FINAL REPORT

Optimizing Facility Operations by Applying Machine Learning to the Army Reserve Enterprise Building Control System

Installation Energy and Water Projects
EW19-5300

January 2025

B. Ford, E. Wendel, T. Yoder, V. Chandan

PNNL-33364

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACRONYMS AND ABBREVIATIONS

AMI	advanced metering infrastructure
AMP	Amy Metering Program
ARIMD	Army Reserve Installation Management Directorate
ARNet	Army Reserve Network
ASHRAE	American Society of Heating, Refrigerating, and Air-Conditioning Engineers
ATO	Authority to Operate
AUC	area under curve
BAS	building automation system
BCS	building control system
CAC	Common Access Card
CART	Classification and Regression Trees
C.E.M.	Certified Energy Manager
CONUS	continental United States
CPU	central processing unit
CSS	Customer Support System
CTO	Certificate to Operate
CY	calendar year
DoD	U.S. Department of Defense
DOE	Department of Energy
DQA	data quality analysis
DPW	Directorate of Public Works
DRL	deep reinforcement learning
EBCS	Enterprise Building Control System
EMIS	energy management information system
ESTCP	Environmental Security Technology Certification Program
EUI	energy use intensity
FCNN	fully connected neural network
FDD	fault detection and diagnosis
FEMP	Federal Energy Management Program
FISMA	Federal Information System Management Act
FOUO	For Official Use Only
FTP	file transfer protocol
G-6	Office of the Army Chief Information Officer
GPU	graphics processing unit
HOA	hand-off-auto

HQ	Headquarters
HVAC	heating, ventilation, and air conditioning
IEEE	Institute of Electrical and Electronics Engineers
ISD	Integrated Surface Database
IT	information technology
LEED	Leadership in Energy and Environmental Design
MB	megabyte
MDMS	Meter Data Management System
ML	machine learning
MMBtu	million British thermal units
MNIST	Modified National Institute of Standards and Technology
MPC	model predictive control
MSSQL	Microsoft SQL Server
NCEI	National Centers for Environmental Information
NIST	National Institute of Standards and Technology
NMBE	normalized mean bias error
NOAA	National Oceanic and Atmospheric Administration
OA	outdoor air
OLS	ordinary least squares
OUO	Official Use Only
PII	Personal Identifiable Information
PNNL	Pacific Northwest National Laboratory
PNSO	Pacific Northwest Site Office
PR	public relations
RD	Readiness Division
ReLU	Rectified linear unit
RNN	recurrent neural network
ROC	receiver operating characteristic
SVM	support vector machine
SVR	support vector regression
UCPM	Use Case Prioritization Matrix
USACE	U.S. Army Corps of Engineers
USAR	U.S. Army Reserve
VAV	variable air volume
VM	Virtual Machine
XGB	Extreme Gradient Boosting

ACKNOWLEDGEMENTS

This demonstration is the result of numerous people working to identify the best opportunities for applying machine learning to U.S. Army Reserve buildings. The authors wish to acknowledge the ESTCP Energy and Water Program lead and committee for their guidance and leadership. This could not have been possible without the partnership with the U.S. Army Reserve, led by Mr. Paul Wirt, who has championed innovation in energy management across the Army complex. The authors would also like to thank the following current and former PNNL staff for collaboration on this project: Bill Chvala Jr, Sarah Newman, Xiaoli Duan, Tim Salsbury, Stephanie Johnson, Eric McKay, Hayden Reeve, Osman Ahmed, Majid alDosari, James Goddard, Abinesh Selvacanabady, and Lucy Huang.

ABSTRACT

Introduction and Objectives

Thousands of U.S. Department of Defense (DoD) buildings have building automation systems (BASs) and/or advanced meters. Although these systems have a wealth of data, performance optimization requires time and expertise to review and act on that information. Machine learning (ML) can provide automated and actionable insights to controls operators. This demonstration implemented proven ML methods on the Army Reserve Enterprise Building Control System.

Technology Description

ML refers to algorithms that "learn" from data and improve their performance on a given task over time. In the buildings domain these tasks range from predicting future energy consumption, to identifying operational issues before faults occur, to optimizing control decisions. To learn, ML requires input data, which – for buildings – typically consists of *instrument data* such as energy consumption data and subsystem controls information such as set-point temperatures, and *context data* consisting of information such as the physical location of the building, the area of the building, and the weather. ML models use the relationships learned from the input data to make predictions with new, previously unseen, data.

Performance and Cost Assessment

The team was able to investigate and successfully implement the following ML use cases: labeling consumption data as anomalous or non-anomalous; baseline whole-building load prediction (unknown fault status); fault detection (validation not possible); and site prioritization for energy-related projects. Due to the constraints of the project, interventions were not able to be implemented during the demonstration; therefore, assessments of operational cost savings and maintenance avoided could not be performed.

Implementation Issues

Throughout the demonstration, the primary obstacle to successful use case investigation was a lack of access to complete, high-quality data. In general, ML algorithms require large volumes of input data to produce models that have high predictive accuracy. For many USAR buildings, however, the minimum required data were simply unavailable.

Publications

The project has been presented at two leading national building conferences[1] and two additional publications to peer-reviewed journals[2] are currently in preparation.

[1] American Society of Heating, Refrigeration, and Air Conditioning Engineers (ASHRAE) Annual conference (June 2021) presentation "Applying Machine Learning to Enhance Building Performance at US Army Reserve Centers". National Institute of Building Sciences Building Innovation Conference (September 2021) presentation "Applying Artificial Intelligence to Buildings with Imperfect Data".

[2] A paper describing the "Challenges to Applying ML to Existing Building Energy and Controls Data at Scale" and a paper describing "ML for Buildings: A Use Case Perspective" (both in Draft to be submitted).

EXECUTIVE SUMMARY

INTRODUCTION

Thousands of U.S. Department of Defense (DoD) buildings have building automation systems (BASs) and/or advanced meters. These systems have the potential to enable significant reductions in energy use through identification of operational issues. Yet in many buildings, the systems are often not actively monitored or leveraged to improve operations. The building operators do not have time to sift through the BAS lists of alarms, alerts, and settings. The BASs do not prioritize the most important actions needed to be taken on a given day. In some cases, the way BASs are set up causes inefficient operations and premature failure of equipment. Advanced approaches to energy management can contribute significantly to meeting energy-reduction goals and reduce the mission impact of downtime associated with building system failure that could have otherwise been avoided.

The U.S. Army Reserve (USAR) has an extensive source of operational information through its Enterprise Building Control System (EBCS), which integrates control system data and data from the Army's Meter Data Management System (MDMS) into a common analytic platform. Although EBCS has a wealth of building and system-level data, optimization of performance normally requires a trained controls operator to review and act on that information. Machine learning (ML) methods can provide automated and actionable insights to controls operators.

A team from Pacific Northwest National Laboratory (PNNL) with expertise in building controls, ML, and technology assessment executed this demonstration, and hosted by USAR from 2019-2022.

OBJECTIVES

The goal of the demonstration documented in this report was to show that common, industry-standard machine learning (ML) algorithms can be applied to USAR buildings data to automate the identification of operational issues and energy-savings opportunities. The demonstration aimed to investigate a variety of use cases, including prediction of baseline electricity demand, automated fault detection and diagnosis, and controls optimization methods. Further, the demonstration aimed to quantify cost savings associated with measures identified by the ML algorithms.

TECHNOLOGY DESCRIPTION

ML refers to algorithms that "learn" from data and improve their performance on a given task over time. In the buildings domain these tasks range from predicting future energy consumption, to identifying operational issues before faults occur, to optimizing control decisions. To learn, ML requires input data, which – for buildings – typically consists of *instrument data* such as energy consumption data and subsystem controls information such as set-point temperatures, and *context data* such as the physical location of the building, the area of the building, and the weather. ML models use the relationships learned from the input data to make predictions with new, previously unseen, data.

Figure ES.1 illustrates how different ML technologies can be applied to buildings data to produce an optimized solution suite. The first step is to collect and process the input *instrument*

data and *context data.* Data *integration and preprocessing* involves tasks such as making the data accessible in a format and location such that ML algorithms can easily access them and resolving any data quality concerns such as data gaps and noise.

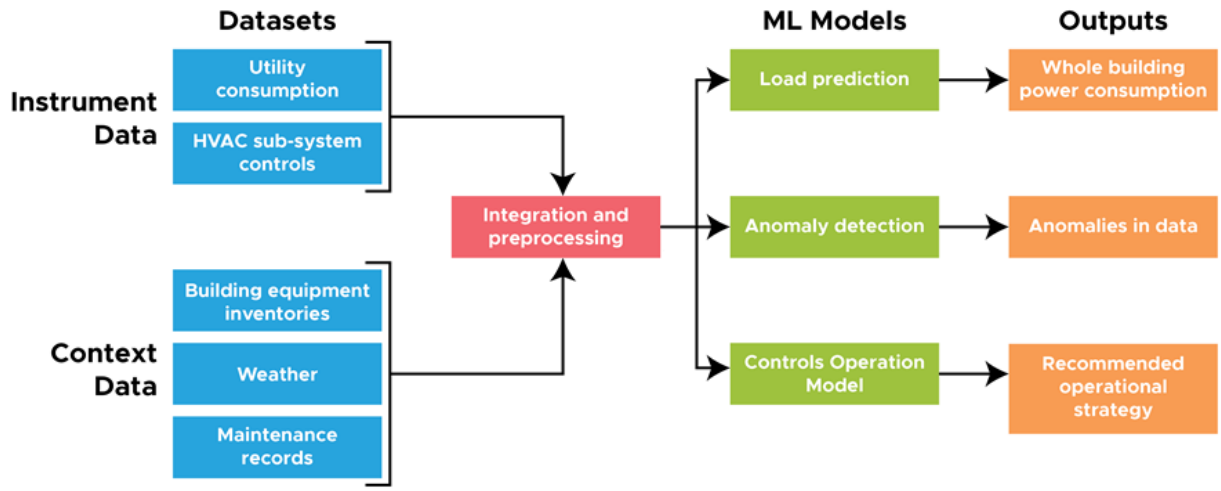

Figure ES.1. Overview of Machine Learning Methods for Buildings

The ML models can then be trained using the processed input data to make predictions for various use cases. Some of the common use cases for ML in the operation of commercial buildings are (note that not all of these use cases were investigated in this demonstration):

1. **Baseline consumption modeling:** Predict power consumption (under business-as-usual conditions/without optimization of controls) at the building level for a specified time period in future.
2. **Unsupervised fault/anomaly detection:** Use a baseline consumption model to predict baseline consumption. Then compare the baseline prediction with real/measured consumption. If the deviation is more than a certain use-prescribed threshold, classify the performance as anomalous.
3. **Labeling consumption data as anomalous/non-anomalous:** Use cluster analysis to segregate (training data only) into anomalous and non-anomalous. However this is only applied on training data for the purpose of autonomously labeling data as anomalous and non-anomalous (in the absence of expert opinion). Such labeled data are needed as an input for the use case below.
4. **Supervised fault/anomaly detection:** Use data labeled as anomalous vs. non-anomalous (either labeled by an expert or by using the methodology in the above use case) as training data to train classification ML algorithms for fault detection.
5. **Baseline control-oriented modeling:** Train regression ML models that predict energy use and indoor environment variables (e.g., temperature). The inputs are control knobs and exogenous variables.

6. **Model-based control optimization:** Use models trained in the above use case to optimize control settings to achieve a prescribed objective (reduce energy/cost) over a prescribed period of time. Can be implemented as a lookup table.
7. **Model-free control optimization**: Train control policies directly (without the aid of a model, hence different from model-based control optimization use case) to optimize them to achieve a prescribed objective (reduce energy/cost) over a prescribed period of time. Implementation on an actual building is outside the scope of the project; therefore, the model developed in above use case or a pre-existing simulation model can be used as a proxy to demonstrate the proof-of-concept.
8. **Transfer Learning**: Transfer models learned on one "source" building to another "similar" "target" building. Both baseline consumption model and baseline control-oriented models can be examined for transferability.
9. **Supervised Fault Diagnosi**s: Use data labeled in two layers: (1) anomalous vs. (2) non-anomalous. Anomalous data is labeled with the cause of the fault. Train classification ML algorithms on such data to diagnose (specify) the type of fault. Data can be generated from simulation models if field data required for the analysis are challenging to obtain.
10. **Predictive Maintenance**: Analyze the health of the subsystems in a building using data from equipment and estimate probability of failure.

PERFORMANCE ASSESSMENT

Given the available data during the demonstration performance period, the team was able to investigate the following ML use cases:

- labeling consumption data as anomalous or non-anomalous
- baseline whole-building load prediction (unknown fault status)
- fault detection (validation not possible)
- site prioritization for energy-related projects.

Models trained to label data as anomalous or non-anomalous and to predict baseline loads generally performed well. For the baseline energy consumption modeling use case, the project team tested a number of leading regression algorithms to test which would produce the highest accuracy results across all the buildings with sufficient input data. The regression scores of the different models for each of the buildings with sufficient data are shown in Figure ES.2. All the models exceeded the success criteria metrics for accuracy, but the random forest model was ultimately chosen for implementation in the demonstration deployment application due to its high accuracy across a range of inputs.

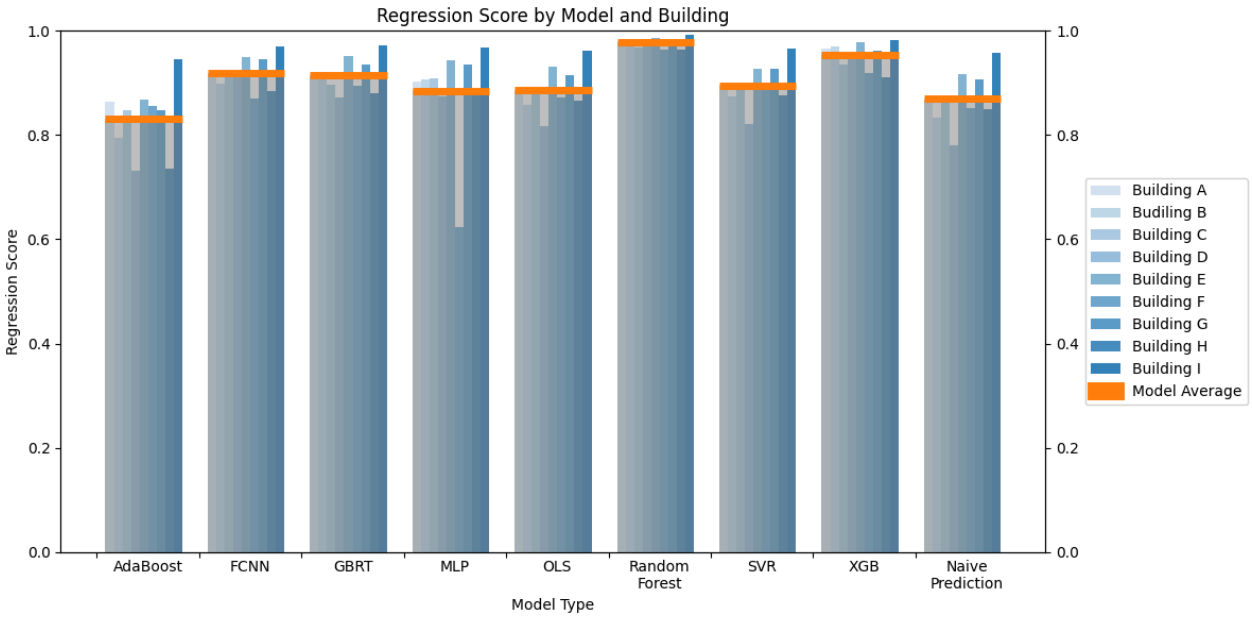

Figure ES.2. Comparison of Model Performance (full set regression score) for All Buildings that Had Sufficient Data

Baseline prediction is a precursor to fault detection, wherein a model is trained to predict demand patterns under normal operations; then, deviations in actual demand from predicted demand can be flagged and investigated as potential faults. As discussed further below in Implementation Issues, it was not possible to verify whether the baseline prediction models were trained on fault-free data, which undermined the reliability of subsequent fault detection models. Finally, a separate random forest regression model was trained to prioritize USAR sites for energy-related projects. The model was used in a virtual Installation Energy and Water Plan (v-IEWP) performed by PNNL for USAR to identify sites at which energy conservation measures should be prioritized.

Despite the deployment and data challenges, the demonstration accomplished the following tasks:

- Successfully demonstrated anomaly detection, baseline modeling, and fault detection use cases (with caveats).
- Created a Use Case Prioritization Matrix for evaluating which ML applications are appropriate for an end user given data availability, user priorities, and other key decision metrics.
- Developed an extensible Python-based data integration platform that allows users to readily merge controls, meter, weather, and other data sources and feed integrated data into an ML modeling pipeline.
- Documented opportunities and constraints that shape the ML solution space within the USAR context and recommendations for improvement to extend the scope of ML applications.
- Produced a user-friendly demonstration web app (currently hosted on the PNNL network) that allows users to build their own baseline prediction models on USAR data and generate graphs and compare prediction to actual consumption.

COST ASSESSMENT

This demonstration highlighted the fact that the installation costs to integrate machine learning models into data and decision systems, and timeline can be much greater than originally planned, in large part due to the data preparation costs. The specific calculation varies depending on the use case, but the economic benefits component would relate back to one or more of the performance objectives, namely analysis effort, building energy use, and system maintenance.

Due to the constraints of the project (discussed in the following Implementation Issues section), interventions could not be implemented during the demonstration; therefore, assessments of operational cost savings and maintenance avoided could not be performed.

IMPLEMENTATION ISSUES

Several challenges that were not well understood at the outset of the demonstration caused delays and created barriers to successful use case investigation and deployment of ML algorithms.

Deployment: When the demonstration began, the USAR was planning to transition a number of Army Reserve Network (ARNet)-hosted systems to the Microsoft Azure cloud platform. Among the systems planned for transition was the Enterprise Building Control System (EBCS), which integrates control system data across dozens of USAR buildings into a common analytic platform. To date, however, that transition has not occurred; consequently, throughout the demonstration there has been uncertainty regarding the ultimate deployment environment and the cybersecurity requirements associated with deployment. Because of the ongoing uncertainty about deployment requirements, the ML tools developed during this demonstration were not deployed to EBCS directly but instead are hosted on the PNNL network. The team will continue to work with USAR after the demonstration concludes to plan a future EBCS deployment.

Data: Throughout the demonstration, the primary obstacle to successful use case investigation was a lack of access to complete, high-quality data. In general, ML algorithms require large volumes of input data to produce models that have high predictive accuracy. For many USAR buildings, however, the minimum required data were simply unavailable. The most prominent example of data unavailability occurred in the Meter Data Management System from June 2018 to September 2019, during which time almost no USAR meter data were recorded due to a networking connectivity issue. However, even when high-resolution utility meter and control systems data were available for a building, there was rarely a long enough period of temporal overlap between sources that the data could be integrated and used to train a baseline prediction model.

An additional barrier to investigating fault detection use cases was the facility maintenance data provided by the Customer Support System (CSS), an enterprise system for tracking work orders at USAR sites. The CSS tickets did not provide sufficient detail to definitively identify fault causes or durations. Hence, it was not possible to build fault-free baseline prediction models or validate the predictions of fault detection models. When the fault status is unknown—i.e., we do not know when and if there are faults present in the training data—it precludes generating baseline models that are capable of positively identifying faults, although it does not eliminate the possibility of creating baseline models entirely. Baseline models can be generated for the buildings with sufficient data, but we cannot say whether they reflect the ideal fault-free

operation of the building. We only know the models can predict the current operation, faults included.

For example, if a building has a fault of a stuck open terminal damper, which has caused an increase in the total building energy consumption, that fault would be present in the training data for the baseline model, and the model's baseline power prediction would include the higher energy consumption caused by the fault. Any faults in the available training data will be incorporated into the models' whole-building consumption predictions. Because we do not have any information about the faults in the training data (or even know if they exist), we cannot teach the model anything about those faults or use the model to identify existing faults.

Ultimately, only nine USAR buildings had adequate data for investigating ML use cases beyond data quality analysis and baseline prediction. Table ES.1 summarizes the data requirements and actual data availability associated with three ML use cases investigated during the demonstration. The table is an application of the Use Case Prioritization Matrix, which was developed for the demonstration.

Table ES.1. ML Use Case Prioritization Matrix Applied to USAR Buildings Data

Use Case Description	Use Case Category	Data Attribute	Data Required	Actual Data Available
Labeling consumption data as anomalous or non-anomalous	Pre-processing anomaly detection	Measurements	Utility use data (hourly or smaller resolution): • Gas consumption • Electricity consumption • Water consumption Outdoor environmental data (hourly or smaller resolution).	Data requirements met.
		Data Volume	No strict minimum requirement.	Data requirements met.
		Data Quality	Gaps are tolerable.	Data requirements met.
Baseline consumption modeling	Fault detection, energy benchmarking	Measurements	Utility use data (hourly or smaller resolution): • Gas consumption • Electricity consumption • Water consumption Outdoor environmental data (hourly or smaller resolution).	Outdoor environmental data available at hourly resolution from NOAA across all of CONUS. Where utility meter data is available, it is available at hourly or smaller intervals.
				Utility meter data are only available for a portion of the buildings that have control data. Many buildings with utility meters do not have control data available. Where utility meters are present

Use Case Description	Use Case Category	Data Attribute	Data Required	Actual Data Available
				at buildings, electricity is the most common meter type. Gas and water utility meter data are not consistently available.
		Data Volume	At least 1 year. Multiple years are preferred.	At least 1 year of overlapping meter and controls data available for just 9 of ~70 buildings. Multiple years not available.
		Data Quality	Must be fault-free. Sparse gaps are tolerable (a few data points missing).	Fault status unknown, not possible to say whether data are fault-free. Depends on building, but large gaps (a few hours up to a few months) are present.
Unsupervised fault/anomaly detection	Fault detection	Measurements	Step 1 (baseline consumption model training): fault-free data as required to train baseline consumption model.	Step 1: Same data availability constraints as described in the baseline consumption modeling use case, fault status unknown.
			Step 2 (fault detection): consumption measurements for time period for which fault detection is to be performed (can be a real-time stream).	Step 2: N/A, cannot progress past Step 1 without fault-free data.
		Data Volume	Step 1 (baseline consumption model training): At least 1 year. Multiple years is preferred.	At least 1 year of overlapping meter and controls data available for just 9 of ~70 buildings. Multiple years not available.
		Data Quality	Step 1 (baseline consumption model training): must be fault-free.	Step 1: Fault status unknown, not possible to say whether data are fault-free.
			Step 2 (fault detection): can have faults. Both: Sparse gaps are tolerable (a few data points missing).	Step 2: N/A, cannot progress past Step 1 without fault-free data.

The initial results of this demonstration are promising, but long-term changes in data acquisition and storage would be needed to extend the applications of ML. These changes include the following:

- Implement metadata or semantic models (e.g., standardize point categorization and mapping across all buildings in EBCS).
- Assure that complete point histories are saved in long-term data storage, and restore points that go off-line as soon as possible to prevent large data gaps.
- Assure that histories of the rule-based fault alarms are generated by EBCS and saved in long-term data storage.
- Improve maintenance recordkeeping to include specific detail about when faults occur and what specifically occurred (e.g., terminal damper stuck open).
- Continue to prioritize EBCS integration in buildings with advanced meters. Connecting the metering data directly to the EBCS was shown to improve data quality. (Note that USAR is already doing this.)

1.0 INTRODUCTION

This section provides a general overview of this demonstration for the Environmental Security Technology Certification Program (ESTCP), including background on operations at Army Reserve facilities, legislative and policy drivers relating to the demonstration, objectives of the demonstration, and a discussion of activities planned at the outset of the demonstration that were not completed.

1.1 BACKGROUND

Thousands of U.S. Department of Defense (DoD) buildings have building automation systems (BASs) and/or advanced meters. These systems have the potential to enable significant reductions in energy use through identification of operational issues. Yet in many buildings, the systems are often not actively monitored or leveraged to improve operations. The building operators do not have time to sift through the BAS lists of alarms, alerts, and settings. The BASs do not prioritize the most important actions needed to be taken on a given day. In some cases, the way BASs are set up causes inefficient operations and premature failure of equipment. Advanced approaches to energy management can contribute significantly to meeting energy-reduction goals and reduce the mission impact of downtime associated with building system failure that could have otherwise been avoided.

