the Reynold's number, traditional DOE as well as adaptive methods that only choose one test point at a time were unpractical. To address these risks, the STAT COE proposed a more efficient DOE approach, a split-plot experimental design that balanced the hard to change factors (Reynolds number) with easier to change factors (AoA, AoS, and sweep direction). Inlet pressure data and binary start/unstart conditions were recorded for each test point.

Model

After completing an initial set of test runs in the wind tunnel, the collected data was used to train a set of machine learning models that included logistic regression, general linear regression, and decision trees. The logistic regression model predicted the likelihood of the inlet's state (start/unstart) based on AoA, Reynolds number, AoS, and sweep direction, and focused on identifying the decision boundary where the model's uncertainty was highest (predicted probability ~0.5). This allowed the team to direct further testing efforts to the most informative regions of the design space, regions of mixed results. By incorporating both binary and continuous responses, the team was able to refine their predictions and develop robust models for both characterization and validation purposes.

Key aspects of the modeling process included:

- Model Comparison and Selection: For prediction purposes, logistic regression was selected for binary classification and general linear regression for the continuous pressure data.
- Validation: High R-squared values (~0.95-0.96) demonstrated strong model fits for pressure data.
- Decision Tree Development: The decision tree model was used to provide an interpretable flowchart for classifying the inlet state based on AoA and other factors.

Deploy

The deployment phase in the HIFiRE program followed an iterative process designed to balance rigorous test coverage with real-time model refinement. This phase quickly iterated through the entire ML lifecycle and involved deploying the model to plan the next test points, data collection, model updates, and validation, ensuring continuous improvement of predictive accuracy throughout the test campaign.

Using the logistic regression model, the Prediction Profiler in JMP (JMP Statistical Discovery, 2025) was used to identify the next set of test points based on the uncertainty boundaries, (predicted probability ~0.5). The profiler helped visualize the relationships between the input factors and predicted outcomes, guiding the selection of the next set of test conditions that would be most informative. This process was performed iteratively to plan a series of wind tunnel test points, with new test points planned after updating the model with the previous iteration of data collected.

Monitor

Following each series of test points, the collected data were analyzed to evaluate the performance of the ML models. The logistic regression and general linear models were assessed for accuracy, and their predictions were compared against observed inlet states. Metrics such as the receiver operator characteristics (ROC) curve and confusion matrices were used to evaluate classification performance.

The ML models were updated with the new data, recalculating coefficients for the regression

models and adjusting the split points in the decision tree. This update allowed models to refine their predictions based on the latest observations from the wind tunnel experiments. The updated models were then deployed again for the next test series.

This iterative process continued with the updated models being used to predict outcomes for newly collected test data. With each new test series, the models were evaluated on how well they predicted the observed results. If the models demonstrated satisfactory performance with the new data, their parameters were further refined, with coefficients and decision boundaries adjusted to reflect the most recent insights.

This continuous feedback loop ensured that the deployment and monitor phases were not merely the application of a static model but rather a dynamic, evolving process where the models adapted and improved throughout the testing campaign. By integrating real-time data updates and model refinement, the HIFiRE team was able to maximize the effectiveness of each test cycle while maintaining a structured approach to test point selections and validation.

The insights gained from this process highlights the importance of monitoring ML modeling performance over time, particularly in dynamic testing environments.

Long-Term Deployment and Monitoring

After the conclusion of wind tunnel testing, the final model was deployed to the HIFiRE team. Since the model was developed in the JMP software, the JMP files were provided to the team, and the team was trained on how to implement the models in future testing. Since the developed models could be expressed with mathematical formulas, they could also be provided to users without access to the JMP software. These models were then used as alternatives to the long-runtime CFD model, enabling the team to answer the same questions in a much shorter time span. While this concluded the STAT COE integration with the HIFiRE team, the model could continue to be monitored with trigger-based retraining when new data was collected during future testing campaigns.

Conclusion

Machine learning (ML) provides a unique ability to continuously update models with data across the capability lifecycle. With applications from clustering to surrogate modeling, ML can flexibly meet a variety of T&E objectives. For an ML model to be well-trained and remain trustworthy over time, a model must iterate through each phase of the ML lifecycle: plan, data, model, deploy, and monitor. The HIFiRE application demonstrates how a model can quickly iterate through this lifecycle within a single testing series, as well as the potential for continued use following deployment to a larger test team. Continuous application of ML models in T&E ultimately enables earlier and better-informed decision making, helping to deliver capabilities at the speed of relevance.

References

Adams, W., Divis, E., Jones, N., Kershner, C., Lazarus, J., Marshall, M., McBride, A., Mott, T., Natoli, C., Oimoen, S., Provost, K., Ramert, A., Sgambellone, A., Sigler, G., Theimer, J., Truett, L., & Weeks, C. (2022). Test Planning Guide. https://www.afit.edu/images/pics/file/Final%200930_Test%20Planning%20Guide_2_2%20(1).pdf

Artificial Intelligence + (2024). *The Complete Machine Learning Lifecycle: A Step-by-Step Guide for 2025*. https://www.aiplusinfo.com/blog/the-complete-machine-learning-lifecycle-a-step-by-step-guide-for-2025/

Collins, C. & Senechal, K. (2023). Test and Evaluation as a Continuum. The ITEA Journal of Test and Evaluation, 44(1). https://itea.org/journals/volume-44-1/test-and-evaluation-as-a-continuum/

Diliberto, L. (2021, May 6). *When Should You Retrain Machine Learning Models?* phData. https://www.phdata.io/blog/when-to-retrain-machine-learning-models/

JMP Statistical Discovery (2025). *Overview of the Prediction Profiler*. https://www.jmp.com/support/help/en/18.1/index.shtml#page/jmp/overview-of-the-prediction-profiler.shtml

Natoli, C. W., Sigler, G. S., Guldin, S. A., Harman, M. J., & Ahner, D. K. (2020). Design of Experiments in Characterizing Hypersonic Flow on a Wind Tunnel Model. *The ITEA Journal of Test and Evaluation*, *41*(3), 166–175.

Ng, A. (2018). *Machine learning yearning: Technical strategy for AI Engineers in the Era of Deep Learning*. https://www.deeplearning.ai/machine-learning-yearning/

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Stafford, C. & Pai, R. (2024). *Cross-Validation for Machine Learning Models*. Best Practice. Scientific Test & Analysis Techniques Center of Excellence.

Zamzmi, G., Venkatesh, K., Nelson, B., Prathapan, S., Yi, P. H., Sahiner, B., & Delfino, J. G. (2024, February 12). *Out-of-Distribution Detection and Data Drift Monitoring using Statistical Process Control.* arXiv.org. https://arxiv.org/abs/2402.08088