Like many DoD commands, the USAR can access whole-building meter data through the Army's Meter Data Management System (MDMS), yet these data are largely underutilized for real-time energy management. At the inception of this project, the USAR was analyzing meter data quarterly for 364 buildings and identifying optimization measures using a rules-based approach. This process is very effective but requires highly skilled analysts to sift through the data and manually prioritize actions, which is time-consuming and expensive.

The USAR has another extensive source of operational information through its Enterprise Building Control System (EBCS), which integrates control system data into a common analytic platform. At the end of calendar year (CY) 2019, 60 buildings across the U.S. were scheduled to be connected to the EBCS, enabling remote monitoring and control of building systems.

Although the EBCS will have a wealth of building and system-level data, optimization of performance normally requires a trained controls operator to review and act on that information. Machine learning (ML) methods can provide automated and actionable insights to controls operators. Once fully operational and consistently trending data, the USAR EBCS will offer a prime platform for implementing ML methods that provide both diagnostic and predictive intelligence to building operators. Although EBCS was still under development during the demonstration performance period, the demonstration attempted to implement proven ML methods on the USAR enterprise system to automate the identification of operational issues and energy-savings opportunities through system optimization.

1.2 DRIVERS

The following drivers relate to this demonstration and support the methods and goals:

- Executive Orders (E.O.s): E.O. 13834 of May 17, 2018, Efficient Federal Operations (83 FR 23771, revoked) was in effect during the start of the project and required agencies to achieve

and maintain annual reductions in building energy use and implement energy efficiency measures that reduce costs. E.O. 14057 of December 8, 2021, Catalyzing Clean Energy Industries and Jobs Through Federal Sustainability (86 FR 70935), defines a Federal Sustainability Plan and sets out a range of goals to deliver an emissions reduction pathway, including through building efficiency.

- Legislative Mandates: The Energy Policy Act of 2005 requires electric meters to be installed at the building level. The Energy Independence and Security Act of 2007 requires comprehensive energy and water evaluations, including an assessment of building retro commissioning opportunities. The Energy Act of 2020 added to the requirements for water metering, evaluations and efficiency.
- DoD Policy: Strategic Sustainability Performance Plan – Inform decisions, optimize use, assure access, build resilience, drive innovation. The DoD Utilities Meter Policy (January 2021) requires each Component to meter energy and water use to provide installations with the information necessary to improve resilience and mission assurance, increase utility systems reliability, and optimize resource use.
- Service Policy: Army Directive 2020-03 – Installation security policy requiring that critical missions be supported with power and water for a minimum of 14 days in the event of service disruption. Reducing loads and equipment failure leads to increased reliability. Army Directive 2014-10 specifies an advanced metering policy in accordance with the DoD Utilities Meter Policy, requiring the use of advanced meters to quantitatively determine how much energy and water the Army is using. The Army is currently targeting 60% coverage for electric, natural gas and water. The Army Climate Strategy, released in 2022 has a 50% reduction in net greenhouse gas pollution by 2030 and aims to get to zero emissions by 2050. Line of Effort 1.4 requires all land holding commands to implement installation-wide building control systems by 2028.
- Specifications: American Society of Heating, Refrigerating, and Air Conditioning Engineers (ASHRAE); Leadership in Energy and Environmental Design (LEED); Institute of Electrical and Electronics Engineers (IEEE); and International Code Council codes (International Mechanical Code, International Plumbing Code, International Energy Conservation Code, etc.) require building control systems (BCSs) and optimized operations.

1.3 OBJECTIVE OF THE DEMONSTRATION

The objective of this demonstration was to show that the application of ML methods to building data can automate the identification of operational issues, leading to cost and energy-savings opportunities.

Pacific Northwest National Laboratory (PNNL), the directly funded performer of this demonstration, tested several different ML methods and applications of ML on buildings data. The initial demonstration plan was focused on using ML methods to automate tasks like fault detection and diagnosis (FDD) and controls optimization, which have traditionally required highly skilled building analysts to perform them.

The findings from this demonstration create value for DoD in several ways:

1. The methods validated and deployed on USAR building data in this demonstration could be leveraged to deliver cost and energy savings.
2. The methods will be transferable to other DoD buildings for which the same or analogous data sources are available. At a minimum, this means that as additional USAR buildings are connected to EBCS, the ML models developed for this demonstration could be adapted and retrained to identify operational issues in those buildings as well. These algorithms could be extended further to data sources for other Components and services, provided that the appropriate input data were available. Technology transfer activities planned for this demonstration are described in Section 8.0.
3. The demonstration will generate a record of the opportunities and constraints that shape the ML solution space within the USAR. ML methods present the possibility of automating operational issue identification at a large scale; however, the methods are only as good as their input data. Throughout the demonstration, the project team discovered significant limitations in the quality and availability of the EBCS and MDMS data sources.
4. A key output of this demonstration is a thorough description of data quality and availability issues, in order to inform future deployments (see Section 9.0 for additional details). Results from this demonstration will help DoD understand the scope of what is possible given potential data limitations. This information could be used to inform future decisions about how and where to invest resources in data collection, storage, and communication technology; for example, whether to deploy new advanced meters or dedicate more resources to maintaining existing ones. It could also be considered in a future procurement process for building data analytics software that includes ML-based capabilities.

DoD has an interest in supporting research into applied ML across a wide range of activities. It is currently investigating the use of ML in self-driving vehicles and drones, among other applications. This demonstration extended that research into the area of building systems and produced results that can be communicated throughout DoD to justify investment in further research and deployment of applied ML technologies.

1.4 ACTIVITIES PLANNED BUT NOT COMPLETED

At the outset of the demonstration, the team planned to perform several activities that ultimately could not be completed during the performance period. Various issues contributed to this outcome, most notably an overall lack of meter and controls data that delayed model development and led the team to request a 9-month no-cost extension for the demonstration. (Data quality and availability issues are discussed in greater depth in Section 9.0) Even with the extension, the team was not able to perform certain planned activities, which are described below. Nevertheless, PNNL has a multi-year support contract with the USAR and plans to continue with the joint deployment of Control Score and the ML models developed under this demonstration to EBCS after the ESTCP demonstration performance period ends.

Implementation of recommended actions

Once an operational issue is identified, the appropriate operations and maintenance staff must be notified of the issue and the recommended corrective measure. Examples of such measures include implementing building scheduling, optimizing equipment setpoints, restoring HOA (hand-off-auto) switches to auto mode, replacing miscalibrated sensors, and replacing faulty

components such as stuck valves and dampers. Implementation of the recommended measures would result in additional cost savings derived from decreased building energy use and reductions in unscheduled system maintenance.

As discussed further in Section 6.0, data quality issues limited the team's ability to investigate fault detection use cases. In particular, facility maintenance tickets did not provide sufficient detail to definitively identify fault causes nor durations. Hence, it was not possible to build fault-free baseline prediction models nor to validate the predictions of fault detection models. Consequently, the team is not currently able to make ML-based recommendations regarding corrective measures. This report includes a discussion of the data quality issues that limit the development of ML-based fault detection models and recommendations for long-term changes in data acquisition and storage would be needed to extend the applications of ML.

Cost models

The PNNL project team planned to develop models to quantify the costs and benefits of implementing ML algorithms for operational issue identification. The cost models, empirically informed by data generated by the demonstration itself, would be used to characterize costs and benefits at the level of an individual building as well as at scale for an entire portfolio of buildings. Marginal cost was expected to decrease with increasing scale. In addition, the demonstration planned to estimate the cost and energy savings resulting from the implementation of recommended measures.

As noted above, the team is currently unable to make ML-based recommendations regarding corrective measures. As a result, no measures have yet been implemented at USAR facilities based on ML-modeled fault detection predictions and no empirical cost savings data have yet been collected.

2.0 TECHNOLOGY DESCRIPTION

The technology demonstrated in this project is machine learning (ML) applied to utility meter and automation controls data at USAR buildings across the country.

2.1 TECHNOLOGY OVERVIEW

Machine Learning Overview

"Machine learning" refers to algorithms that learn representations of data. This representation can be used to perform tasks such as *regression* (the prediction of a continuous value), *classification* (the prediction of a category), and *optimal strategy* identification (the prediction of value and optimal policy). These methods have been applied to solve problems in image recognition, voice detection, visual analysis, and autonomous control and optimization. In a buildings context, ML tasks could include baseline energy use prediction (regression) and FDD (classification). ML can be applied at multiple scales, ranging from the entire building to subsystems (e.g., air handlers, boilers, chillers).

ML is a mature technology with a diverse range of industrial, commercial, and research applications. ML algorithms are integral to the functionality of many services and products on the market today, including internet search engines, voice recognition technologies, and robotics. DoD is currently investigating the use of ML in self-driving vehicles and drones, among other applications. Research in this field of computer science has been ongoing for 70 years, although widespread industry adoption was generally hindered by limitations in computational capacity and data availability for algorithm development until the late 1990s. Today, the availability of highly parallel computing systems (e.g., graphics processing units [GPUs] and tensor co-processors) reduces cost barriers and provides optimized tensor calculations to speed up deep neural network training. Similarly, the use of state-of-the-art sensing devices facilitates the deployment of cost-effective sensor networks able to capture data at unprecedented volumes and level of detail.

Since Google announced a substantial reduction in the amount of energy used for cooling their data centers by using an ensemble of deep neural networks on a substantial amount of sensing points, there has been a growing interest in applying deep learning to building energy optimization. In particular, the application of deep reinforcement learning has shown promising results in this field (Li et al. 2017; Mocanu et al. 2018). Diverse ML approaches have also been used to detect and identify faults in building systems (Kim and Katipamula 2018). Automatic fault detection is a mature technology that facilitates the design of preventive maintenance plans.

Machine Learning for Buildings

Figure 1 illustrates how different ML technologies can be applied to buildings data to produce an optimized solution suite. The first step is to collect and process different types of *instrument data* such as energy consumption data and subsystem controls information such as set-point temperatures, and *context data* such as the physical location of the building, the area of the building, and the weather. Data *integration and preprocessing* involves tasks such as making the data accessible in a format and location such that ML algorithms can easily access them and resolving any data quality concerns such as data gaps and noise.

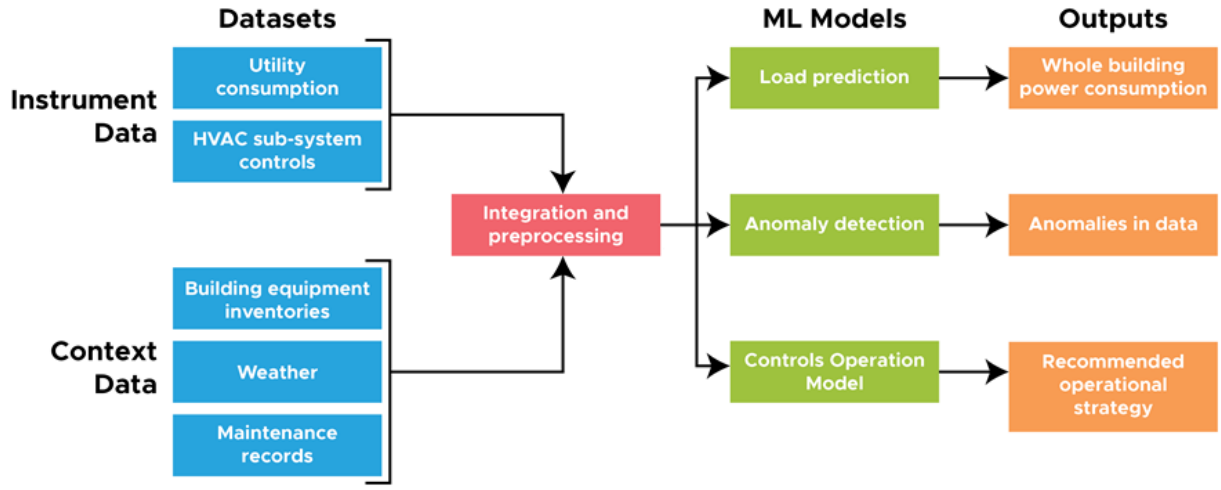

Figure 1. Overview of Machine Learning Methods for Buildings

An important next step is to separate the data into two categories: one corresponding to fault-free conditions and the other corresponding to known faults, with the underlying faults also known. Regression algorithms can then be applied to fault-free consumption data to establish baseline consumption values through *load prediction* models. A comparison of predicted baseline consumption with real-time measured consumption data can be used to *detect anomalies* or faults. For instance, if the measured consumption at a given time instance deviates significantly (based on an appropriate threshold) from the baseline consumption, that indicates that there might be a fault in the system.

Once the presence of a fault is indicated, the next step is to diagnose the fault. This can be done by training classification algorithms on labeled faulty data. The training can be done off-line (on historical data), resulting in pre-trained fault diagnosis algorithms. These algorithms identify signatures for specific faults in the data. For example, a stuck variable air volume (VAV) damper is likely to exhibit a different response on measurements such as delivered air flow or indoor temperature than a miscalibrated thermostat. Such pre-trained fault diagnosis algorithms could then potentially be applied online to diagnose faults in real time.

In addition to FDD, ML algorithms can also be used to *optimize control decisions*. Fault-free data can be used to build dynamic models that map outputs of interest (e.g., power consumption, energy cost, indoor temperatures) to control inputs. These models can be used to train algorithms that optimize the control inputs based on user/operator-specified criteria such as minimizing the cost of energy while maximizing occupant comfort.

The project team planned to investigate solutions in all three areas—baseline prediction, FDD, and controls optimization—subject to the opportunities and constraints discussed in Section 9.0. As explained further in that section, limitations in data quality and availability meant the team was able to implement models for baseline prediction and anomaly detection, but not controls optimization. The open-source ML libraries used for this demonstration are industry standard, with scikit-learn, TensorFlow, and Keras all originating at Google. The demonstration applied existing algorithm implementations (i.e., ML libraries) to building data on the USAR EBCS

platform; hence, no new ML algorithm development was required. See Appendix A: for a more detailed discussion of ML methods.

2.2 ADVANTAGES AND LIMITATIONS OF THE TECHNOLOGY

ML has proven to be very effective in many areas, such as speech recognition, natural language processing, and image classification. Recently, deep learning has emerged as one of the most popular and powerful concepts in ML. The buildings domain is not left untouched by the current wave of ML, especially deep learning. ML is well suited for the buildings domain, given the complexity of nonlinear operations and the increasing abundance of data. However, several challenges also exist when applying ML that are unique to the buildings domain. Some factors that have contributed to the popularity of ML methods in recent times are listed below under advantages and disadvantages along with explanations of how they apply to the buildings domain.

- Advantages
  - *Availability of data:* Widespread deployment of both traditional and newly emerging, low-cost sensor technologies has led to the ubiquitous presence of data in several domains. Society has also entered the era of "big data." The exponential growth of big data has enabled researchers to develop and train ML algorithms, which are powerful in revealing nonlinear and complex patterns. This has resulted in the ability to mine more and more useful and actionable information. The buildings domain has not been untouched by this trend. The increasing use of BASs to manage buildings, especially modern buildings, has resulted in much more available data that spans long time windows (e.g., multiple years). While data are certainly increasing in "breadth" (time window), they are still limited from a "depth" perspective; sensing is limited to measuring certain key quantities of interest, such as zone temperatures and building energy consumption. Hence, there is a need to develop ML methods that can navigate this data depth challenge in buildings.
  - *Higher performance computation:* Significant improvements in hardware affordability have accelerated the adoption of ML in various domains. Advances in hardware (i.e., central processing units [CPUs]) and new computational models (i.e., GPUs) have created a large opportunity for ML to handle a high degree of parallel operations and perform matrix multiplications efficiently. In particular, GPUs provide a parallel architecture that has been especially powerful when applied to the types of calculations needed for neural networks. CPUs have also made advances for better parallelization and more efficient matrix computations. However, these developments have not yet penetrated the buildings domain, and the computational infrastructure used within buildings is still quite limited. New computational models (e.g., cloud computing), which provide inexpensive access to high-performance computational resources using pay-per-use business models, can potentially be leveraged.
  - *Advanced algorithms:* The advent of algorithms that are of higher performance and lower in computational resources makes it easier to test and develop advanced algorithms, because they are not limited by the hardware constraints of the past. These advances in learning algorithms have made training deep architectures feasible; it has been shown that such advanced ML models can provide performance that is superior or comparable to

state-of-the-art methods. For example, deep learning algorithms have been shown to perform better than human experts in various fields (e.g., some learning algorithms have become world champions at a variety of games, from Chess to Go). These advanced algorithms appear to hold promise for improved performance in the buildings domain as well.

- *Software libraries:* Deep ML algorithms can be very easily built and customized today using available software libraries such as *TensorFlow*, *Caffe*, *Torch*, and *Theano*. Various ML algorithms are made available as "black boxes" within these libraries. The advent of such software packages is unlocking new possibilities that were unimaginable a few years ago. These libraries allow powerful learning algorithms to be built with a few lines of code. The availability of these libraries also enables researchers to build, implement, and maintain ML systems for a variety of real-world domains. This is promising in the context of building systems, where such libraries can be used to build control applications in less time and with less effort on top of the cloud computing infrastructure discussed earlier.

- Disadvantages

  Despite the recent successful applications of ML to the buildings domain, certain challenges and limitations remain as described below:

- *Data requirements:* ML imposes strict requirements on data quality and quantity. For instance, data should be of high quality (with minimal gaps) and a sufficiently large amount of data should be available—at both temporal and spatial scales—so that the models trained on them are accurate. Data that are deficient in quality and quantity are likely to result in poor representations. Data quality, availability, and fault labeling are expected to be a challenge for this demonstration. To address these limitations, where feasible, unreliable or missing data were detected and corrected using methods described in Section 5.7. It is important to recognize, however, that calibration and connectivity issues pose challenges not only to this demonstration, but to system users in general. One key output of this demonstration is a thorough characterization of data quality and availability, as discussed in Section 9.0.

- *Scalability:* A major limitation of ML is that models that were trained on one system are not easily generalizable to other systems, especially in the buildings domain. This creates a challenge with regard to scaling the ML algorithms to a large set of systems. For example, a model trained on one building can be quite inaccurate when applied to a different building. In recent years, transfer learning-where some attributes of a model learned on one system are re-used to build a model for a different but related system-has been proposed as a potential solution to address this problem. Without transfer learning, ML can require a large amount of training data to achieve satisfactory accuracy. Although transfer learning has been demonstrated to address the generalizability and scalability challenges of ML models in other domains, it has been investigated only to a limited extent for buildings. To address this limitation, in this project we attempted to use transfer learning between buildings of similar types. However, a full-fledged demonstration of transfer learning is beyond the scope of the project.

- *Rigorous feature selection requirement*: A key benefit offered by deep learning, which has made it very popular in domains such as computer vision and computer games, is that

it overcomes the need to identify input features rigorously. However, feature optimization is still an important requirement in literature related to ML for buildings. This can be attributed in part to the observation that the networks were not deep enough.

- *Lack of generalizability*: There is a need to develop a generalized modeling framework that can work across buildings, which has not been attempted in this domain so far. In the absence of such a framework, the modeling problem for each building becomes a tedious exercise in which the appropriate architecture and parameters must be determined separately for each building, requiring time-consuming effort. Lack of standardization in BAS implementations across buildings plays a large role in this problem, because different systems are measured and trended at different buildings and a variety of inconsistent point naming and metadata conventions are used across the building automation industry (see Section 9.0 for further discussion). The lack of generalizability impedes the ability to use these models for large-scale investigations, such as those at the level of a distribution grid involving tens or thousands of buildings. In this project, we were able to use energy consumption data from multiple buildings to implement a generic baseline prediction ML model that represents the portfolio of buildings. However, a large-scale demonstration of a generic ML model for buildings at the subsystem controls level was beyond the scope of the project.

2.3 DEMONSTRATION PLATFORM: EBCS

In 2014, the Army Reserve Installation Management Directorate (ARIMD) started an initiative to implement an USAR-wide EBCS. The objective of the initiative is to connect existing building-level control systems to a common EBCS server on the Army Reserve Network (ARNet) to greatly expand the capability and impact of the building-level control systems. The EBCS Working Group was formed with representatives from Readiness Divisions, Mission Support Command, ARIMD, G-6, and PNNL spanning fiscal years 2014 through 2019.

A “pilot” phase was conducted from May 2016 through August 2017, successfully connecting 40 buildings across 3 regions and 16 states to the central EBCS server. The sites represented 8 different controls vendors, and more than 28,000 data points were successfully integrated. A total of 141 controls optimization measures were identified, with potential savings estimated at $137K per year for all 40 buildings (Koehler et al. 2017). An additional 40 buildings were added to the EBCS during Phase 2, which began in October 2017 (see Figure 2). Additional phases for adding buildings to EBCS are currently under way.

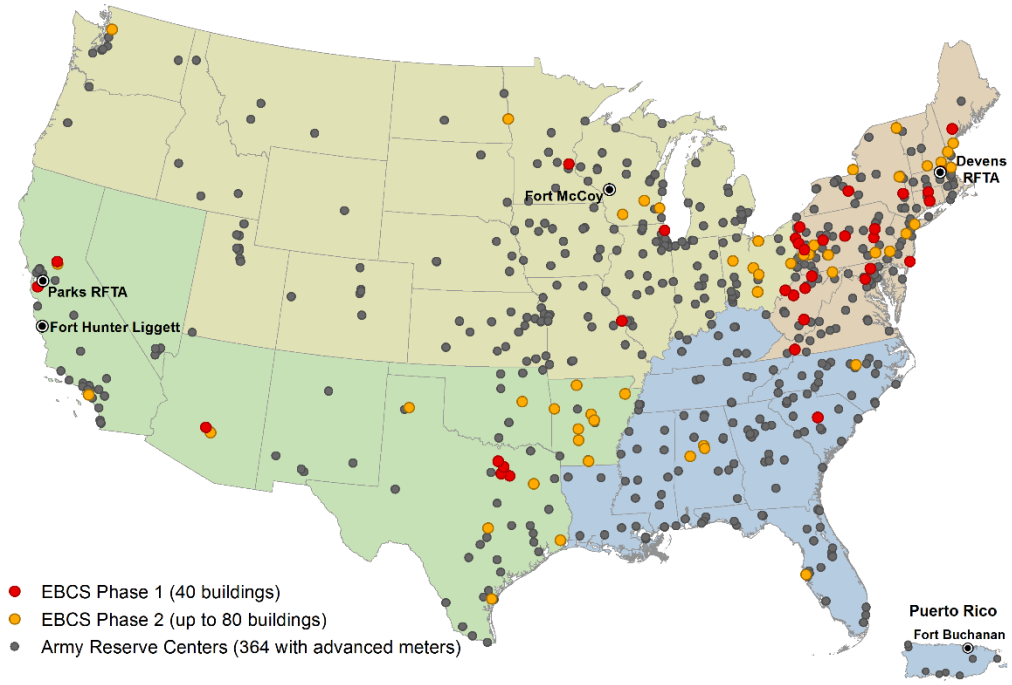

Figure 2. Reserve Centers on the Enterprise Building Control System

The EBCS is available to USAR personnel with appropriate permission through the ARNet, including remote Virtual Private Network access. The primary user interface (see Figure 3) includes region- and site-level navigation, building-level graphics, system-level graphics, scheduling screens, and trend data viewers. In fiscal year (FY) 2019, the system, which is based on the Tridium Niagara Framework® platform, migrated to the N4 version, adding further analytic features and tools.

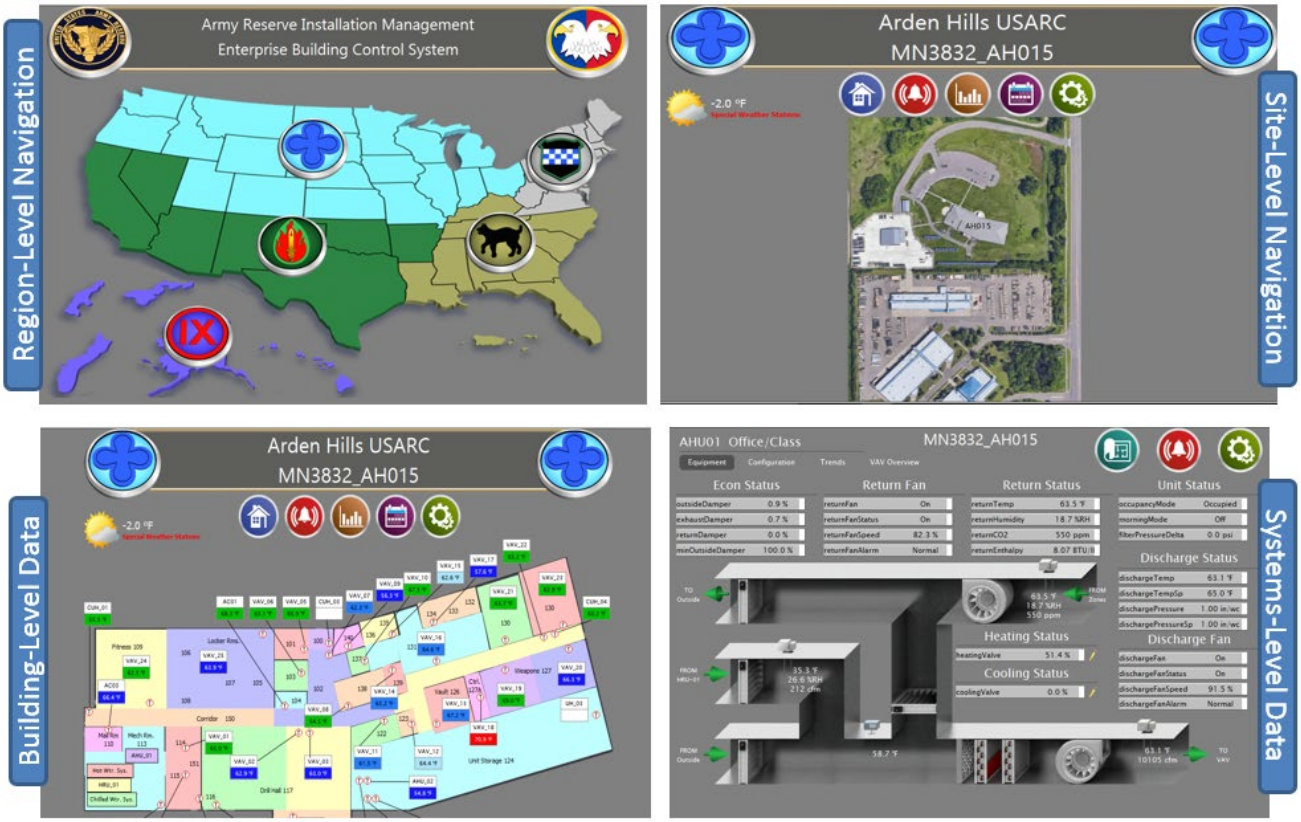

Figure 3. Enterprise Building Control System Interface

In the near term, the demonstration team plans to deliver the initial set of ML modeling tools for baseline prediction and anomaly detection to EBCS as part of a package deployment with the Control Score tool, which is also under development by PNNL. More discussion of this deployment is included in Section 8.0 on Technology Transfer.

3.0 FACILITY/SITE DESCRIPTION

Reserve Centers can range in size from as small as 20,000 ft^2 to as large as 200,000 ft^2. Generally, a Reserve Center encompasses between two and four buildings (see Figure 4), although several large sites have dozens of buildings. Although some sites have buildings with specialized functions, all sites have an administrative building, which includes an assembly hall, fitness room, and showers. A typical Reserve Center also has an organizational maintenance shop, an area maintenance support activity, or both. Often, sites include a storage building, heated storage building, or heated and cooled storage.

Reserve Centers are unique in their operation in that most have limited staff during the work week and an influx of soldiers on training weekends. It is common to have a staff of three during the week and host 300 people on the weekend. These facilities also host special training events at any time and have increased usage during summer months.

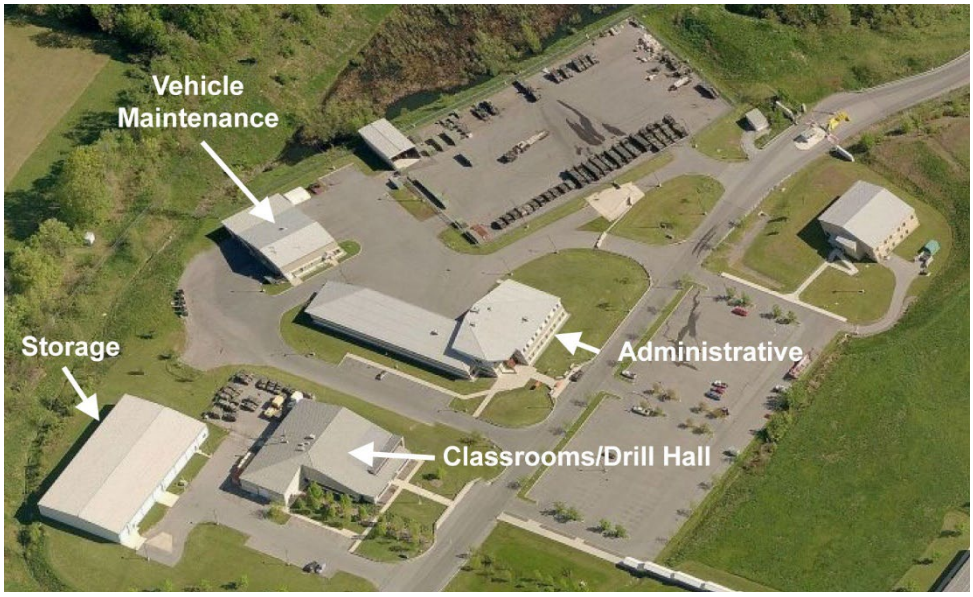

Figure 4. A Typical Army Reserve Center

3.1 GENERAL FACILITY/SITE SELECTION CRITERIA

The ML demonstration will be an integrated part of the EBCS system, connected to data sources as previously described. The actual ML tools currently reside on a secure test platform at PNNL until they are deployed on the EBCS. The actual buildings evaluated were based on data availability across all of the information systems. As of September 2019, advanced meters installed at most USAR buildings were not communicating with the Army MDMS server. USAR attempted to obtain missing data in early FY 2020 but was unable to due to the local storage limits on the devices. Due to this lack of metering data, initially only 10 Reserve Centers were included in the initial ML algorithm testing and selection phase. This sample size was feasible for the study, but not ideal. As of April 2020, meter communications began to be restored; however, data for most meters dating back to summer 2018 were irretrievably lost, meaning the demonstration required a no-cost extension while waiting for new data to accumulate in a

sufficient quantity to support ML model development. (See Section 9.0 for a characterization of meter data loss in the MDMS and EBCS.)

3.2 DEMONSTRATION FACILITY/SITE LOCATION AND OPERATIONS

The USAR buildings in the dataset are located in different regions across the U.S. and have similar equipment types. Table 1 describes 10 initial demonstration buildings with size, regional, and climate context.

Table 1. Characteristics of 10 Initial Demonstration Buildings

Building	City	State	U.S. Department of Energy Climate Zone	Heating Degree Days	Cooling Degree Days	Floor Area (ft^2)
Building 1	Mountain View	CA	3C	2,983	0	179,575
Building 2	Rockville	MD	4A	4,647	927	31,316
Building 3	Schuylkill Haven	PA	5A	5,395	551	23,129
Building 4	Bellefonte	PA	5A	5,577	559	26,662
Building 5	Cranberry Township	PA	5A	5,667	572	29,814
Building 6	Seagoville	TX	3A	2,161	2,424	60,158
Building 7	Seagoville	TX	3A	2,161	2,424	31,517
Building 8	Seagoville	TX	3A	2,161	2,424	27,013
Building 9	Grand Prairie	TX	3A	2,161	2,424	23,475
Building 10	Grand Prairie	TX	3A	2,161	2,424	44,618

Figure 5 illustrates the most common types of equipment in EBCS-connected buildings at the outset of the demonstration; VAV units, air handling units, and hot water supply pumps were the most common points in the EBCS. Each “number of records” represents a single type of system in the building, and there could be several points associated with the control of that equipment.

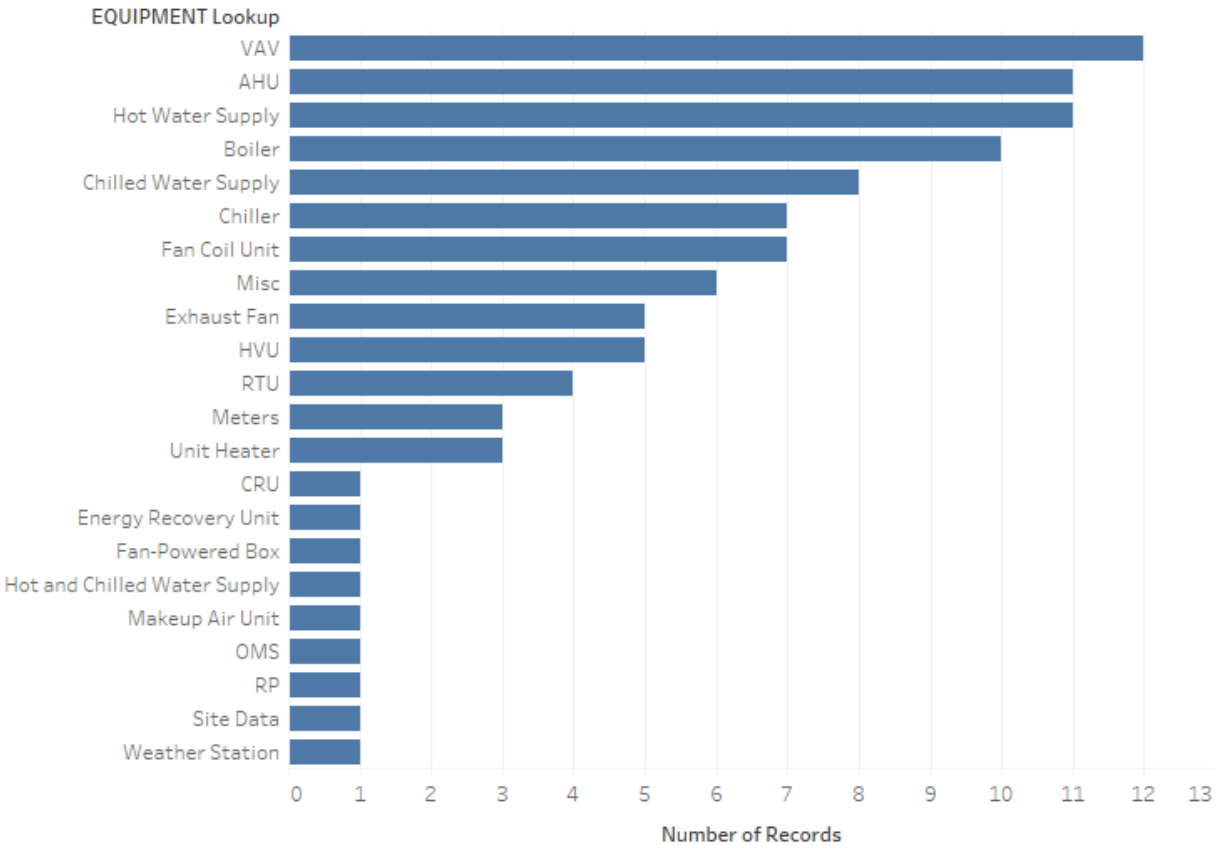

Figure 5. Equipment Types Present in a Set of 12 EBCS-connected Buildings

3.3 SITE-RELATED PERMITS AND REGULATIONS

Because this demonstration is occurring on existing systems, and only dealing with the data from the systems, no site-related permits are required. The EBCS has gone through the Risk Management Framework process and is approved for use on the ARNet. The programming language used for both the exploratory ML algorithm testing and the demonstration platform, Python, does not have explicit approval for use on the ARNet, although embedded versions (e.g., Python is embedded, or packaged, within geographic information system(GIS) platforms) do exist on the network.

3.4 PROPERTY TRANSFER AND DECOMMISSIONING

Because this demonstration is occurring on existing systems, and only dealing with the data from the systems, no property transfer is required. If USAR deems the demonstrated tool to be useful, a production-grade version of the application can be deployed on their network. Conducting the demonstration on the PNNL network allows USAR to evaluate the tool without having to incur the cost of a full deployment. USAR will own a copy of the ML code if they choose to continue to use it.

4.0 PERFORMANCE OBJECTIVES

4.1 SUMMARY OF PERFORMANCE OBJECTIVES

The performance objectives listed in Table 2 were designed to evaluate how well the ML methods identify opportunities for improving operational efficiency. The success of ML model performance is measured by the method's ability to detect faults over the baseline, and estimated reductions in building energy use.

Table 2. Performance Objectives

Performance Objective	Metric	Data Requirements	Success Measure
Fault Detection	Accuracy	Whole-building meter data and EBCS data.	Number of faults identified by ML compared to the baseline and manual evaluation
Building Energy Use	Energy use intensity ($MMBtu/ft^2$)	Whole-building meter data; building square footage; results from prior manual data evaluation tasks	Percent reduction (estimated) compared to the baseline and manual evaluation

The success of the demonstration relative to each of these performance objectives was dependent on the input data. As data exploration proceeded, the team discovered significant limitations in the quality and availability of the EBCS and MDMS data sources. Although it is not a formal performance objective, a key output of this demonstration is a thorough characterization of data quality and availability.

Because of data limitations and the current pre-deployment stage of the demonstration, key inputs for evaluating performance relative to these metrics were not available at the time of this report. With respect to fault detection, ground-truth validation data were not available either through maintenance records or through EBCS because histories for the rule-based fault detection algorithms used on EBCS are not stored. The demonstration team instead explored an approach of using alarm points that are automatically recorded by the system as a proxy for rule-based faults. The alarm points are generally rule-based and activate when a threshold is exceeded (e.g., a zone temperature is below the cooling set point). While the alarm proxy approach generated promising results (discussed further in Section 6.3), this unfortunately means that it was not possible to evaluate the success measure directly by comparing anomalies/faults detected by the ML model to those identified by the current approach in EBCS, nor to make ML-based recommendations regarding corrective measures. In the absence of specific recommendations, it is not possible to estimate savings from implementing corrective measures.

4.2 PERFORMANCE OBJECTIVE DESCRIPTIONS

The original performance objective descriptions from the Demonstration Plan are reproduced below for reference.

Fault Detection

Purpose: When equipment operates outside of intended parameters, it risks using excessive energy, shortening the equipment lifespan, and triggering component failures that can lead to shutdown operations. Fault detection algorithms can help facility energy managers detect when equipment is operating improperly and potentially avert such unwanted outcomes. Traditional fault detection algorithms are based on rule-based engineering calculations, and while they often perform well, they can sometimes lead to excessive alarms or overlook more complex system interactions. There is the potential for ML methods to supplement traditional rule-based methods by optimizing fault detection thresholds for more precise alarms, as well as by potentially detecting additional faults that are not explicitly accounted for in the existing FDD rule logic.

Metric: The metric for this performance objective is accuracy of fault detection, which will be calculated by comparing the number of faults detected by ML methods to baseline and manual evaluation. In addition, the team will characterize faults detected by the ML algorithms beyond those captured using the baseline rule-based approach.

Data: Input data for this performance objective are whole-building meter data from MDMS and building control system data from EBCS.

Analytical Methodology: The algorithms will use whole-building meter data from MDMS to predict baseline energy use and detect usage anomalies. Those anomalies will be cross-referenced with EBCS building controls data by the ML methods to identify operational signatures that correspond to faults. ML-flagged faults will be compared to rule-based alarms recorded by EBCS to calculate algorithm accuracy, and if no alarm exists, a potentially undiagnosed fault will be documented and investigated.[3]

Success Criteria: Percent accuracy of ML methods relative to the baseline, as well as distribution of undiagnosed faults.

Building Energy Use

Purpose: Progressive reductions in building energy use are mandated by federal legislation as well as DoD and service policy (see Section 1.2). Reducing energy demand increases energy security and resilience, thereby ensuring a greater capacity for the military to meet mission needs. The ML algorithms tested in this demonstration will attempt to automate the process of identifying certain operational issues and corresponding corrective measures that result in energy and cost savings.

Metric: The metric for this performance objective is building energy use intensity (EUI). It is calculated as the sum of energy used at a given building over a particular period of time divided by the total floor area of that building. It is usually reported as yearly MMBtu per gross square foot.

Data: Whole-building meter data will be provided by the MDMS and may be additionally validated with monthly utility data when available. Building floor area data available from MDMS or EBCS will be used to normalize energy use and calculate the EUI metric.

[3] The term "potentially undiagnosed fault" implies that a fault occurred that was not captured by the existing rule-based FDD logic.

Analytical Methodology: Because it is unlikely that the demonstration team will have the opportunity to recommend ML-informed energy conservation measures (ECMs) and analyze post-implementation data during the demonstration period, savings from recommended ECMs will be estimated. Energy-savings estimates will be based on standard engineering calculations for estimating savings from operational improvements or equipment retrofits. PNNL has developed specific engineering calculations for USAR buildings based on prior work. Cost savings derived from energy savings will be calculated based on the general cost model described in Section 7.0.

Success Criteria: Percent reduction (estimated) compared to baseline and manual methods.

5.0 TEST DESIGN

5.1 CONCEPTUAL TEST DESIGN

Hypothesis

The project team hypothesized that applying ML methods to building energy and control systems data could produce automated identification of operational issues, which can lead to energy and cost savings if the appropriate corrective measures are implemented.

Use Case Selection

ML encompasses a vast range of methods and potential use cases. The first phase of the process is to identify the relevant opportunities and constraints that shape the solution space. This step was critical to the success of the demonstration. Use cases were selected with consideration given to the following:

- Sponsor priority: What are the highest-priority use cases from the USAR's perspective?
- Value: Which measures produce the highest value with respect to the performance objectives of the demonstration?
- Data characterization
  - *Availability:* For which buildings are energy data, control system data, maintenance records, and other supporting data available? What is the time period spanned by the data? Which equipment/systems are represented in the data? What is the prevalence of faults/operational issues in the data?
  - *Quality:* Are there issues with data gaps and spikes, meter/sensor calibration, measurement accuracy and precision, or ground-truth validation that could compromise the analysis?
- State of research: Which building energy systems have been previously studied and which ML methods have been applied? Of those, which are relevant to this demonstration given the opportunities and constraints identified above?

Use case categories initially explored included the following:

- Data Cleaning: Utility meter and control system time-series data are often noisy and incomplete, and significant effort may be required to pre-process such data for use by ML algorithms. The problem of poor-quality data itself can be addressed using various ML methods, however; in fact, commercial developers of ML-based analytics for building energy management information systems have reported that most of their effort is focused on data quality-related use cases.
- Load Prediction: Accurate prediction of whole-building power consumption with business-as-usual operations provides a baseline in comparison to which anomalies can be detected. Baseline load prediction is an important prerequisite for many fault detection algorithms.
- Fault Detection: When equipment operates outside of intended parameters, it risks using excessive energy, shortening the equipment lifespan, and triggering component failures that

can lead to shutdown operations. Fault detection algorithms can help facility energy managers detect when equipment is operating improperly and potentially avert such unwanted outcomes.

- Controls Optimization: Advanced ML methods can assist building operators in identifying optimal control sequences that reduce energy use and cost.
  - Ultimately this use case was not feasible given the data quality issues.

Algorithm Testing

Once use cases were defined, the next step for the team was to proceed with algorithm investigation. This process was guided by the principle of simplicity: All other things being equal, a less complex model is preferable to a more complex model. If a heuristic provided adequate performance, there would be no need for an ML model; likewise, if a linear regression model performed similarly to a deep neural network, the linear model would be generally preferable (Zinkevich 2019). The more easily interpretable the model, the more likely its outputs are to gain acceptance by the end user.

The general framework for ML algorithm development is an iterative process between training, testing, and validating.

- Train: In training, most of a dataset is used to tune an algorithm for optimal performance. The performance is quantified through an appropriate loss function, which represents a metric that the ML algorithm is trying to optimize. For instance, in the problem of fault detection, an appropriate loss function could be the number of true positives.
- Test: The optimized algorithm from the training phase is then evaluated on a separate test dataset. Results from the testing can then inform algorithm selection and configuration choices in training-test cycles. Even within the same algorithm, several parameters— called hyperparameters—need to be tuned to optimize performance. For instance, in the case of recurrent neural network (RNNs), the activation functions, the number of layers, and the number of nodes are the underlying parameters that need to be optimized.
- Validation: After selecting the final model configuration in the test step, a third held-out dataset is used to evaluate model performance after initial testing. It is important that the dataset used for validation be non-overlapping with the training and test datasets. In other words, the ML algorithm should not have seen the validation dataset, in order to remove any potential bias. Often, the metric used to report the model performance is the same as the loss function used in the training and validation steps.

The performance of each algorithm was evaluated with respect to one or more of the external validation metrics detailed in Section 6.0. Algorithm performance depends on proper model specification as well as adequate data quality and availability. Poorly performing algorithms were documented alongside well-performing algorithms. As noted in Section 1.3, there is value in studying unsuccessful cases, because they may lead to a deeper understanding of which methods are appropriate for which use cases, how performance may be affected by data limitations, and other considerations.

5.2 BASELINE CHARACTERIZATION

Unlike the typical demonstration project, this demonstration did not require an initial period of baseline data collection. As stated above, the objective of the demonstration was to apply ML methods to *existing* building data sources. Sufficient energy and control systems data already exist to characterize baseline performance for multiple buildings on the EBCS. Baseline data collection continues automatically because the MDMS and EBCS record meter readings and sensor values at regular intervals. For each building studied, the baseline collection period spanned the period from the earliest date for which building energy and control systems data are available to the last point at which data are available. Data sources are described in further detail in the following section.

5.3 DATA MANAGEMENT

Data management involves handling the sources of data, equipment calibration and data quality issues, and instrument data processing needs.

Data Sources

Data collection and storage are handled by a set of functionally related but distinct data management systems that are owned and maintained by stakeholders external to the project. The project team was granted access to each of the systems and implemented a data pipeline from the external sources to an internal development environment.

Copies of external data sources are stored locally at PNNL. All project data were backed up automatically and routinely.

U.S. Army Reserve Enterprise Building Control System

The EBCS links buildings at USAR sites to a common analytic platform, allowing for remote monitoring and control of building systems. Using EBCS, an operator can view control system data for geographically distributed buildings from a single location (see Figure 6). In addition to real-time monitoring and control, EBCS is configured to record trend data for the hundreds or thousands of control system points in each building. Examples of control system points include valve and damper positions, supply and return air temperatures, and discharge fan speeds. Trends are recorded at varying intervals; some trends are configured to record values at pre-defined intervals, such as 1 hour or 15 minutes, while others record a value only when a state change is detected. These control system trends represent the core input data to the algorithms tested in this demonstration.

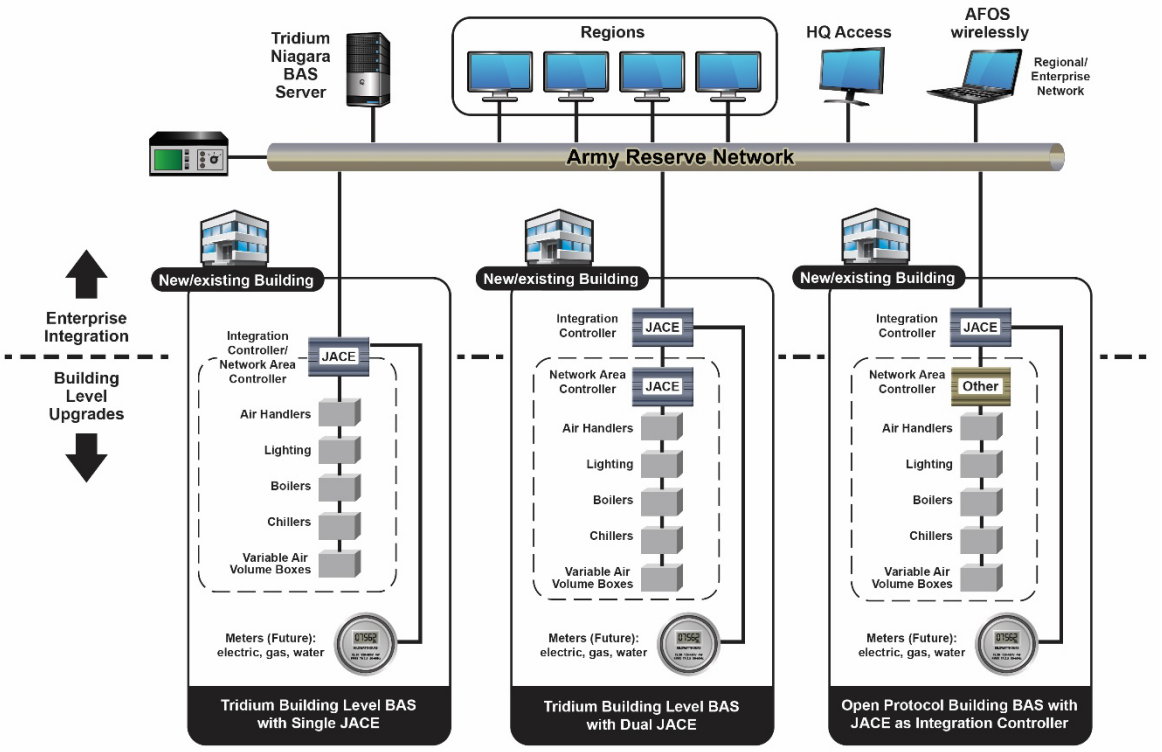

Figure 6. Schematic of EBCS Network Architecture

During the project, EBCS migrated from Tridium Niagara version 3.8 to version 4 (N4). At the end of CY 2019, 60 buildings across the U.S. were connected to the EBCS, enabling remote monitoring and control of the building systems. This migration process resulted in reporting gaps for buildings while they are migrated to N4. This and other data availability challenges are discussed in Section 9.1.

U.S. Army Meter Data Management System

The U.S. Army MDMS is an enterprise system for tracking the Army's energy and water use at facilities on installations around the world. The Army has installed over ten thousand advanced meters to track whole-building electricity, natural gas, and water use at high-priority buildings. Meters are generally configured to report usage at 15-minute intervals. As a Component of the Army, the USAR maintains a subset of those advanced meters at the building level, which are of significance to this demonstration. The Army's advanced meters report usage data to a central MDMS database server. The project team accesses that external MDMS database server to acquire energy use data for the buildings of interest in this demonstration. This demonstration used energy meter data as an input to the ML models, but not water data. Starting in FY 2019, the USAR began using the EBCS infrastructure to also transmit the MDMS meter data to the central Army MDMS server. So, the EBCS database also contains MDMS data.

Customer Support System

The USAR CSS is an enterprise system for tracking work orders at USAR sites. The CSS database contains information about orders for heating, cooling and air conditioning (HVAC)

system installation, maintenance, and replacement at USAR sites. Orders are dated, and in some cases estimated and actual service costs are recorded, usually at the building level. At the onset of the project, this dataset was hoped to be used to validate the algorithms' inferences about equipment faults and failures.

NOAA NCEI Integrated Surface Database

That National Oceanic and Atmospheric Administration's (NOAA's) National Centers for Environmental Information (NCEI) is the world's largest provider of weather and climate data. The Integrated Surface Database (ISD) consists of global hourly and synoptic observations compiled from numerous sources into a single common ASCII format and common data model. The ISD includes more than 35,000 stations worldwide, and tracks parameters such as wind speed and direction, wind gust, temperature, dew point, cloud data, sea level pressure, altimeter setting, station pressure, present weather, visibility, and precipitation. For this demonstration, only the outdoor air temperature was considered as a model input.

The ISD is publicly accessible at https://www.ncdc.noaa.gov/isd. For this demonstration, data were accessed via FTP at ftp://ftp.ncdc.noaa.gov/pub/data/noaa.

Other Data Sources

The project team accessed additional non-structured data sources, including various reports about comprehensive energy and water evaluations, security readiness assessments, and other related audits at USAR sites at the building level. These sources provided contextual information about building systems and operations that were used to inform use case selection and help the project team interpret data.

Equipment Calibration and Data Quality Issues

The PNNL project team did not have direct control over any data collection equipment for the data sources described in the previous section. All data sources are externally owned and maintained by the USAR, NOAA, and/or the Army. Whenever possible, the project team alerted the system owners (or parties responsible for sustainment/operation) of calibration, connectivity, and other data quality issues.

Data quality and availability were a challenge for this demonstration. Section 9.1 discusses these issues in greater detail. Prior to the project getting under way, several major instrument data processing issues, discussed below, were known to the project team. Where feasible, unreliable or missing data were detected and corrected using methods described in Section 5.7. It is important to recognize, however, that calibration and connectivity issues posed challenges not only to this demonstration, but to system users in general. One key output of this demonstration is a thorough characterization of data quality and availability.

Instrument Data Processing

ML models do not generally take instrument readings, such as those reported by control points and building energy meters, as direct input. The characteristics of the data must match the requirements of the algorithm. Consequently, some processing of data in preparation for use by a

given algorithm is typically necessary. Data processing may involve imposing logical structure, categorization, aggregation, or correction of bad or missing data.

"Ideal" data (1) represent the actual process being measured, and (2) are sampled at a frequency high enough to capture relevant patterns. Deviations from this ideal can limit the number of applicable models and/or complicate their application. The following are examples of deviations present in "non-ideal" data that can be characterized statistically.

- Noise: Data can appear to be volatile at short time scales yet exhibit smoothness at longer scales. This volatility can be taken as noise in the measurement if the characteristics of this volatility are the same (predictable) throughout the data.
- Gaps: Data gaps can be flagged if the time difference between readings is not as expected. However, a data gap can be interpolated if it is not severe. The project team defined limits below which data interpolation was acceptable. If a gap in a trend at a particular building was large enough, all data for that building during that time period had to be excluded from the analysis.
- Instrument drift: Instrument drift is a type of instrument calibration problem. Ideally, over long periods of time, instrument measurements should not drift from some central tendency.

These problems were identified by inferring some expected characteristics and flagging deviations. There are various statistical approaches to doing this, depending on the nature of the data. In addition, energy managers and building analysts were consulted to define these expectations based on expert judgment.

5.4 DESIGN AND LAYOUT OF SYSTEM COMPONENTS

The system integrates building energy and controls data from multiple external sources, applies ML methods to the data to identify specific issues as defined by the use case selection process (see Section 6.1), and communicates the results to building operators and energy managers via the EBCS platform. The key components of the system in this demonstration are:

- USAR buildings
- physical sensors and meters
- external database servers
- a PNNL internal ML development environment
- the EBCS platform
- building operators and energy managers.

5.5 OPERATIONAL TESTING

Performance Objective Analysis Overview

The performance objectives of the demonstration were:

- fault detection
- building energy use (MMBtu/ft^2/yr).

As discussed in Section 4.0, it was not possible to fully calculate these metrics due to data limitations. The project team initially planned to measure the metrics associated with these

performance objectives before and after the period of implementation of the ML analytics tools, to determine the effect of the technology. The metrics were to be calculated individually for each building, and the results aggregated to get the mean and standard deviations for each metric across the dataset. Because insufficient data were derived from this demonstration, t-tests were not able to be conducted to determine the significance of performance changes across the population of buildings. As such, the determination of project success relies on the metrics described in Table 3 below.

Model Predictive Accuracy

Additional analysis was conducted to validate the accuracy of the ML models. While this analysis did not directly measure the improvement of performance objectives, the accuracy of the models informs the usefulness of their application to the performance objectives and allows for comparison to state-of-the-art models used across the industry.

Three types of model validation were employed: (1) internal validation during model training, (2) external validation on a subset of historical data not used for model development, and (3) external validation on real data in a set of test buildings for which the technology has been deployed. The first two types of validation will require ground-truth data of building faults from a set-aside validation dataset. It was hoped these data would be available from work order logs, on-site confirmation with staff, and the results of the clustering algorithms used for data preparation. However, as discussed in Sections 6.1 and 9.1, this validation was not possible given the data quality issues present.

Internal Validation

The following techniques were employed to determine internal model validity and reduce overfitting:

- k-fold cross-validation
- bootstrap validation
- recursive feature elimination.

These are standard techniques used for ML model evaluation.

External Validation

For external validation using both historical and post-implementation data, the metrics described in **Error! Reference source not found.** below were calculated. These metrics focus on measuring how well the ML models estimate baseline energy consumption, predict the presence of faults, identify the correct fault type, and improve fault detection over the baseline.

The baseline model accuracy was characterized by the adjusted r^2 value of the model, also known as the coefficient of determination. This metric determines how well a model predicts its target compared to just taking the average of the target value and is often used to evaluate regression models. The baseline models were also evaluated using the CV(RMSE) metric (Coefficient of Variation of the Root Mean Square Error) and the NMBE metric (Normalized Mean Bias Error), which both measure the normalized deviation of the model from the target value.

The baseline model accuracy metrics are defined as:

$$\overline{r^2} = 1 - (1 - r^2)\frac{n-1}{n-p-1} \quad (1)$$

where $\overline{r^2}$ = adjusted r^2 value,
r^2 = regression score (r^2) – un-adjusted,
n = sample size, and
p = number of explanatory variables in the model.

$$CV(RMSE) = \frac{1}{\bar{y}}\,RMSE \times 100\% = \frac{1}{\bar{y}}\sqrt{\frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{n}} \times 100\% \quad (2)$$

where $\bar{y}$ = average value of the true series,
y_i = the true value of the i-th sample,
$\hat{y}_i$ = the predicted value of the i-th sample, and
n = sample size.

$$NMBE = \frac{1}{\bar{y}}\,MBE \times 100\% = \frac{1}{\bar{y}}\frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)}{n} \times 100\% \quad (3)$$

where $\bar{y}$ = average value of the true series,
y_i = the true value of the i-th sample,
$\hat{y}_i$ = the predicted value of the i-th sample, and
n = sample size.

The fault detection and model accuracies were to be measured using the ROC (receiver operating characteristic) curve, which compares the true positive (detection) rate of a model with its false positive (false alarm) rate. This metric is very commonly used to evaluate the predictive capability of ML models and is very useful for optimizing a model to meet specific goals (i.e., maximizing the true prediction rate or minimizing the false prediction rate). However, as discussed in Sections 6.1 and 9.1, it was not possible to assess the developed models with these metrics because they require ground-truth fault data, which were not available.

Table 3. External Validation Metrics

Objective	Metric	Success Criteria
Baseline model accuracy	Adjusted r^2 value CV (RMSE) NMBE	$r^2 > 0.7$ CV < 25% -0.5% < NMBE < 0.5%
Fault detection model accuracy	ROC curve (tradeoff of detection and false alarm rates) for identified faults	ROC area under the curve (AUC) > 0.9
Fault detection model performance[(a)]	Number of faults detected; number of data points required to detect fault as compared with previous methodology	Statistically significant improvement in the number of correctly detected faults and a reduction in the number of data points required for detection
Fault-identification model accuracy [(b)]	ROC curve for each fault type	Average ROC AUC > 0.9
(a) Ability of the model to detect that a fault has occurred. (b) Ability of the model to correctly identify the fault type.		

In addition to model accuracy, the project team initially planned to compare the performance of the ML models to the baseline methodology for fault detection (manual data inspection) by examining the number of detected faults and the number of data points required to detect the faults with both methods. As with the performance objectives, the average and standard deviations of these metrics were not able to be obtained due to insufficient data.

5.6 DATA INTEGRATION PLATFORM

The data for the project were collected from multiple sources and compiled in a single database on the PNNL network. As described in Section 5.3, the primary data sources for the project are NOAA temperature data, MDMS utility consumption data, and EBCS control point data.

Each building integrated into EBCS can have thousands of individual points, depending on the control system present at the building, each with its own stored time-series history. These time series can record as often as every 5 seconds for multiple years. Given the volume of data involved, it was necessary to develop a programmatic way to interact with the data, because manually downloading and analyzing data at that scale are difficult. To that end, a data integration platform was created to query the databases programmatically (i.e., with the Python programming language). The basic functionality of the data integration platform is illustrated in Figure 7.

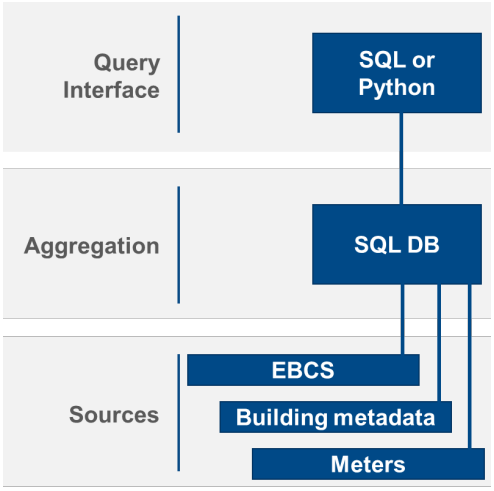

Figure 7. Data Integration Platform Schematic

Not only does this programmatic access allow the database to be queried without having to write raw SQL scripts, but it also allows the process to be highly repeatable and extensible to multiple buildings. The platform not only handles querying the database, but also brings the data into the Python programming environment so that they can be incorporated into ML models.

PNNL Cybersecurity Posture

PNNL has the Authority to Operate (ATO) to meet Federal Information System Management Act (FISMA) requirements and approval to operate information systems as approved by the Department of Energy's (DOE's) Pacific Northwest Site Office (PNSO). PNNL is approved to manage and operate moderate risk level systems—those that collect, process, or store Official Use Only (OUO)/ For Official Use Only (FOUO)/Personal Identifiable Information (PII) data. This includes Impact Level 4 Controlled Unclassified Information (CUI) data, as defined in DoDI 5200.48.

PNNL is one of 10 DOE Office of Science Laboratories, which has been operated by Battelle for DOE since the Lab's inception in 1965. PNNL has had a formal computer security program in place since the 1980s. DOE National Laboratories must maintain an ATO for each major information system they operate. The DOE ATO process is based on Federal Information Processing Standards (FIPS), FISMA, and National Institute of Standards and Technology (NIST) guidance for the security authorization of information systems.

PNNL maintains a baseline security profile for any system managed or owned by PNNL, which is identified in the Site Security Plan and approved by DOE as part of the ATO process. PNNL's minimum level of protection assumes OUO/FOUO/PII data are processed and or stored on the system. PNNL has implemented extensive system-wide tools for staff and system administrators to ensure security is part of their system's life cycle. Example measures include a Public Key Infrastructure joined with DOE that is embedded in desktop products, continuous monitoring of all systems, regular proactive patching, malware signature updates, and mandatory annual operational security and security training for all staff and collaborators.

Any software considered for deployment on the EBCS was subjected to operational testing on a PNNL virtual machine with appropriate Security Technical Implementation Guides applied to

replicate the controls in the EBCS deployment environment. Cybersecurity concerns related to the technology transfer and application deployment are discussed in Section 9.2.

During the demonstration, EBCS data were captured via file transfer by PNNL/Army staff via DoD SAFE (Secure Access File Exchange) and did not rely on a live connection to the ARNet. In accordance with the PNNL contract with USAR PNNL staff are authorized to have USAR computers and contractor access to the EBCS and MDMS. All performers are aware of the cybersecurity requirements and constraints on access to the Army servers. Future live connections between systems will be coordinated and implemented by USAR.

5.7 DATA QUALITY ANALYSIS AND CLEANING

As stated throughout this report, the quality of the input data has a direct impact on the output accuracy of a ML model. For this reason, an important part of every ML project workflow is an analysis of the input data quality and cleaning of the data prior to their being fed into a model for training.

Before any analysis can be performed, data formats must be standardized. Different systems are responsible for recording the data from each source. These systems have different recording frequencies and temporal precisions. NOAA temperature data are stored at an hourly frequency with a minute precision (e.g., each value is recorded with a time stamp like “2019-03-18 12:30). The data from the MDMS and EBCS are recorded as often as every 5 seconds, with timestamps captured to a millisecond precision (e.g., each value is recorded with a time stamp like “2019-03-18 12:30:00.047). In order to be combined, these time series have to be aligned to the same frequency and precision. Each time series is rounded to the nearest 15-minute increment. For some signals this means they are being up-sampled (e.g., the NOAA hourly temperature data are up-sampled to 15-minute increments using linear interpolation). For other signals it means they are being down-sampled (e.g., a discharge temperature point that records every 5 seconds is down-sampled so that only one sample in each 15-minute increment is kept).

After standardizing the timestamps associated with the data, the NOAA temperature data and the EBCS control point data are ready to move to the next data processing step. However, additional data preparation is required for the MDMS meter data. The advanced meters in the Army’s Metering Program record the running total consumption of the meter at each 15-minute interval. As such, the interval usage must be calculated as the difference between adjacent 15-minute intervals.

After these initial data-cleaning steps, data quality analysis (DQA) is an important next step in examining what is and what is not possible given the data available. The analysis primarily focuses on two areas: (1) identifying where data are or are not sufficient for ML and (2) identifying outliers/anomalous data. If too many data are missing from a time series in a given date range (i.e., a data gap is too large), it can preclude certain ML use cases that rely on large volumes of data to ensure a minimum level of model accuracy. Moreover, outliers in the training data can degrade the accuracy of a ML model, so they were removed prior to training.

Data Quantity

The most important factor in determining whether data are or are not sufficient for ML is how many data, if any, are missing from a given time series. Hence the first step of the DQA is to

analyze the time series for missing data. Figure 8 shows the results of this analysis, which generates some statistics and visual representations of the data gaps. For the use cases investigated in this report, small gaps are acceptable, but large periods of missing data are not.

```
Total Time: 508 days 11:00:00
Percent Outliers: 0.77%
Percent Missing Time Steps: 5.40%
Largest Gap: 9 days 01:00:00
Largest Gap Percentage of Total: 1.78%
Gaps Description:
count                            8
mean         3 days 10:24:22.500000
std       3 days 19:09:08.419980523
min                 0 days 01:15:00
25%                 0 days 19:45:00
50%                 1 days 13:00:00
75%                 6 days 01:00:00
max                 9 days 01:00:00
Name: dt, dtype: object
Gaps Longer than a Day: 2
Longest Continuous Period: 116 days 22:00:00
Longest Continuous Percent of Total: 22.99%
Continuous Periods Description:
                          timestamps
count                              9
mean               53 days 09:26:40
std     47 days 07:31:14.151683244
min                 0 days 21:00:00
25%                12 days 22:00:00
50%                32 days 22:00:00
75%               108 days 09:00:00
max               116 days 22:00:00
_step_size: 0 days 01:00:00
```

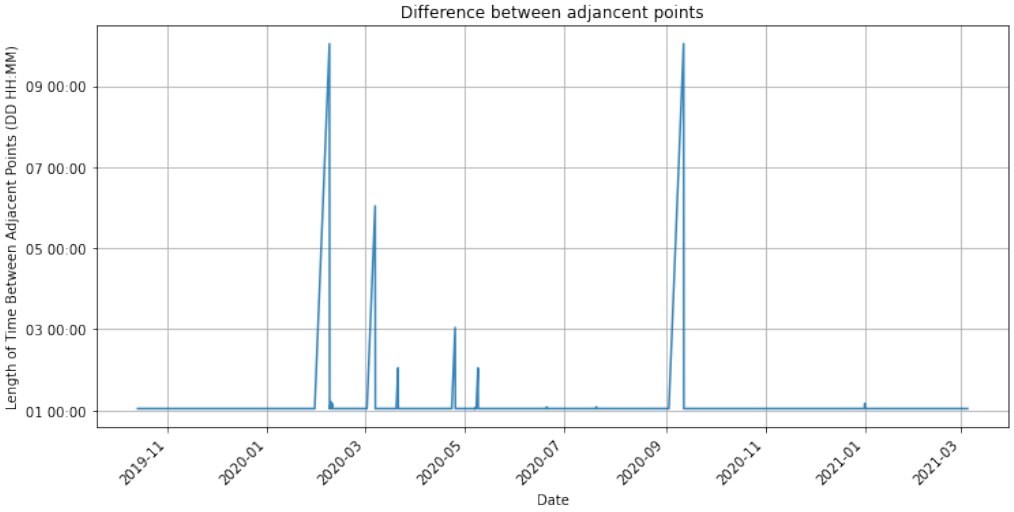

Figure 8. Sample Output of Gap Statistics from MDMS

After the amount of data present has been verified, the DQA generates various plots that show relationships between different features (temperature and temporal measurements) and the prediction target (whole-building energy consumption) as well the expected behavior of the prediction target. This portion of the DQA primarily informs feature selection (discussed further in Section 6.2) and provides context for the range of expected model predictions.

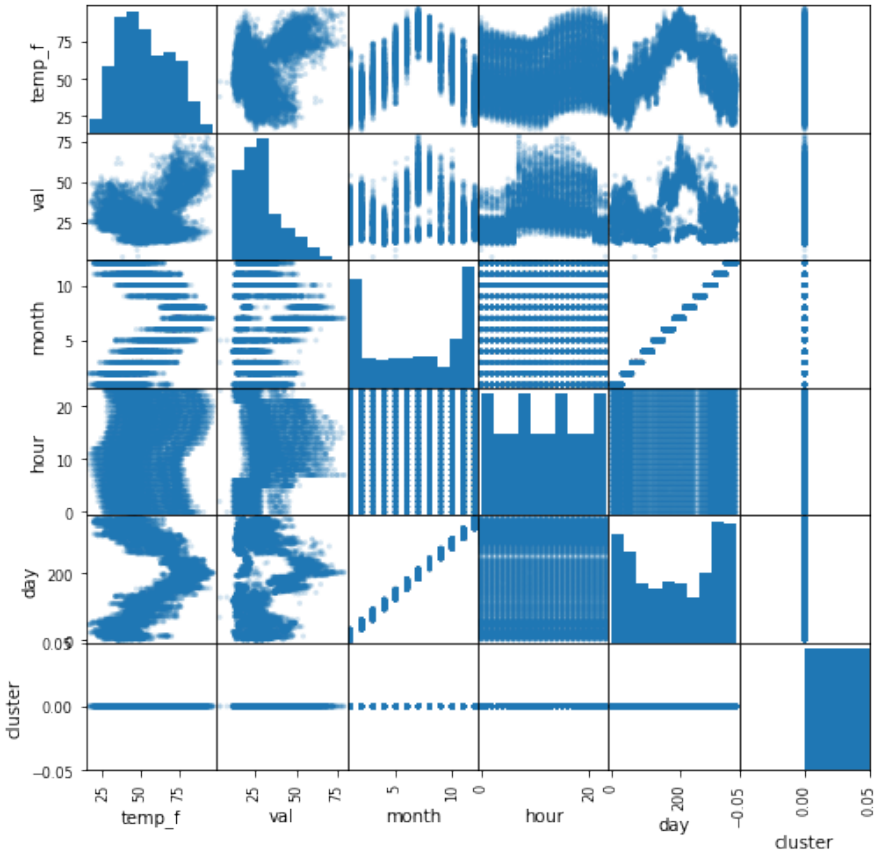

Figure 9. An Example of a DQA Plot Showing the Relationship between Features and the Prediction Target

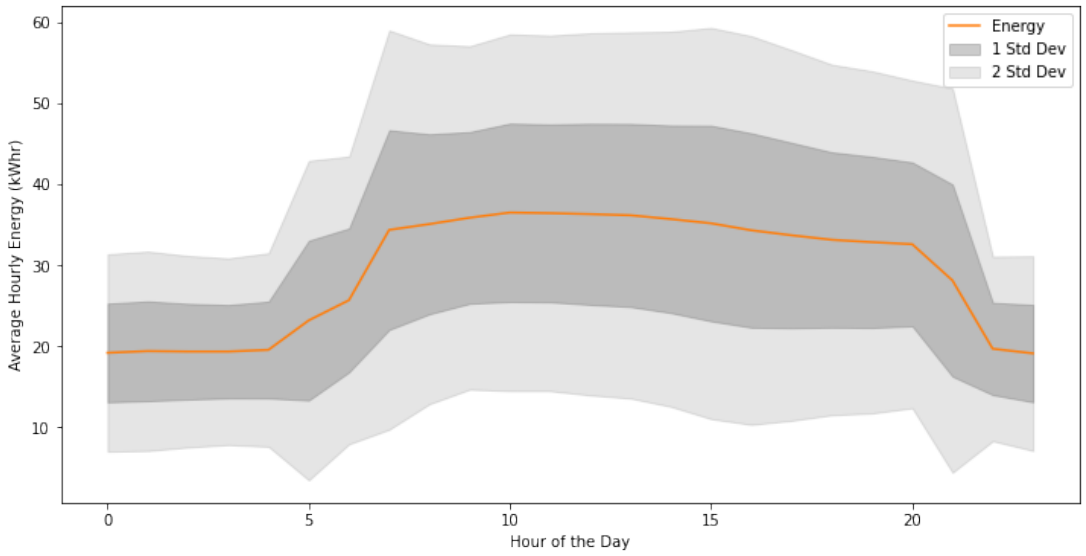

Figure 10. An Example of a DQA Plot Showing the Daily Average Hourly Energy and Two Standard Deviations

Outlier (Anomaly) Detection

The second portion of the DQA addresses the identification of outliers (anomalous data) and their subsequent removal. Outliers can result from faulty measurements, calculation errors, or simply valid data that for some reason is well outside the normal expected range of values. No matter the reason for the outlier, they can cause ML prediction accuracy to degrade if they are present in the training data, so they are typically removed from the dataset prior to training. The outlier detection procedure of the DQA was incorporated into the data ingestion pipeline for each model, similar to the data-cleaning stage. Hence, before any model in the project was trained, it first went through the outlier detection step.

The outlier detection method used for this project is itself an ML algorithm. Instead of relying on hard-coded thresholds (e.g., >95th percentile) as a means of excluding outliers, a clustering approach was used. Clustering is a type of unsupervised learning that groups samples into clusters that are similar based on some defined distance metric. The clustering algorithm used for this project was a Density Based Spatial Clustering of Applications with Noise, or DBSCAN (Ester et al. 1996) as implemented in scikit-learn.[4] The DBSCAN algorithm has a few attributes that make it well suited for outlier identification. First, the total number of clusters does not need to be known beforehand in contrast to other clustering algorithms (e.g., K-means clustering). Second, because the clusters are defined based on the density of surrounding points, the clusters can be an arbitrary shape. Additionally, the DBSCAN algorithm can efficiently handle a large number of samples. All of this together makes DBSCAN well suited to be incorporated as the outlier detection method. Figure 11 provides an example showing clear outliers (identified with red dots) in both temperature and energy time series that were identified and removed.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN

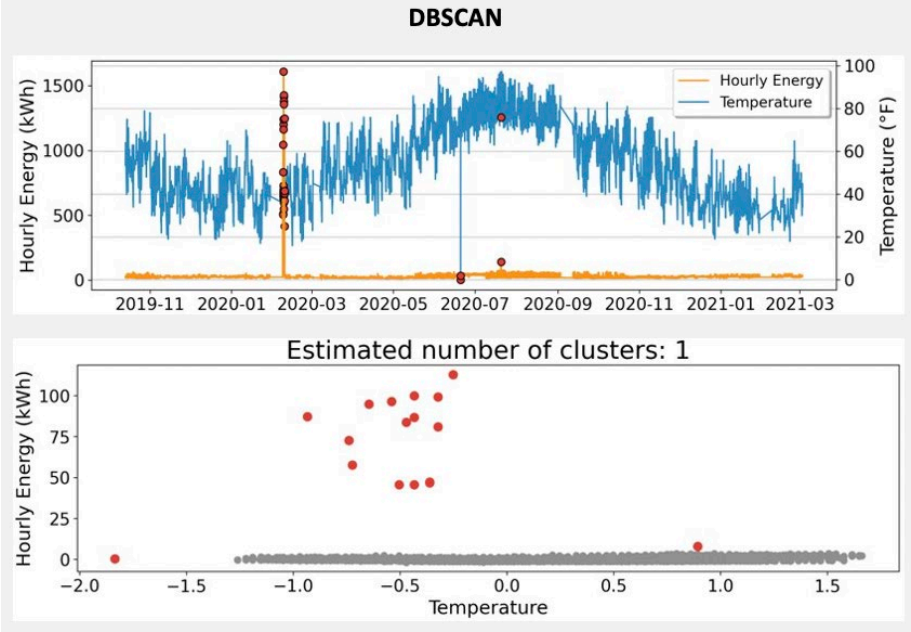

Figure 11. Outlier Identification Illustrated with Hourly Energy and Temperature Plotted over Time and a Scatter Plot of the Same Hourly Energy Data as a Function of Temperature

6.0 PERFORMANCE ASSESSMENT

6.1 USE CASE SELECTION

The range of possible ML use cases is dependent on what data are available to train the algorithms. Each ML use case has inherent data requirements that, if not met, preclude accurate and actionable results and may preclude the use case entirely. During this demonstration, data quality and availability issues presented a challenge to investigating even simple use cases. As illustrated in Figure 12, only nine USAR buildings had adequate data for investigating ML use cases beyond DQA and baseline prediction (using meter and weather data inputs only).

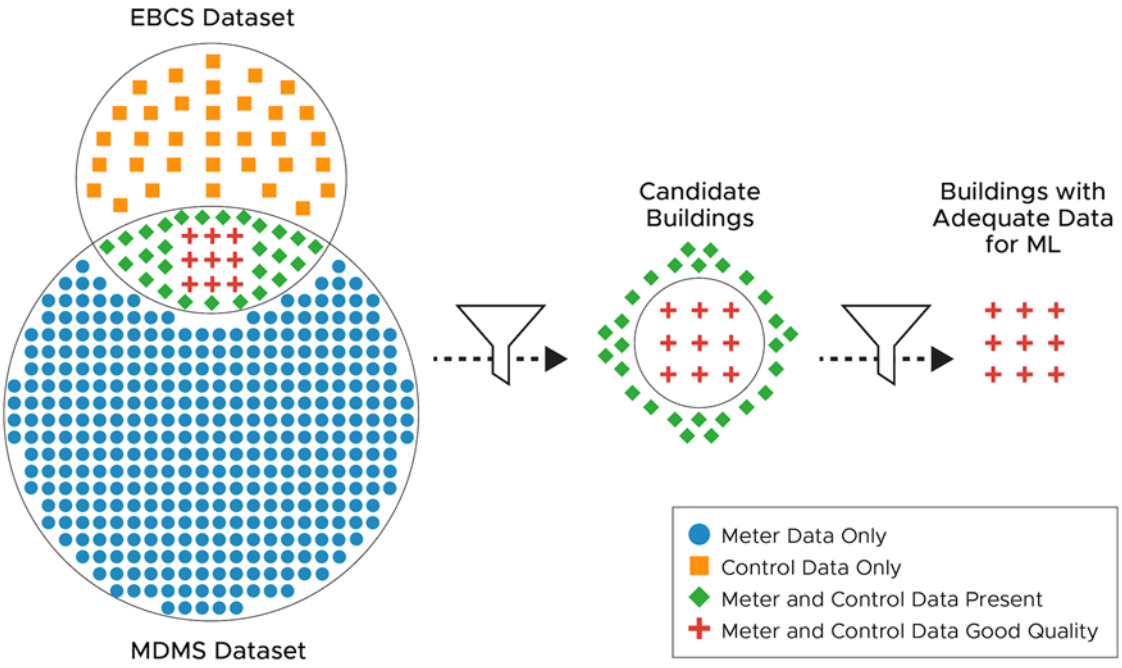

Figure 12. Down-selection to Buildings with Adequate Data for ML

The Use Case Prioritization Matrix, presented in Appendix C, provides a tool for evaluating which use cases are possible given the available data in the context of USAR. Table 4 presents the results of that evaluation for use cases that were initially proposed as part of this project. The background color of the Actual Data Available column indicates whether the requirements are met by the available data; green indicates that the data requirements are sufficiently met by the available data, yellow indicates they are partially met, and red indicates the requirements are not met.

Table 4. ML Use Case Prioritization Matrix Applied to USAR Buildings Data

Use Case Description	Use Case Category	Data Attribute	Data Required	Actual Data Available
Labeling consumption data as anomalous or non-anomalous	Pre-processing anomaly detection	Measurements	Utility use data (hourly or smaller resolution): • Gas consumption • Electricity consumption • Water consumption Outdoor environmental data (hourly or smaller resolution).	Data requirements met.
		Data Volume	No strict minimum requirement.	Data requirements met.
		Data Quality	Gaps are tolerable.	Data requirements met.
Baseline consumption modeling	Fault detection, energy benchmarking	Measurements	Utility use data (hourly or smaller resolution): • Gas consumption • Electricity consumption • Water consumption Outdoor environmental data (hourly or smaller resolution).	Outdoor environmental data available at hourly resolution from NOAA across all of CONUS. Where utility meter data is available, it is available at hourly or smaller intervals.
				Utility meter data are only available for a portion of the buildings that have control data. Many buildings with utility meters do not have control data available. Where utility meters are present at buildings, electricity is the most common meter type. Gas and Water utility meter data are not consistently available.
		Data Volume	At least 1 year. Multiple years is preferred.	At least 1 year of overlapping meter and controls data available for just 9 of ~70 buildings. Multiple years not available.
		Data Quality	Must be fault-free. Sparse gaps are tolerable (a few data points missing).	Fault status unknown, not possible to say whether data are fault-free. Depends on building, but large gaps (a few hours up to a few months) are present.

Use Case Description	Use Case Category	Data Attribute	Data Required	Actual Data Available
Unsupervised fault/anomaly detection	Fault detection	Measurements	Step 1 (baseline consumption model training): fault-free data as required to train baseline consumption model.	Step 1: same data availability constraints as described in the baseline consumption modeling use case, fault status unknown.
			Step 2 (fault detection): consumption measurements for time period for which fault detection is to be performed (can be a real-time stream).	Step 2: N/A, cannot progress past Step 1 without fault-free data.
		Data Volume	Step 1 (baseline consumption model training): At least 1 year. Multiple years is preferred.	At least 1 year of overlapping meter and controls data available for just 9 of ~70 buildings. Multiple years not available.
		Data Quality	Step 1 (baseline consumption model training): must be fault-free.	Step 1: fault status unknown, not possible to say whether data are fault-free.
			Step 2 (fault detection): can have faults. Both: Sparse gaps are tolerable (a few data points missing).	Step 2: N/A, cannot progress past Step 1 without fault-free data.

The only use case that has actual data that meet the data requirements is the preprocessing step of identifying anomalous data. The DBSCAN clustering algorithm described in Section 5.7 was incorporated into the data pipeline as a data-cleaning step before training any of the additional ML algorithms. This use case was successfully demonstrated because the data requirements for the use case were met.

The baseline consumption modeling use case adequately meets the data requirements for both the available measurements and the volume of data for a small subset of all the EBCS buildings, resulting in the "partially met" designation for those requirements. However, the data *quality* requirement is not met by the available data. Specifically, the fault status is unknown—i.e., we do not know when and if there are faults present in the training data. While this precludes generating baseline models that are capable of positively identifying faults, it does not eliminate the possibility of creating baseline models entirely. Baseline models can be generated for the buildings with sufficient data, but we cannot say whether they reflect the ideal fault-free operation of the building. We only know the models can predict the current operation, faults included.

For example, if a building has a fault of a stuck open terminal damper, which has caused an increase in the total building energy consumption, that fault would be present in the training data for the baseline model, and the model's baseline power prediction would include the higher energy consumption caused by the fault. Any faults in the available training data will be

incorporated into the models' whole-building consumption predictions. Because we do not have any information about the faults in the training data (or even know if they exist), we cannot teach the model anything about those faults or use the model to identify existing faults.

As such, the baseline models are not able to identify *existing* faults in the buildings. This limits the models' use in fault detection. Therefore, as shown in Table 4, the minimum data requirements are not met for this use case. However, given additional years of data (compared to the minimum length of time needed for training) the models would be able to detect *new* faults in the building that cause an increase in whole-building energy consumption.

6.2 BASELINE PREDICTION

Despite the data challenges described above, there was sufficient data for nine buildings to train ML models to predict the buildings' baseline whole-building electrical consumption. Two fundamental steps in building those models are (1) selecting which type of model to use and (2) determining which input features (i.e., the input variables fed into the model) to select. Both steps are described below.

ML Model Comparison

Baseline energy consumption prediction is a regression task and many types of ML models are capable of regression. Although some types of models generally outperform others, it is very difficult to determine beforehand exactly which model type will perform the best for any given task on a specific dataset. Therefore, it is necessary to experiment with different model types and architectures to see which one ultimately should be used for the task at hand. To that end, we tested eight different types of ML models to see which would perform the best at predicting whole-building electricity consumption for each of the buildings that had sufficient data. Figure 13 shows an example of several models' predictions of baseline consumption compared to actual consumption.

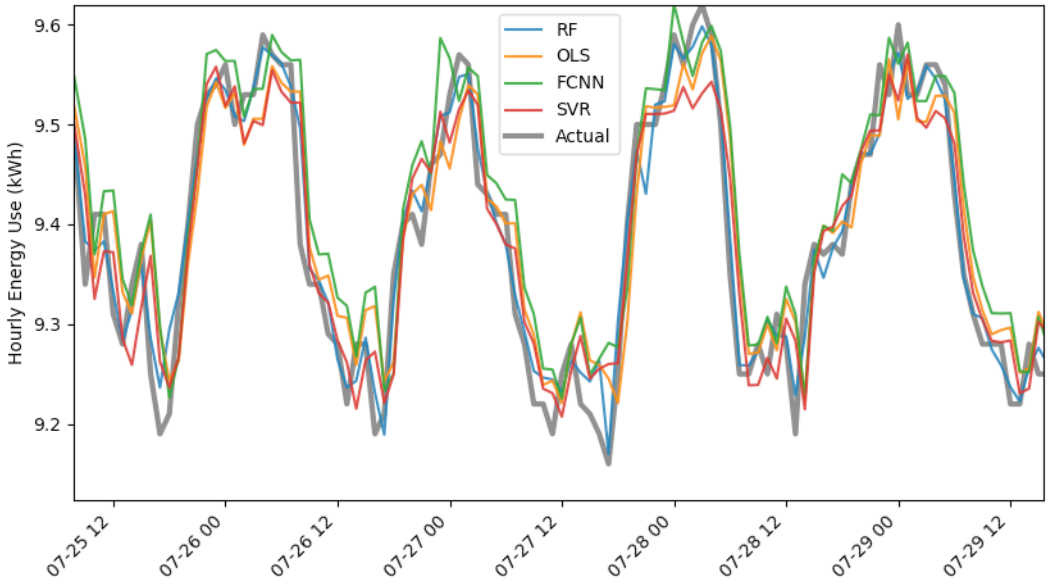

Figure 13. Several ML Model Predictions of Baseline Consumption

Each model was implemented in the data integration platform (described in Section 5.6). We tracked all the ML model experimentation (including this preliminary model type selection experiment) with MLflow, a Python library designed to record and manage ML experiments.[5] These tools allowed us to explore which models performed best without the need to manually record or save the input features, model hyperparameters, or output results. Table 5 shows the types of models implemented, which family those models belong to, and which Python implementation was used for this project.

Table 5. ML Model Types Implemented and Tested

Model Type	Model Family	Python Implementation Used
Ordinary least squares (OLS)	Linear regression	scikit-learn [(a)]
Support Vector Regressor (SRV)	Support vector machines	scikit-learn[(a)]
Random Forest	Classification and Regression Trees (CART)	scikit-learn[(a)]
Gradient Boosted Regression Tree (GBRT)	Classification and Regression Trees (CART)	scikit-learn[(a)]
Adaptive Boosting (AdaBoost)	Classification and Regression Trees (CART)	scikit-learn[(a)]
eXtreme Gradient Boosting (XGB)	Classification and Regression Trees (CART)	XGBoost [(b)]
Multi-layer Perceptron (MLP)	Neural network	scikit-learn[(a)]
Fully connected neural network (FCNN)	Neural network	Keras [(c)]/ TensorFlow
(a) https://scikit-learn.org/stable/ (b) https://scikit-learn.org/stable/ (c) https://xgboost.readthedocs.io/en/stable/		

Relative to many state-of-the-art ML problems, the input data for this project are relatively simple. The training datasets used for any given building never exceed a few hundred megabytes (MB). For comparison, one of the leading language models, OpenAI's Generative Pre-Trained Tranformer-3 (GPT-3) was trained on 45 terabytes (TB)—roughly 1,000,000 times more data than in our case. The relatively small size of our dataset had a notable impact on which model performed best. More complex models, like the fully connected neural network (FCNN), were outperformed by simpler models.

The Classification and Regression Trees (CART) family of algorithms is well suited for dealing with our relatively simple data. Random forest, Gradient Boosted Regression Tree, AdaBoost, and XGB (eXtreme Gradient Boosting) are four of the most commonly used and performant algorithms in this family and all showed excellent performance on this project. The scatter plot in Figure 14 illustrates predictions made by the random forest model compared to actual observations; the orange line represents perfect prediction.

[5] https://mlflow.org/

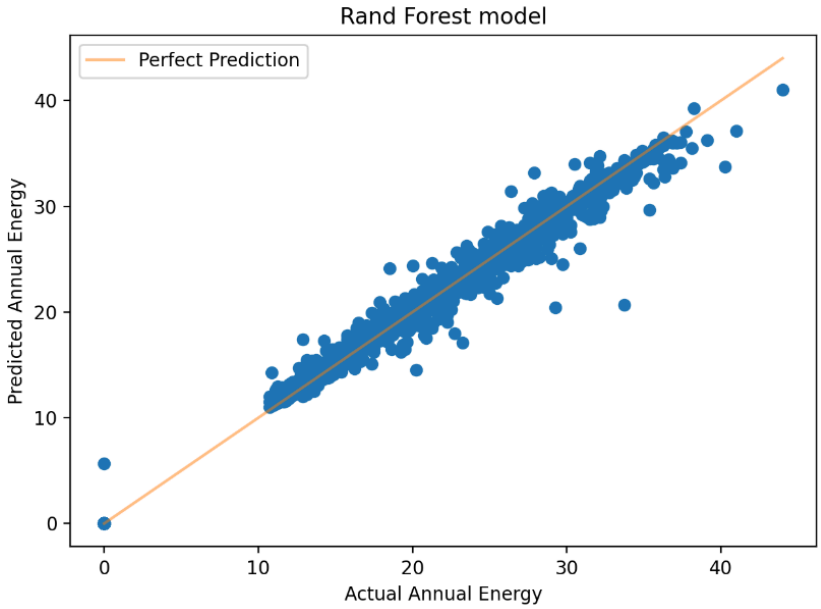

Figure 14. Actual Energy Usage Compared to Predictions by the Random Forest Model

The models were all implemented into a single Python class that served as the ML pipeline for the project. The class standardized the inputs and outputs of model training, making it simple to switch between models during the experimentation phase, and assured that the models received the same inputs for evaluation. The scikit-learn models (see Table 5) and the XGB model all have more rigid architectures; although there are some hyperparameters to tune, they do not significantly affect the model architecture. In contrast, the FCNN (which is implemented with Keras/TensorFlow) is a custom model with manually defined layers. The model is implemented as a sequential model (a linear stack of neural network layers) that consists of three layers. The first two layers are dense layers (each neuron is connected to every neuron in the previous layer) with the same shape as the input features, using rectified linear units (ReLUs) and sigmoid activation, respectively. These layers are referred to as hidden layers. The last layer (output layer) is also a dense layer but only has a single output, using a linear activation. The model is compiled with an ADAM optimizer and uses mean squared error as the loss parameter. Figure 15 illustrates the FCNN architecture employed for this project.

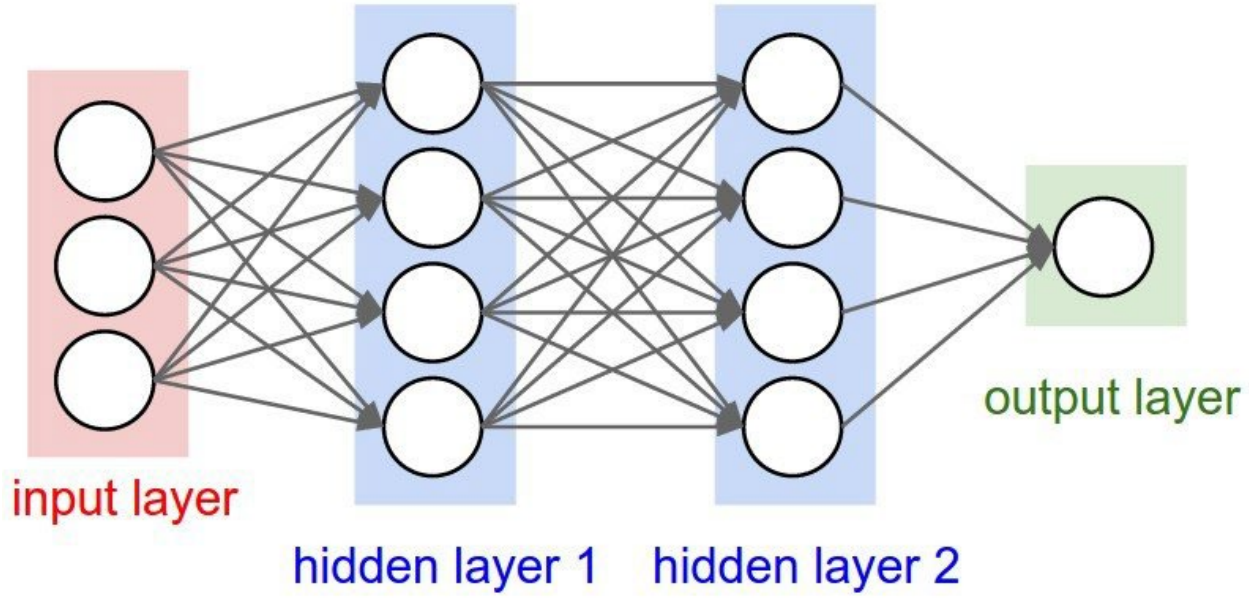

Figure 15. Sample FCNN Architecture from Kim et al. (2022)

Each model type performs slightly differently for each building on which it was trained. A varying amount of data is available at each site, which affects the overall performance. Each model type was trained and tested at each building using the same input features. Figure 16 shows the full set regression score (r^2) for each building, along with the model average r^2 value for all the buildings. These results are also compared to the "naive" prediction, where the prediction is simply the value from the previous timestep. The naive prediction does quite well for a single timestep forecasting window; this is a common benchmark used to gauge the relative performance of the other models.

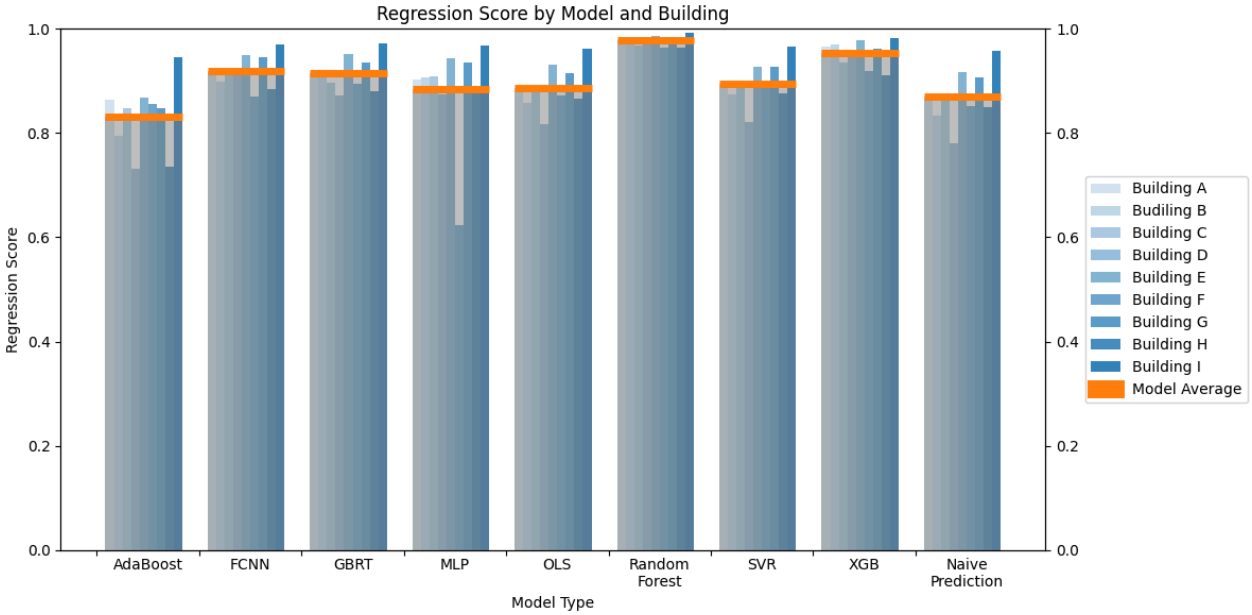

Figure 16. Comparison of Model Performance (full set regression score) for All Buildings that Had Sufficient Data

Table 6 presents the average r^2 values show in Figure 16 along with the model accuracy metrics described in Section 5.5, averaged across all buildings for each model. These results show that all the models met the success criteria for baseline prediction described in Table 3.

Table 6. Average Baseline Model Performance Accuracy Metrics for All Buildings by Model Type

Model	Average Full Set r^2	Average Adjusted r^2	Average CV(RMSE)	Average NMBE
OLS	0.887	0.887	11.1%	0.0%
SVR	0.894	0.894	10.7%	0.6%
Random Forest	0.978	0.977	4.8%	0.0%
GBRT	0.914	0.914	9.6%	0.0%
AdaBoost	0.832	0.839	13.5%	-3.6%
XGB	0.953	0.953	6.9%	0.1%
MLP	0.883	0.887	9.7%	-0.1%
FCNN	0.919	0.921	8.9%	-0.2%
Naive prediction	0.871	0.887	11.1%	0.0%

Note that there are many different types of commercial buildings with a variety of operational profiles. The buildings analyzed in this dataset have well-defined schedules and occupancy so there is fairly minimal hour-to-hour variability in energy consumption except during startup and setback. Figure 17 illustrates an average hourly electricity use profile for a representative Army Reserve training center in the month of June.

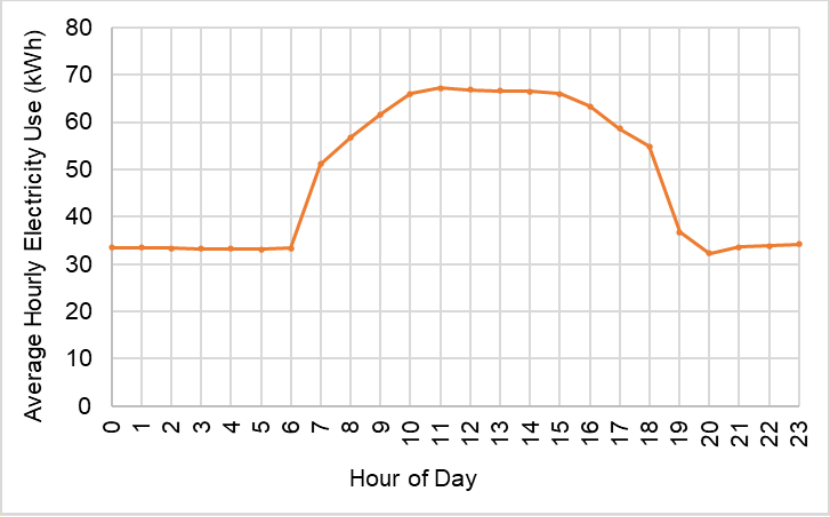

Figure 17. Representative hourly electricity use profile for an Army Reserve training Center in the month of June.

Naïve prediction uses the model Q_t = Q_t-1; as such, accuracy is a function of temporal variability. When the naïve model performs well it means that there is relatively little change between timesteps. This would be expected for a building with well-defined profiles, with large changes limited to a few hours a day during startup and setback. The research team recognizes

the simple predictive models could be less accurate with more diverse commercial building energy consumption profiles.

Input Feature Selection

Input features are what an ML model uses to predict a corresponding output (in our case, whole-building electrical consumption). Input feature selection is just as important as model type selection for a given task. The process of selecting which input features to use and which, if any, transformations to apply to the features is generally referred to as feature engineering. Poorly selected input features can significantly reduce the performance of an otherwise well-performing model, so robust feature engineering is a critical step in the model development process.

As with model selection, it is difficult to determine exactly which combination of features will produce the best performing results beforehand, so feature engineering typically involves varying the input features and evaluating how the changes affect the model's accuracy.

All the features are scaled with scikit-learn's Robust Scaler[6] prior to model training. Each feature is scaled independently. The Scaler subtracts the median and scales the values to the interquartile range (e.g., the 25th and 75th quantile). The input features can have very different scales (e.g., the month range is in [1-12] while recorded electricity consumption could be over 10,000 kWh) which some models are not well suited to handling without hurting performance. Some models also expect the features to be centered around zero or are built with assumptions about their variance. The Robust Scaler accommodates these requirements.

Additional transformations are applied to the time-based features before the centering and scaling of the Robust Scaler. The weekday/weekend feature is transformed using a method known as one-hot encoding. The day-of-the-week value is an integer that is in [1-7]; one-hot encoding transforms these values into a binary feature with 0 corresponding to a weekend and 1 corresponding to a weekday. The other time-based features (month, day-of-the-year, hour, minute) are transformed into two component columns through cyclical encoding. These features are inherently cyclical, always restarting at the beginning of the sequence after reaching the end. Cyclical encoding transforms the raw values into sine and cosine components representing the progress through the cycle. This allows the model to recognize that the 23rd hour (11:00 pm-11:59 pm) is next to the 0^{th} hour (12:00 am-12:59 am). Without this encoding, the model interprets the two endpoints as far apart in the feature space. Figure 18 illustrates how the encoding transforms the raw hour feature into the two cyclical component features. The month, day-of-the-year, and minute features are transformed in the same manner. In the figure, the panels show the raw hour value (left), cyclically encoded hour values (center), and visualization of cyclical relationship of encoded hour feature (right).

[6] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

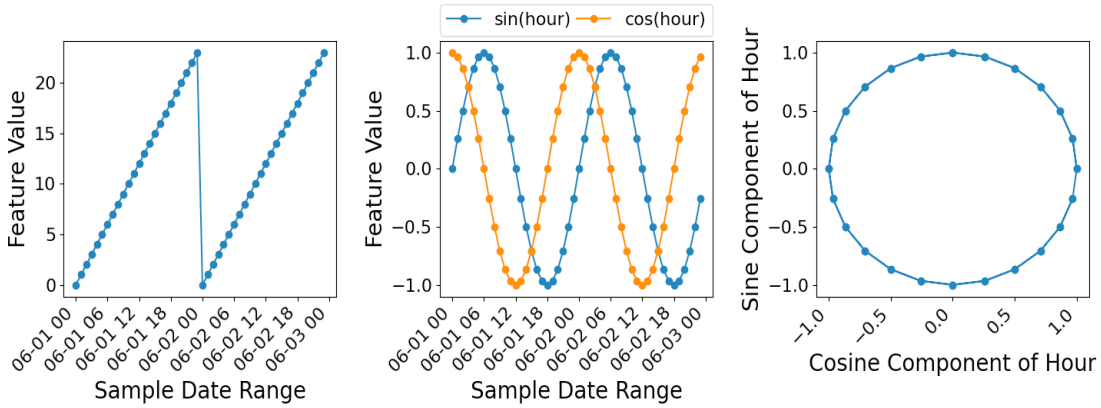

Figure 18. Effect of Cyclically Encoded Time Features

The project started with a baseline set of input features: electricity consumption during the previous time stamp (Q_t-1), outdoor air temperature (OA temp), the hour of the day, the month, and if the day was a weekday or weekend (wk_day). These were the features used in the model selection described above. With these simple inputs, all of the baseline models were able to achieve accuracy that exceeded the success criteria defined at the beginning of the project. Figure 19 shows the relative importance of each of these features on the random forest model. Not surprisingly, the consumption value of the previous stamp is by far the most important feature. It is in effect incorporating the naive prediction (itself a good predictor, as discussed in the ML Model Comparison section) into the model.

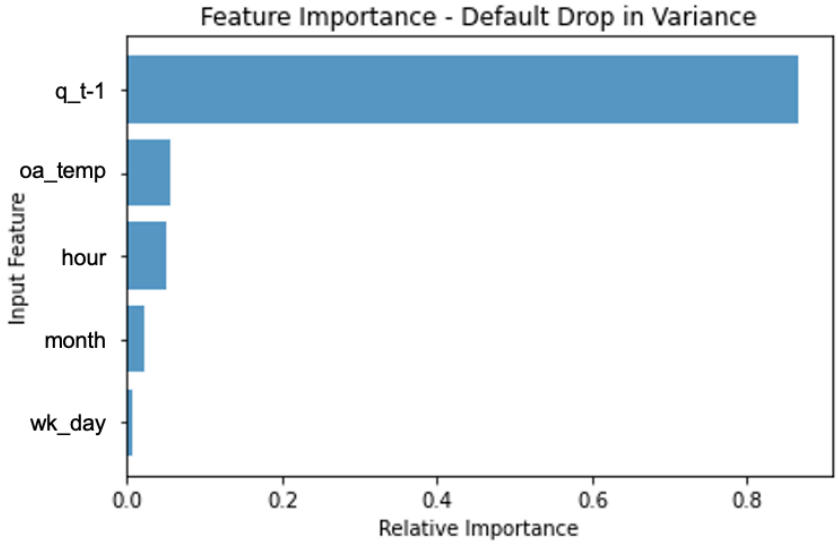

Figure 19. Relative Importance of the Baseline Features for the Random Forest Model

While the baseline features described above produced accurate models, we were interested in investigating different combinations of features to achieve as many of the following outcomes as possible:

1. Improve model performance.
2. Obtain performant models without using NOAA temperature data as an input feature.

(Obtaining outside air temperature data from NOAA requires an open network connection to the internet, and the team was unsure if the deployment environment on the USAR's network would allow that connection.)

3. Obtain performant models without using the previous timestep consumption value as an input feature.

Including the previous timestep value as an input to the model limits the model's utility in predicting faults. Faults are unlikely to affect the energy consumption for only a single timestep. By including the previous timestep as an input, any change in energy consumption caused by a fault would be incorporated into the next prediction, essentially calibrating the model to predict the change in consumption caused by the fault.

To investigate the above objectives, the team ran each model type on each building for each different feature set included in the investigation. In addition to removing the OA Temp and previous timestep features, the team investigated whether model performance could be increased by including data from temperature sensors within the building. To that end, any point with the sub-string "temp" in the point name (referred to as the Building Temps features) were included as features. That led to training eight separate instances of each model type, each with a different input feature set. The model accuracy of each feature set and model type is shown in Table 7 and compared to the naive prediction results. Each feature set iteration varies the input features, but all iterations keep the time-based features (month, hour, and weekday). All the model types were investigated to see if the different combinations of features would lead to a different type of model, beating out the random forest model. While all the CART models performed well, the Random Forest regressor still performed best.

Table 7. Average Baseline r^2 score for All Buildings by Model Type and Input Feature

	Model Type							
Features Used	**OLS**	**SVR**	**Random Forest**	**GBRT**	**AdaBoost**	**XGB**	**MLP**	**FCNN**
Model 0, Naive Prediction	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
Model 1, Q_t-1, OA Temp	0.89	0.89	0.98	0.91	0.83	0.95	0.88	0.92
Model 2, Q_t-1, NO OA Temp	0.89	0.89	0.97	0.91	0.84	0.95	0.88	0.91
Model 3, NO Q_t-1, OA Temp	0.36	0.56	0.85	0.58	0.43	0.72	0.58	0.62
Model 4, NO Q_t-1, NO OA Temp	0.36	0.50	0.58	0.53	0.47	0.58	0.52	0.56
Model 5, Building Temps, Q_t-1, OA Temp	0.90	0.73	0.98	0.93	0.87	0.97	0.28	0.71
Model 6, Building Temps, Q_t-1, NO OA Temp	0.90	0.73	0.98	0.93	0.87	0.97	0.08	0.78
Model 7, Building Temps, NO Q_t-1, OA Temp	0.72	0.61	0.97	0.83	0.75	0.96	-0.04	0.73
Model 8, Building Temps, NO Q_t-1, NO OA Temp	0.72	0.61	0.97	0.83	0.74	0.96	0.11	0.54

Figure 20 breaks the results further into the test and training set accuracy metrics for just the random forest model. The difference in accuracy between these two datasets shows how well the model will perform on unseen data. A model that is overfitted on the training data will have a high predictive accuracy for the training set, but poor accuracy for the test set. As the figure shows, the baseline features (Model 1) produce a model that is highly accurate and performs well on unseen data. Removing the OA Temp feature (Model 2) has little impact on the test or training accuracy. Consistent with the results shown in Figure 19, the Q_t-1 feature is has the biggest effect on the model prediction. If that feature is removed from the baseline set of features, the accuracy decreases significantly (Model 3 and Model 4). Removing the Q_t-1 feature but keeping the OA Temp feature results in a training set that still beats the naive prediction (Model 3). However, upon reviewing the test set, the model is clearly overfitted and does not perform well on new data, as shown by the large difference between train and test set accuracy. Removing both the Q_t-1 and OA Temp features results in a model that performs poorly on both the test and training datasets.

On the other hand, adding in the indoor building temperature point features makes the model much more resilient to the removal of the OA Temp and Q_t-1 features (Models 5-8). Impressively, when the building temperatures are included but the OA Temp and Q_t-1 features are not (Model 8), the overall regression score is comparable to the score generated by the model using the baseline input feature configuration. It does appear that there may be some overfitting, but the model still performs well on unseen data. This is encouraging, because it suggests that even in the absence of NOAA outside air temperature or knowledge of the previous timestep, the model can achieve a high level of predictive accuracy.

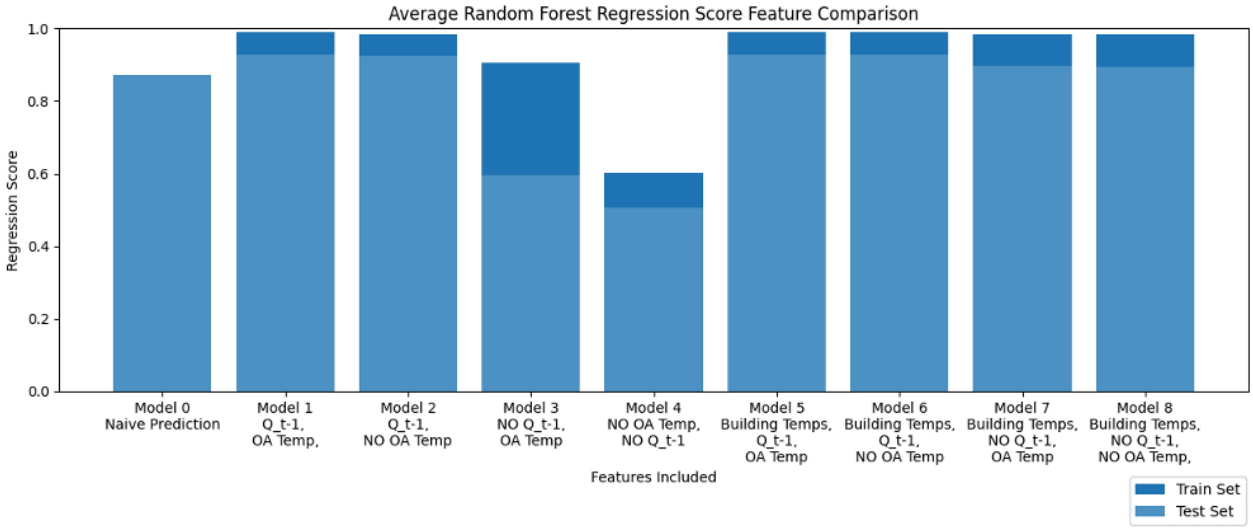

Figure 20. Average Test and Training Set r^2 Score for the Random Forest Model for Different Input Features

The increased robustness of the model comes at the expense of additional training time, however. When using the baseline features (OA Temp and Q_t-1) the average training time for the random forest model was 17 seconds. When the baseline features were removed and the building temperature points were included, the average training time increased to 130 seconds, primarily

due to the increase in data volume. Although each building had a different number of temperature points available, in each case including those points dramatically increased the amount of data in the input feature set. Improved performance and training time requirements are common trade-offs in ML model development.

Ultimately, the random forest model selected for use in the demonstration deployment included the Building Temps features but excluded NOAA outdoor air temperature and the previous timestep value. This model only incorporates input features from the EBCS database and does not require a live connection to the external internet, thereby simplifying the deployment from a cybersecurity standpoint. Not requiring knowledge of the previous timestep value also increases the robustness of the model for potential fault detection; as noted above, if the previous timestep (Q_t-1) is included as an input feature, any change in energy consumption caused by a fault would be incorporated into the prediction of the following timestep (Q_t), thereby essentially calibrating the model to predict the change in consumption caused by the fault.

6.3 FAULT DETECTION

As discussed in Section 6.1, the team was not able to fully implement the fault detection use case because the data requirements were not completely met; without any knowledge of the fault status of the training data, we were not able to train a model to positively identify faults. However, the baseline models can be used to identify periods of energy consumption that *might* include faults. A baseline model trained on data that may or may not have faults present could still be able to identify *new* (i.e., not present in the training data) faults that cause a change in whole-building energy consumption. However, without the known fault data for validation, we have no way to definitively say whether a deviation in actual consumption from the predicted consumption was due to a fault in the building or some other factor.

Fault Detection Data Requirements

The data requirements for the fault detection use case (described in Section 6.1) include high-resolution fault data that would identify all faults in the building. These data would need to be at the same resolution as the input features and output consumption value, meaning they would need to have hourly resolution or finer. Currently, no USAR data source meets those requirements for this project.

The project team originally hypothesized that ground-truth maintenance ticket data, available from the USAR CSS, could be used to identify faults. While these data are not recorded at an hourly level, they do identify a date for each maintenance ticket. These data were insufficient for fault detection, however, due to imprecise or vague fault descriptions and unreliable timestamps (see the expanded discussion in Section 9.1).

In the absence of ground-truth fault data, traditional rule-based fault detection algorithms could be used as a proxy. The EBCS platform does include some fault detection algorithms that would be useful for validation; however, the histories of those analytics are not recorded.

As a last option, the EBCS does record the time-series data for threshold-based alarm points in each building's BAS. These alarm points are generally rule-based and generate an alarm when a certain threshold is exceeded. There can also be a temporal component to the logic (e.g., the threshold must be exceeded for at least 5 minutes). Although the alarm data cannot definitively

substitute for ground-truth fault validation, they can be used as a weak proxy to provide some insight into the performance of the baseline model as a fault detector.

Figure 21 illustrates the relative reliability of each of the data sources described above. Ultimately, the project team only had access to the least reliable data source for fault validation.

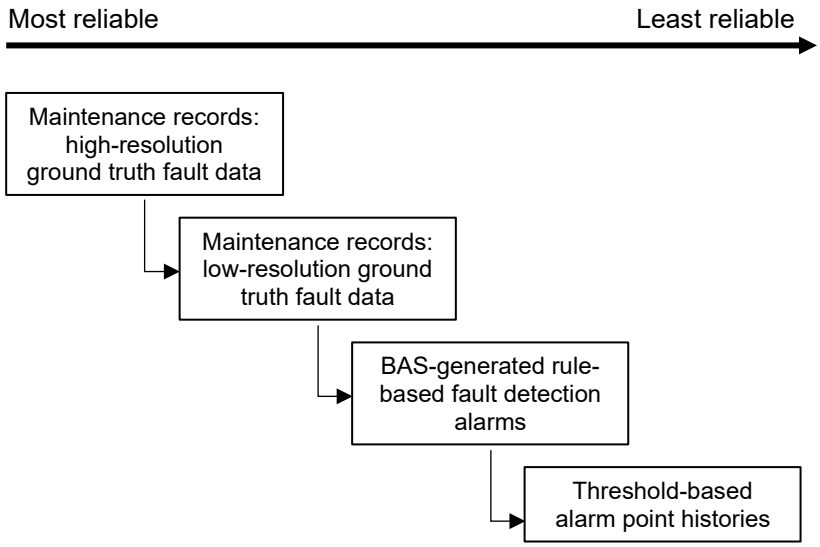

Figure 21. Hierarchy of Fault Validation Data Sources

Fault Detection Performance

The fault detection use case employed the baseline prediction model from the previous use case. The basic idea behind using the model as a fault detector is to compare the actual consumption values to the predicted values from the baseline model. Provided that the model was trained on sufficient fault-free data, a large deviation from the predicted values would indicate the presence of a fault in the actual dataset.

As described in the previous section, we do not have the necessary data to validate performance beyond a qualitative analysis. In the absence of those data, we are using the total number of alarms as a proxy for the fault data. To assess how the model might perform as fault detector we can compare the model accuracy to the number of alarms at different points in the dataset. A negative correlation between the regression score and number of alarms (i.e., the regression score is lower when there are more alarms) would indicate the model correctly identified a fault.

The model accuracy metrics are evaluated over the entire dataset at once. While this is useful for assessing how well the model performs overall, it does not provide information about how that performance varies over time. To use the baseline model as a potential fault detector, it is necessary to identify specific time windows during which the accuracy of the model decreases. To do this, we calculated the regression score of the predictions (compared to the actual values) over a rolling 24-hour window. The score during the rolling window assesses the model accuracy in every 24-hour period in the dataset, allowing us to see what period had the biggest deviation between actual consumption and the predicted values. Because the model can accurately predict consumption over the entire time period, large differences in the predicted and actual values in a 24-period indicate atypical consumption in the actual building.

Another 24-hour rolling window was used to count the total number of alarms in each 24-hour period; that count could then be compared to the rolling regression score at each time step. Because no metadata were available for the points in EBCS that allowed for easy disambiguation of specific alarm points, the names of each point were searched for the sub-string “alarm” and any point containing that string was classified as an alarm point.

While an increase in alarms does not definitively prove there was a fault, it does provide some indication that something was happening with the building operation that caused the alarm thresholds to be exceeded. This can provide some idea of how well the model *would* perform as a fault detector if validated fault data were available to evaluate the model. For the buildings tested there was no correlation between the rolling regression score and the number of alarms, but there were a few large spikes in alarms that could indicate, along with increased energy usage over expected, that there was a fault. Figure 22 shows one of the most striking examples of this where the model’s regression score reached a 6-month low point at the same time as a 6-month high in the number of alarms. This is a strong indicator of something happening in the building operation causing a deviation from predicted consumption, potentially a fault.

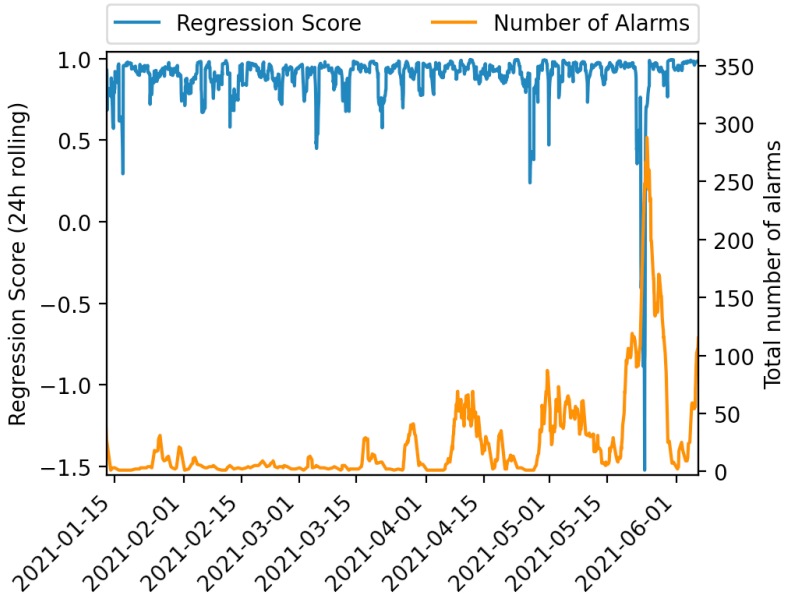

Figure 22. Potential Fault Indicated by a Low Regression Score at the Same Time as a Large Number of Alarms

The alarm proxy evaluation could not be used at every building. The total number of alarm points varied by building, with some containing hundreds of alarm points and others containing only a few. This further reduced the number of buildings that could be evaluated, because each building needed to have sufficient data for baseline modeling as well as a sufficient number of alarms points trended and alarming during the same period. Figure 23 presents an example of a building with only two alarm points that alarmed a total of four total times during the evaluation period. Given these limited data, even the alarm proxy evaluation was insufficient for assessing the baseline model’s fault detection potential.

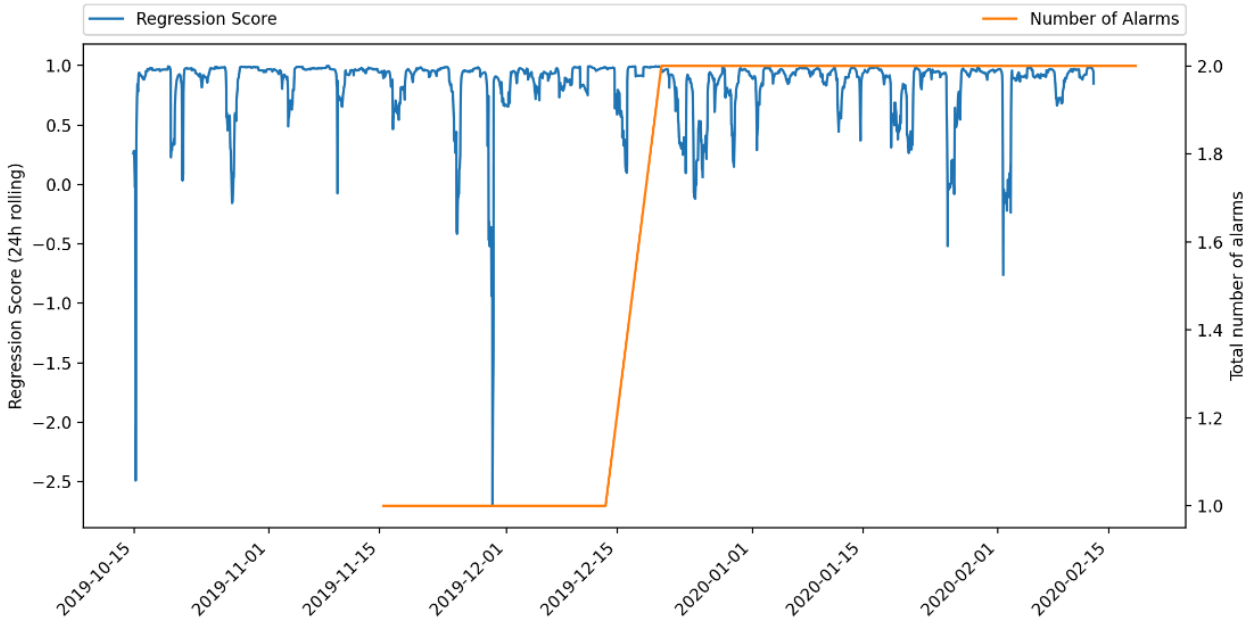

Figure 23. Example of a Building with Sufficient Data for a Baseline Model but Few Alarm Points

That said, Figure 22 suggests that combining traditional rule-based FDD and ML prediction could enhance fault detection. Given sufficient rule-based alarms data, the accuracy of a baseline prediction model could be correlated with the incidence of rule-based alarms and thus could offer an additional confirmatory signal indicating that there are performance issues to investigate at the building. Because traditional alarms can be so numerous, they are sometimes ignored by building operators; however, an additional whole-building prediction signal that deviates substantially from the observed energy consumption could help operators prioritize which alarms to investigate.

6.4 SITE PRIORITIZATION

Given the data quality and availability challenges present in the EBCS control data and MDMS utility data, the project team explored additional opportunities to apply ML to USAR data that could result in actionable results. To that end, an ML model was created to aid in site prioritization for energy-related projects. The model identifies sites that are consuming more energy annually than their peer USAR sites after accounting for differences such as building area and climate. The model was used in a virtual Installation Energy and Water Plan (v-IEWP) performed by PNNL for the USAR to identify sites where energy conservation measures should be prioritized.

The expected annual site energy was predicted using a random forest regression model and then compared to actual site energy use. The annual energy usage data comes from utility billing data that records monthly total utility consumption for each site. The model does not use MDMS utility meter data or EBCS building control data.

For input features, the regression model uses annual weather data at each site along with other site information to predict the total energy consumption for the year. The site information includes building types, areas, vintage, and the different utilities present at the site (e.g.,

electricity alone, or electricity and natural gas). The annual heating and cooling degree days are calculated for each site as well as a similar metric for the site's enthalpy. Table 8Table 8 lists the features used, along with a brief description.

Table 8. Input Features Used in the Expected Annual Site Energy Model

Feature	Description
HDD	Heating Degree Day (65° base temperature)
CDD	Cooling Degree Day (65° base temperature)
Enthalpy_dd	Enthalpy Degree Day (28 Btu/lb base enthalpy)
elec	Does the site have electrical consumption? (True/False)
gas	Does the site have natural gas consumption? (True/False)
propane	Does the site have propane consumption? (True/False)
fuel_oil	Does the site have fuel oil consumption? (True/False)
vintage	Weighted average vintage based on square footage
total_area	Total Building Area
num_shop	Number of "shop" buildings present at site
num_storage	Number of "storage" buildings present at site
num_training_center	Number of "training centers" buildings present at site
perc_shop	Percentage of the total building area that are "shop" buildings
perc_storage	Percentage of the total building area that are "storage" buildings
perc_training_center	Percentage of the total building area that are "training center" buildings
latitude	Site latitude
longitude	Site longitude

The model was trained on historical data from FYs 2015–2019 for all the sites across the 63rd, 81st, 88th, and 99th Readiness Divisions. This analysis is similar to normalizing energy usage for either heating or cooling degree days, but is more robust because it takes into account additional information on the site. Table 9 presents the performance metrics of the model, which show a high prediction accuracy.

Table 9. Performance Metrics of the Expected Annual Site Energy Model

Full Set r^2 Score	Adjusted r^2 Score	CV(RMSE)	NMBE
0.948	0.948	25.93%	0.22%

Next, the input features from the most recent fiscal year available (FY 2020) were used to predict the expected total energy for each site in that year. The percent difference between the expected energy values and the actual values was then calculated to identify the sites that are using much more or less energy than expected of a typical USAR site with similar characteristics. Figure 24 plots the percent difference for each site in the region against the site's total annual energy consumption. The sites that have higher than expected energy usage (>25% percent difference) are labeled in the figure. These sites should be prioritized and investigated to identify the cause of the excess energy usage.

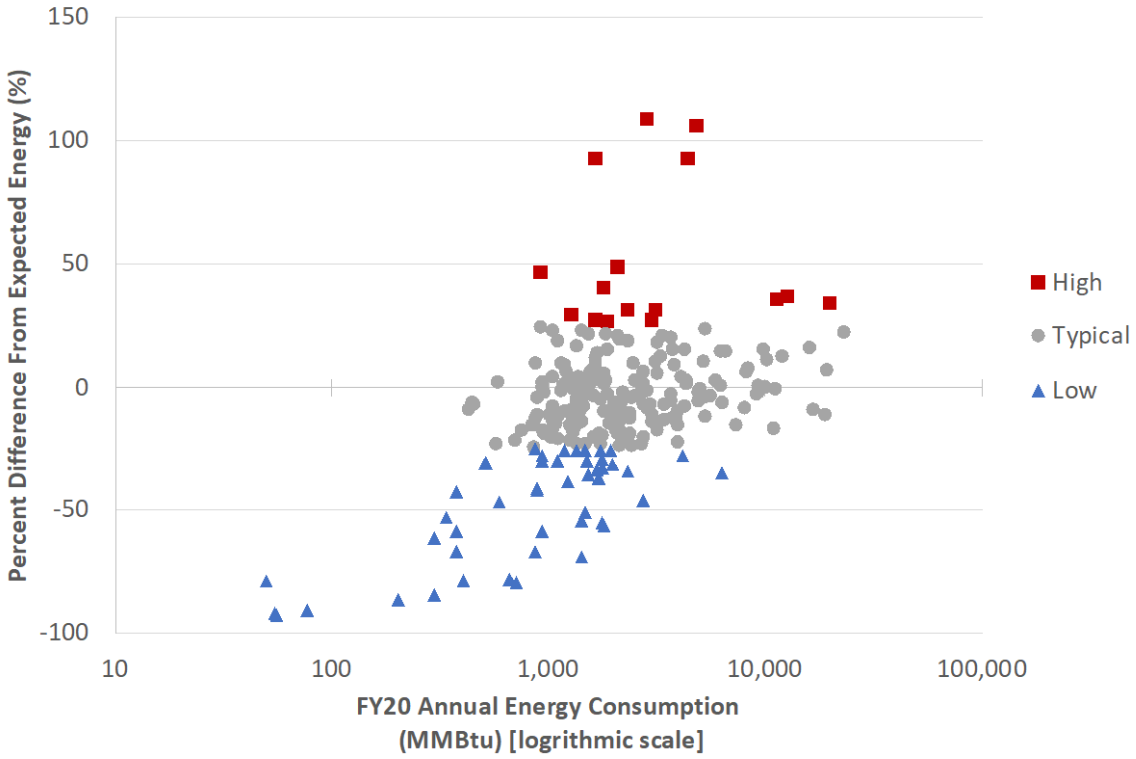

Figure 24. Site Annual Consumption Compared vs. Model Prediction Error

Prediction Task Complexity

Data-driven tasks, like ML, are inherently constrained by the available input data for the task. There is a general correlation between the task complexity and amount of data required to complete the task, i.e., the more difficult the task, the more data are required.

Two important aspects of data availability for any given ML task are the granularity of data and the coverage of the prediction space. The granularity of data describes the level of detail provided by a dataset. For time-series data, the resolution would be the temporal resolution, or the time interval, at which the data were recorded (e.g., 15-minute, hourly, annually, etc.). Coverage of the prediction space refers to the portion of the possible model predictions covered by the data. For a dataset to have good coverage it must contain samples of each distinct model output. An example of a dataset with excellent coverage is the Modified National Institute of Standards and Technology (MNIST) database (Deng 2012), containing 70,000 images of handwritten digits (0-9) because it has many examples of each base-10 number.

These two qualities of a dataset dictate what is and is not possible with the dataset; the complexity and range of outputs directly depends on the input data available for training. For example, the MNIST dataset is perfect for recognizing handwritten numbers, but could not be used to train a model to recognize letters or words because it contains no letters.

It may be counterintuitive that aggregated monthly utility consumption data could produce a more actionable models than much more granular (e.g., 15-minute or hourly) building controls and advanced utility meter data. However, the difference in available data and prediction task

difficulty can combine to enable a model trained on less granular data to produce a more useful prediction.

To illustrate this point, consider an analogy of an image classifier that is attempting to identify dogs, cats, and horses (good, bad, and average performing buildings in our case). For the image classifier, thousands of 128x128 pixel images containing dogs, cats, and horses (analogous to annual utility usage in this example) can train a much better identification model than dozens of 100-megapixel (~12,000 x 9,000 pixels) cat images (analogous to 15-minute utility interval data). The three orders of magnitude increase in data resolution would not help the model identify dogs or horses from cats because it has only seen cats. Similarly, the 15-minute interval data, which are 8,760 times higher resolution than annual data, are ultimately less effective at separating poorly performing buildings because the data are only available for a small subset of all the buildings.

Predicting total annual energy consumption at an entire site is a much simpler task than predicting 15-minute or hourly energy consumption at a single building. Total annual energy consumption is more stable than hourly or sub-hourly interval energy consumption and is unlikely to change drastically from year to year absent major changes or malfunctions in the buildings' operation. In contrast, 15-minute and hourly energy consumption is highly dependent on occupant behavior (e.g., occupancy patterns) or operational actions (e.g., a manual change over from summer to winter operation). These human actions are not always performed predictably from day to day or year to year, making it difficult for the model to distinguish between normal fluctuations in energy usage and conditions that indicate a problem in the building's operation, such as a fault.

In terms of data quality, the monthly utility billing data are manually curated; a human enters the monthly utility totals into the billing system and checks the inputs for errors, thereby deriving a mostly gap-free dataset. In contrast, the EBCS control point data and MDMS utility meter data are part of automated data pipelines that record data without human interaction or error checking. Not only is the data quality higher in the monthly utility billing system, but it also covers more sites and a longer time period; specifically, it covers around 530 USAR sites over 6 years.

Longer-term data availability gives the model an opportunity to observe a site over multiple years and allows a change in the site's energy consumption from its "normal" consumption to be detected. In contrast, even where EBCS and MDMS data are jointly available for a building, they only cover more than a year in a few instances. Where there is only a year or less of data, the model only observes a point in time once during training, making it harder for it to distinguish normal from abnormal operations.

7.0 COST ASSESSMENT

At the onset of this project, the team proposed to develop and validate the expected life cycle operational costs for incorporating ML into building systems data analysis structures. The data required to run a traditional cost model based on the outcomes of energy savings and maintenance cost reduction are described in Table 10.

Table 10. Inputs to a Traditional Cost Model for ML Technology Implementation

1. Installation costs	Labor and material required to install ML on system, including data preparation costs. Analytics will be targeted at the use cases.
2. Facility operational costs	Reduction in energy required vs. baseline data. Examples include reducing operating hours, fan speed, or space temperature.
3. ML system maintenance	• Frequency of required maintenance on ML algorithms • Labor per ML maintenance action
4. ML lifetime	Estimate based on the effectiveness of ML algorithms during demonstration.
5. Facility maintenance	Estimate of avoided maintenance and downtime per ML action identified.

This demonstration highlighted the fact that the installation costs and timeline can be much greater than originally planned, in large part due to the data preparation costs. The general cost model is described below. The specific calculation varies depending on the use case, but the economic benefits component would relate back to one or more of the performance objectives identified in Section 4.0, namely analysis effort, building energy use, and system maintenance.

Due to the constraints of the project, interventions could not be implemented during the demonstration; therefore, assessments of operational cost savings and maintenance avoided could not be performed.

7.1 GENERAL COST MODEL

The project team proposed a simple model that captures the economic value of a solution as the difference in economic benefits and the solution cost:

$$EV = EB - SC \quad (1)$$

where

EV = net economic value of proposed solution ($),
EB = economic benefits that are derived as a result of a solution ($),
SC = cost of solution ($).

Each use case solution would have slight modifications to the calculation, with an overall structure shown in Equation (1). Economic benefits could be calculated as an annual average or projected over an expected lifetime, depending on the use case.

7.2 COST DRIVERS

For ML projects, labor is the largest, if not the only, component of the cost. DoD deployments could include purchasing the ML capability through a software as a service provider. PNNL did not purchase software for this project and relied on open-source Python libraries and staff labor to execute the project.

During different phases of the project, the team discussed the challenges and findings with peers in academia, other laboratories, and industry experts. The cost drivers for this project were similar to those of others undertaking similar efforts. The largest cost, due to manpower is in the data preparation and cleaning stages as shown in Figure 25.

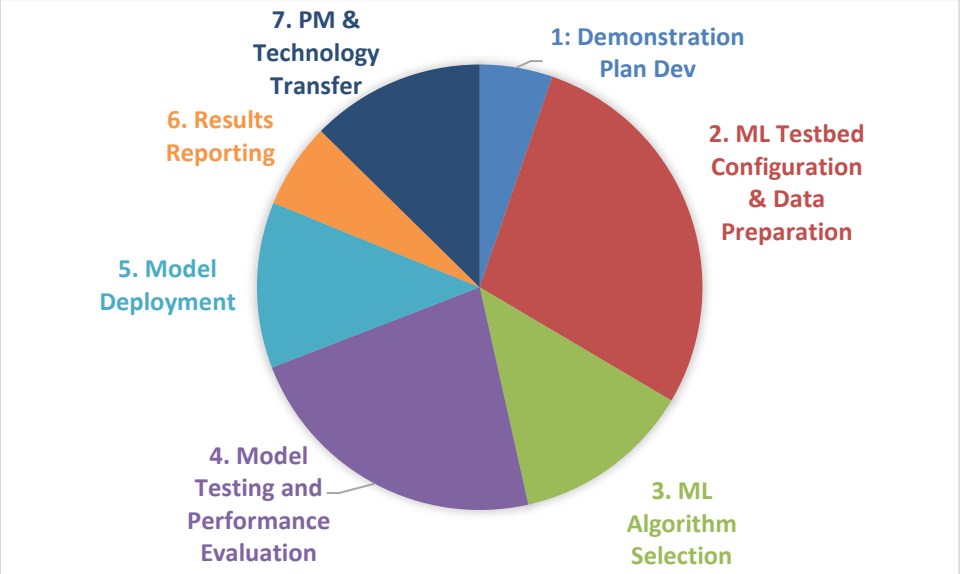

Figure 25. Cost Breakdown for Demonstration Project by Major Task

A demonstration project of this type spends more effort in identifying appropriate tools, testing methods and quantifying accuracy than a standard deployment would incur. The work breakdown structure for the project consisted of seven major task areas:

- Task 1. Demonstration Plan Development
  - 1.1 Write draft Demonstration Plan
  - 1.2 Draft plan review by ESTCP committee
  - 1.3 Comments incorporated & final plan submitted
- Task 2. Machine Learning Testbed Configuration
  - 2.1 Development environment configuration
    - 2.1.1 Configure ML development software/libraries
    - 2.1.2 System configuration documentation
    - 2.1.3 Define codebase structure and development process
    - 2.1.4 ML life cycle management planning
  - 2.2. Data Preparation

        - 2.2.1 Establish data pipeline
        - 2.2.2 Data documentation
        - 2.2.3 Exploratory data analysis and quality characterization
        - 2.2.4 Define datasets for each ML use case
- Task 3. Machine Learning Algorithm Selection
    - 3.1 Define use cases
    - 3.2 Identify ML methods to evaluate for each use case
    - 3.3 Expert elicitation
- Task 4. Model Testing and Performance Evaluation
    - 4.1 ML model development and testing
        - 4.1.1 Iterative testing
        - 4.1.2 Model validation
    - 4.2 Performance evaluation
        - 4.2.1 Performance objective: analysis effort
        - 4.2.4 Cost assessment
- Task 5. Model Deployment
    - 5.1 Configure deployment environment
    - 5.2 Deploy ML tools
- Task 6. Results Reporting
    - 6.1 White Papers
    - 6.2 Final Technical Report
    - 6.3 End of Project Presentation
- Task 7. Project Management and Technology Transfer
    - 7.1 Quarterly Progress Reports
    - 7.2 Annual In-Progress Reviews
    - 7.3 Annual ESTCP SERDP Symposia
    - 7.4 Outreach and Conference Presentations

7.3 COST ANALYSIS CONSIDERATIONS

The cost-benefit analysis for applying ML to building controls and utility metering data relies on a number of factors. Meters and sensors themselves do not save energy; it is actions taken based on the information provided that result in economic benefits. Applying advanced analytics and ML to data can improve the recommendations and information provided to the user; resulting in either autonomous changes or manual changes to systems. The DOE Federal Energy Management Program has partnered with the National Laboratories over the years to provide guidance on how to justify installing meters from a cost and benefit perspective (Parker et al. 2015). An expansion of the best practice developed for this project (see Appendix C) could be beneficial for DoD as the data quality and connection of data points improves.

8.0 TECHNOLOGY TRANSFER

There is potential to extend this demonstration from the USAR EBCS to similar systems operated by other DoD commands, which have also deployed thousands of smart meters and BCSs at their installations. The Army is also moving toward requiring connected control systems at their installations, as required by Army Policy and the Army Climate Strategy (2022). The methodologies, results, and lessons learned from this demonstration are documented and will be made available across all DoD commands. PNNL disseminated the results of this effort through this technical report, (pending) journal articles, and conference presentations.

The ML tools deployed and validated for the USAR could be deployed to other commands immediately following completion of this effort. The National Guard would be a prime candidate because of its similarity of building types and missions. These techniques also apply to larger military installations, although additional ML algorithm training may be needed for larger, more complex facilities.

The greatest challenge for an ML platform is to ensure cybersecurity by building a hardened system. Within the Army, the G-6 organization controls hardware, software, and all connections to the network requiring a Certificate of Networthiness (now moving to Risk Management Framework). The USAR system, including the metering and building controls, has G-6 approval and will continue to provide updated patches and requirements. Transitioning to other commands within the Army (that have separate information technology groups) would move quickly because of the work with USAR G-6. Transition to the Navy or Air Force would require a restart of the approval process, which can add up to 1 year to the transition process.

Formal technology transfer activities includes the following:

1. American Society of Heating, Refrigeration, and Air Conditioning Engineers (ASHRAE) Annual conference (June 2021) presentation "*Applying Machine Learning to Enhance Building Performance at US Army Reserve Centers*".[7]
2. National Institute of Building Sciences Building Innovation Conference (September 2021) presentation "*Applying Artificial Intelligence to Buildings with Imperfect Data"*.[8]
3. Paper describing the "Challenges to Applying ML to Existing Building Energy and Controls Data at Scale" (in Draft to be submitted).
4. Paper describing "ML for Buildings: A Use Case Perspective" (in Draft to be submitted). The audience focus is on DoD and the document will be distributed through Headquarters communication channels. Guidance for DoD when considering ML as a procured service (in Appendix C).

[7] ML for USAR_ASHRAE_2021.

[8] NIBS Presentation_BI2021_

5. Progress and innovation highlighted as part of the Army Energy Managers Community of Practice Webinar, chaired by Ms. Christine Ploschke, Acting Deputy Assistant Secretary of the Army, in February 2022. Over 180 energy managers and public works staff from the Army were in attendance.
6. Coordinated with the USAR EBCS technical team to share the code base and leverage the work for other related efforts (e.g., Control Score algorithm deployment).
7. Open-source release of software framework to be used in future analytics deployments.
8. SERDP & ESTCP Symposia: including the poster sessions 2019, 2020 and 2021, and Presenting at the Energy and Water session in 2020.
9. ESTCP In Progress Reviews in 2020, 2021 and 2022.
10. Final Technical Report.
11. Out-brief webinar with DoD Energy Managers after submission of the final deliverables.

Because the primary purpose of the tool deployment was to show what ML can accomplish when applied to the USAR's existing data and not to deliver a production-grade web application, the demonstration application has been deployed on the PNNL network in an environment that mirrors the EBCS deployment environment. This allows the functionality of the tool to be demonstrated to users without incurring the monetary and time costs of deployment on the EBCS system.

The mirrored environment of ARNet on PNNL's network consists of a Windows Virtual Machine (VM) that hosts a copy of the EBCS Microsoft SQL (MSSQL) database. The ML demonstration application is hosted on another VM on PNNL's network inside a Docker container. Docker containers are a type of virtualized operating system that allow programs to be neatly packaged with their dependencies and replicated on all types of systems (e.g., Windows or Unix operating systems), thereby simplifying deployment. The "Dockerized" application can easily be deployed on any VM with Docker installed on ARNet.

Another PNNL project developing an application called Control Score is also exploring ways to deploy a Python-based application that uses EBCS data on ARNet. Our demonstration team worked with the Control Score team to leverage our joint efforts and lower the development effort for both teams. The two apps are implemented in the same web framework application, although they are completely separated from each other. Both project teams and the USAR benefited from this configuration due to the efficiencies of a single deployment effort. The application architecture and mirrored computing environment are shown in Figure 26.

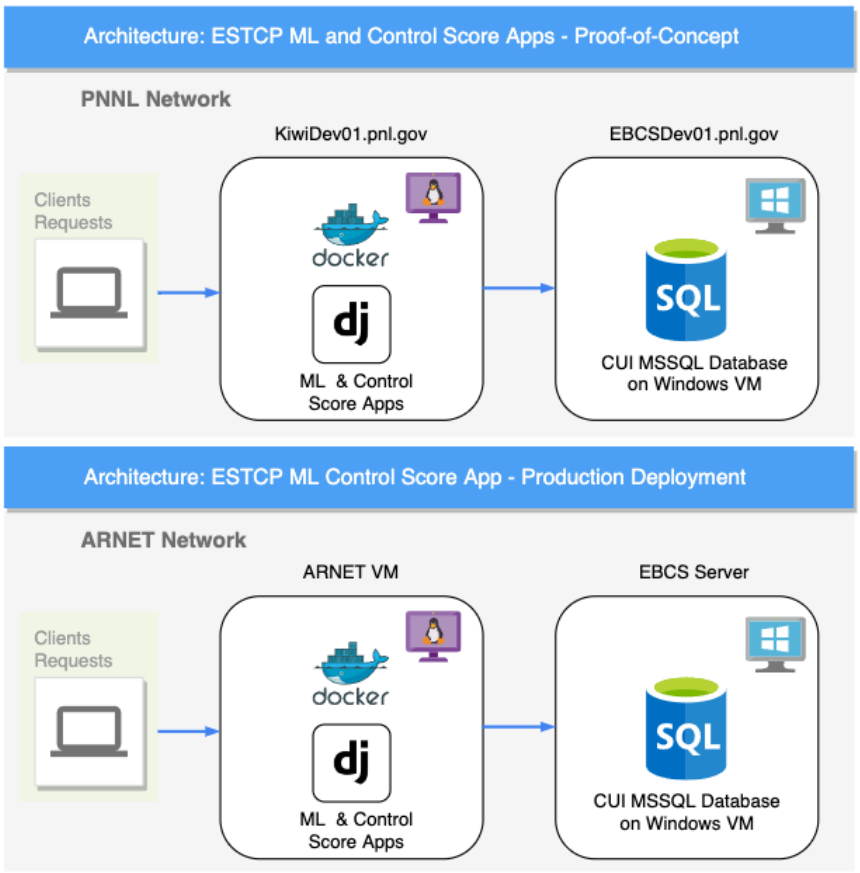

Figure 26. Application Architecture for Proof-of-Concept and Production Deployment

The demonstration deployment application has a live connection to the MSSQL database that stores the EBCS control data and the utility meter data captured by MDMS meters. The application allows meter data to be viewed in any time window for each building. It also trains and stores a baseline prediction model for any selected building. The predicted consumption can be viewed along with the actual consumption during any selected window of time. Figure 27 shows screenshots depicting both capabilities, with utility meter data displayed on top and baseline model predictions compared to actual consumption displayed below.

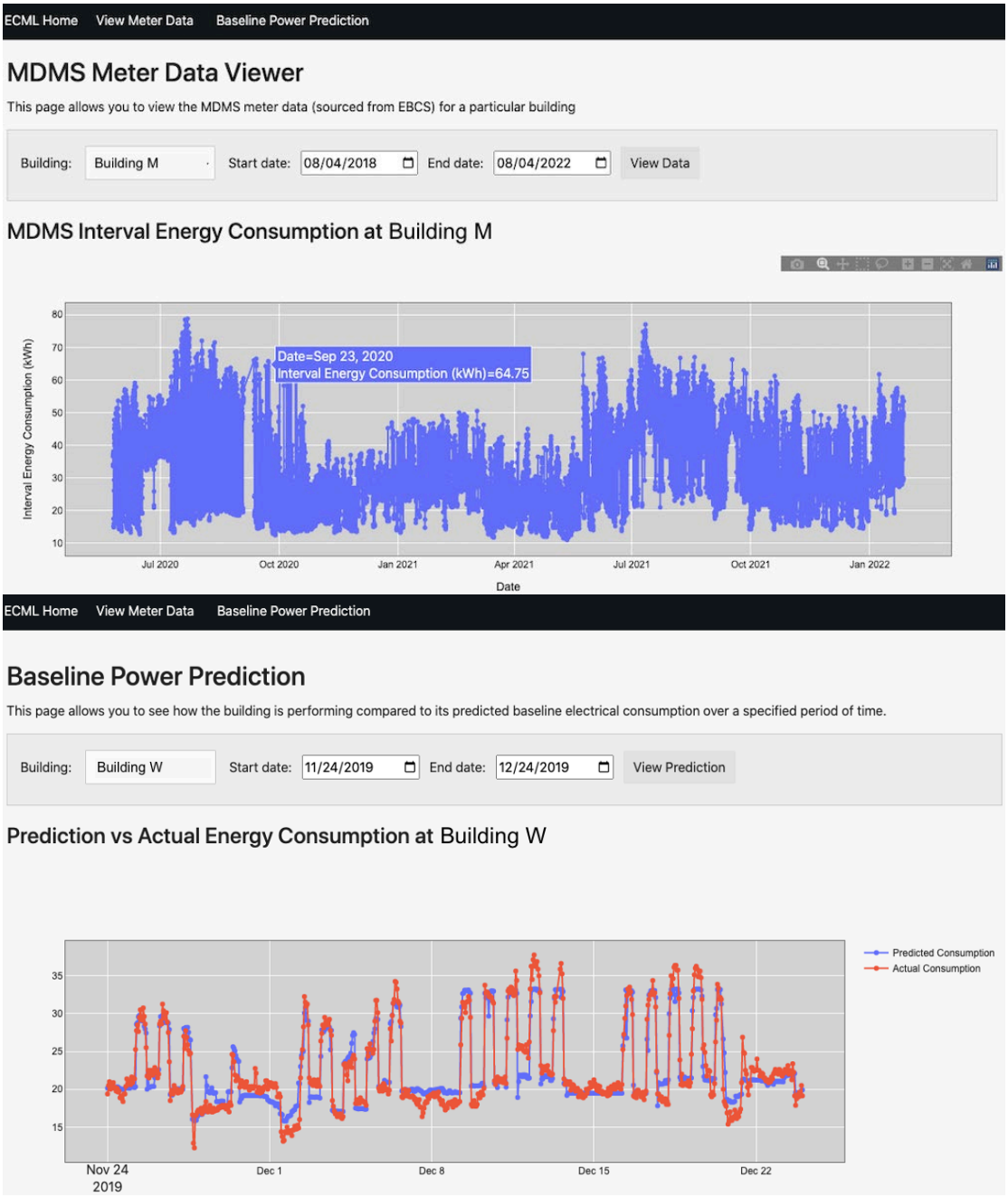

Figure 27. Screenshots from the Deployment Application Demonstration

The open-source software framework (Item 7 in technology transfer list) builds upon and extends existing tools for building and managing related computational environments.[9] Simply put, a computing environment is the active set of software tools and libraries that are used when running a program. Traditionally these tools provide software package management and deployment workflows for a single environment. Variations are created manually. In contrast, fundamentally, the proposed framework compositionally relates multiple computational

[9] https://github.com/pnnl/hydraconda and https://www.osti.gov/doecode/biblio/74986

environments, which provides a basis for programmatically building them for a variety of scenarios. By enabling multiple related environments to be used by both software engineering-oriented developers as well as data scientists in the same codebase, both systemized as well as exploratory programs can be managed.

For example, data processing code can be systemized and centrally managed (as a unit), and (multiple) exploratory codes (that depend on data processing) can be managed separately, all while retaining the relationship between them. Furthermore, “finalized” analyses could be deployed as (delivered) solutions.

The primary audiences for this framework are integrated analytics teams consisting of data scientists as well as software engineers. The framework could be used for any analytics development project. It could benefit future DoD/federal deployments by automating, standardizing, and expediting development and solution delivery.

9.0 IMPLEMENTATION ISSUES

Several challenges that were not well understood at the outset of the demonstration caused delays and created barriers to successful use case investigation and deployment of ML algorithms.

9.1 DATA QUALITY ISSUES

Data-driven approaches like ML are inherently reliant on the input data available. If sufficient data are not available, it is not possible to implement them. Throughout this demonstration, a variety of data quality issues were identified that posed challenges to achieving the originally stated performance objectives. These issues included missing data, suspicious or invalid readings, time stamp offsets, and reporting unit miscalibration.

Data Availability and Quality

Data availability was the biggest data challenge faced by the project by far. While there are hundreds of USAR buildings with advanced meters incorporated into the Army Metering Program's MDMS, only a fraction of the buildings are also incorporated into EBCS. Meanwhile, many of the buildings that are incorporated into EBCS do not have MDMS advanced utility meters. Thus, the limited overlap of buildings with both EBCS and MDMS data restricted the candidate buildings significantly prior to other data volume and quality considerations.

The use case prioritization matrix (Appendix C) describes the data quality requirements for the use cases that were implemented in this project. Filtering out the buildings that do not meet these data requirements further reduced the number of buildings available for the project. The down-selection from all possible buildings in MDMS and EBCS to the buildings with adequate data for ML is illustrated in Figure 28.

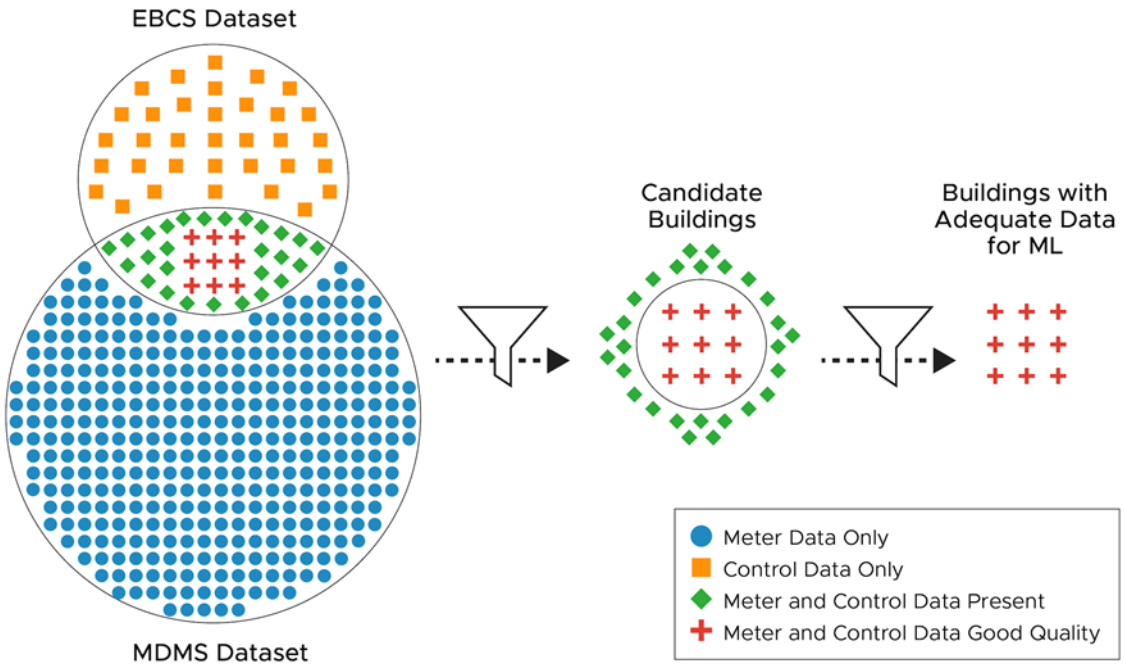

Figure 28. Buildings with Adequate Data for ML

Among the EBCS buildings for which sufficient data were available to meet the testing and training requirements of ML model development, data quality varied widely from building to building. Data quality problems derive from many sources, among them:

- Interruptions in network connectivity
  - Routine networking issues
  - Cybersecurity-related interruptions
  - Power loss
- Data synchronization problems across the Army advanced metering infrastructure (AMI) network
  - Data transfer losses between advanced meter, Building Point of Connection (BPOC), Enterprise Energy Data Reporting System (EEDRS)/ Utility Monitoring and Control System (UMCS), MDMS gateway, and MDMS central server
  - Data storage limitations of field devices
  - Limited lookback windows for data queries
  - Variation in data acquisition, formatting, and storage protocols by EEDRS/UMCS software (e.g., JCI Metasys, Tridium Niagara N4)
- Miscalibration of measurement devices
  - Meter multipliers
  - Incorrect reporting units
- Interoperability constraints between devices using different communications protocols and data formats.

While many of these problems presented challenges, consistent data availability was by far the most significant. Even where buildings were incorporated into both the MDMS and EBCS platforms, missing data in one or both sets of data often precluded using the building in this project. The most prominent example of these challenges is a loss of all meter data from approximately June 2018 to September 2019 due to a networking connectivity issue. Meter data availability is visualized in Figure 29, in which each horizontal line represents a distinct meter and white space represents a day of missing data. The outage affecting all MDMS meters can clearly be seen in the left figure, and the continued data availability challenges are evident in both figures as shown by the amount of white space present in the figures (perfect reporting would result in a solid-colored rectangle).

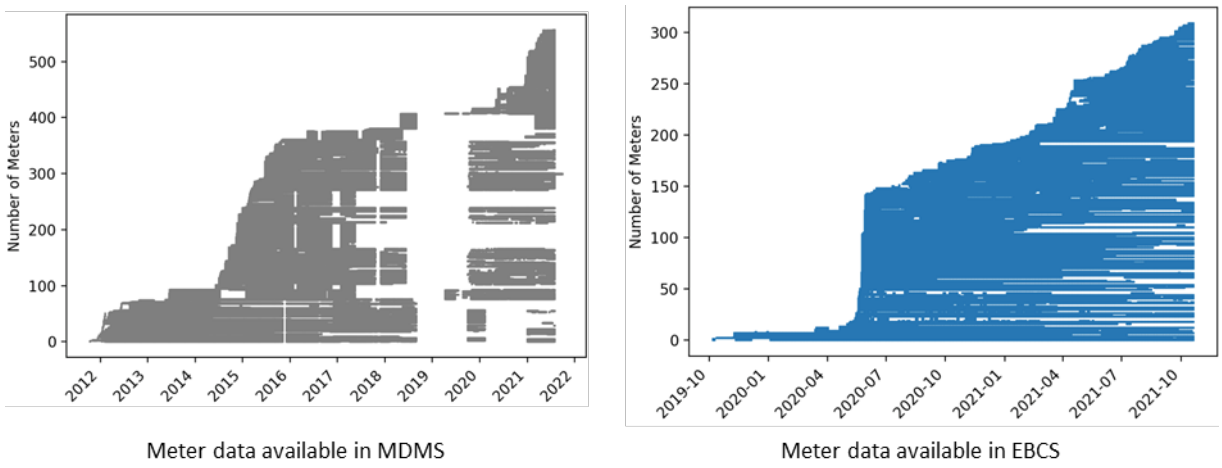

Figure 29. Meter Data Availability in MDMS and EBCS

Up to a point, data quality problems can be mitigated by using preprocessing methods such as interpolation for short gaps in time-series data and anomaly detection for erroneous readings. Such processes are usually human-supervised to ensure that gaps are not so large that interpolation would generate specious data, or that anomaly detection would filter out potentially valid readings. There are limits to how many missing or bad readings can be allowed in a time series before it becomes unusable from the standpoint of ML model training, however. The specific determination depends on the quantity of data available, the use cases and ML algorithms under consideration, and the distribution of bad data within the time series.

Data Integration

Data integration across sources introduced an additional challenge to the demonstration. Even simple use cases typically require merging multiple sources; for example, the whole-building load prediction models trained in this demonstration used outdoor air temperature data from NOAA as an input feature. This required integrating utility metering data from MDMS with temperature data from an external source. When using high-quality sources such as NOAA historical weather data, data integration is a straightforward process; however, combining multiple sources of poor-quality building data can render the integrated dataset unusable for ML model development.

In particular, integrating multiple time series introduces the problem of non-overlapping coverage. Figure 30 illustrates the problem of non-overlapping coverage for control points at multiple EBCS buildings; colored regions indicate that a reading is available for a control point at a particular time step, while white space indicates that no reading is available. There is only a roughly three-month window in 2020 during which readings are available for all points at all buildings.

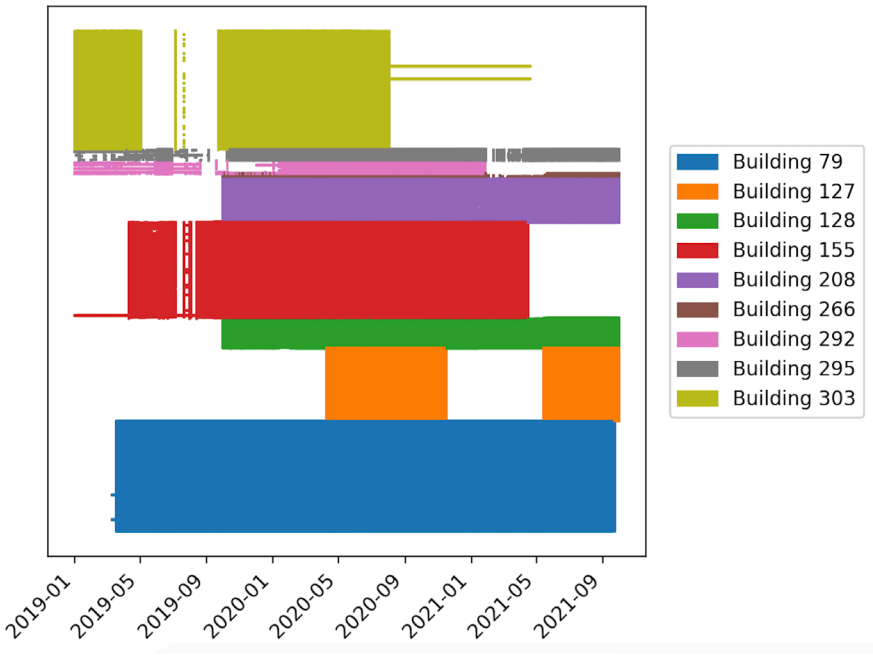

Figure 30. Non-overlapping Coverage of Control Point Readings for Multiple EBCS Buildings

A useful feature of ML is the ability to extend a model's predictive capabilities from one target to another, assuming that the phenomena being modeled are sufficiently similar across prediction tasks. It is more efficient to leverage a pre-trained model to make predictions about a new target than to train an entirely new model from scratch. The extension of a pre-trained model from one prediction task to another is known as transfer learning.

At the outset of the demonstration, the project team expected that transfer learning would allow for the extension of controls-oriented models to many buildings; however, the control points trended in EBCS differ substantially from building to building, which limits model transfer between different buildings. Variation in point trending at different buildings occurs for a number of reasons, including the following:

- There are many different building designs with a wide variety of HVAC and other energy-consuming systems.
- Even in two buildings with identical systems, different points are chosen to be trended (i.e., their data recorded for long-term storage).
- Building owners have numerous competing interests, among which achieving energy efficiency through automation is often a lower priority.
- Field controllers and automation systems are offered by a wide range of vendors, and many rely on proprietary standards that are not interoperable with each other (this is particularly true with pre-BACnet legacy systems).
- Only some systems and equipment are automated.

Unavailable Metadata and Inconsistent Point Naming Conventions

One of the largest data quality issues this project faced was a lack of available metadata for the EBCS points. Metadata that describe what the point data represent (e.g., a space temperature measurement or discharge pressure set point) and how they relate to other points in the building (e.g., identifying all the points belonging to a single piece of equipment) are essential for many of the use cases explored in this project. Without having these data formally documented, the project team attempted to infer this information, to the extent possible, from the name of the points. The robustness of this inference relies on how well the naming convention describes that information and how rigorously the naming convention was followed.

Although a meter naming convention is specified in the U.S. Army Corp of Engineers (USACE) Army Metering Program Guidance for Advanced Meters (USACE 2016), it is inconsistently applied across meters at both the UMCS/EEDRS level as well as in MDMS. Efforts have been made over time to update these names to conform with the current standard, but legacy issues remain. Even with the convention in place, in some cases meter install contractors and Directorate of Public Works (DPW) personnel implemented an alternative approach instead.

For the purposes of this demonstration, however, inconsistent meter naming was a relatively minor issue when compared to the much greater challenge of inconsistent point naming conventions for BCS point histories across different buildings in EBCS. There is usually only one advanced meter per utility at each USAR building, meaning that manual resolution of meter names is possible; however, BCS trend hundreds or thousands of points per building. Investigation of EBCS data yielded a wide range of point naming conventions.

Inconsistent point naming conventions pose a major challenge to scaling ML models from one to multiple buildings, and even to implementing models at a single building. The first issue is that of interpretability. In a BCS, control points are associated with graphical models, accessible through the BCS graphical user interface, which aid human building operators in interpreting the physical system that the points represent. Figure 31 presents an example from EBCS.

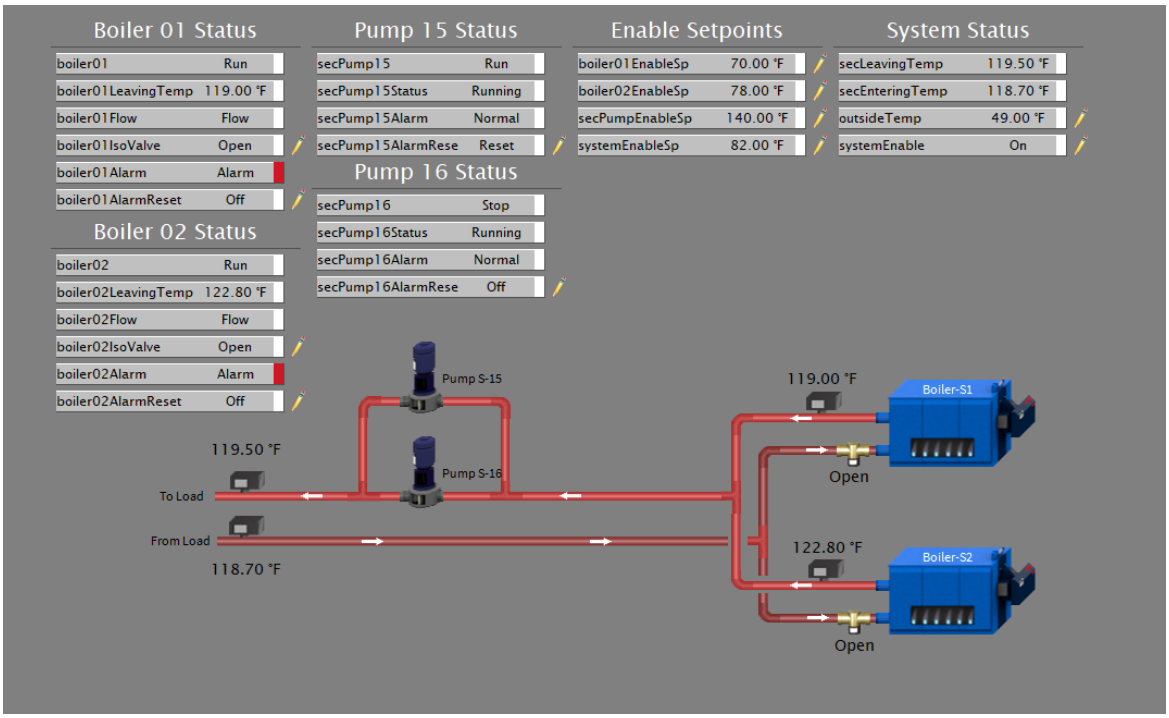

Figure 31. Example Graphical Model of a Boiler System in EBCS

Conventional ML models cannot make use of this graphical context the way that human operators do, however; instead, the point name needs to be linked to metadata that describe the point unambiguously, i.e., a semantic model or metadata schema. In the absence of such metadata, the model developer must interpret the meaning of point names and incorporate the appropriate points into the model accordingly. With hundreds or thousands of points to review per building, this is a time-consuming process, prone to errors of interpretation and judgment. And because naming conventions are inconsistent between buildings, the interpretations applied to a particular set of points at one building often do not apply to the next building.

An automated approach to the problem is to develop a set of syntactic rules for automatically parsing point names (e.g., "all point names ending in '_sp' refer to a control set point") and assigning semantic meaning to them. Inconsistent point naming conventions inhibit the scalability of this approach, however; a set point at one building might be coded as "_sp" and at another as "-set-point". In some cases, these conventions can vary even among points within the same building. For the model developer, this complexity leads to a disproportionate expenditure of time on point name resolution, involving some ad hoc combination of syntactic parsing rules and data modeling.

The USAR is far from alone in terms of point naming consistency issues. Among energy management information system (EMIS) developers, it is a widely recognized problem throughout the commercial buildings sector.

While there are a number of emerging standards for point ontologies (e.g., Project Haystack, Brick, and ASHRAE Standard 223P), they have not been widely adopted throughout the building industry. Creating the models is currently a manual process, which can be labor-intensive, particularly for larger buildings with many points. This is especially true for existing buildings that may lack drawings of the building and associated systems. There is active research being

conducted on data-driven approaches to automatically determine the point ontology, but the accuracy of these methods varies significantly.

Limited Detail in Maintenance Logs

To implement effective fault diagnosis and predictive maintenance models, detailed maintenance histories are required. This is a basic requirement of supervised ML tasks—for the model to predict a target outcome, it must be trained on data that include both predictive features and observed outcomes. Empirical observations of the target are required to validate the accuracy of the model predictions. If target data are missing or are not sufficiently detailed and precise, it is not possible to build a model to predict that target. In the case of fault diagnosis, a high degree of specificity is necessary for the diagnosis to be useful.

The project team reviewed maintenance data from the USAR CSS, the enterprise system for tracking work orders at USAR sites. After the review, the team concluded that the information contained in CSS was not sufficiently detailed to support the development of useful ML models for fault diagnosis and predictive maintenance. Table 11 presents a sample of maintenance data contained in CSS.

Table 11. Example Data from CSS

Summary	Actual Cost	Priority	Request Status	Created	Work Complete
No hot water in building	$2,032.98	Routine	Closed	9/8/2017	5/10/2017
HVAC Boiler - Building 8001	$1,750.00	Emergency	Closed	1/16/2018	1/19/2018
HVAC - Mini split	$276.00	Routine	Closed	2/23/2018	5/31/2018
Bay heaters - Inoperable	$1,620.00	Routine	Approved for Closure	11/28/2018	3/1/2019
hot water pump	$1,667.61	Urgent	Approved for Closure	1/22/2019	3/25/2019

As the table illustrates, the CSS ticket summary generally provides a brief description of the equipment in need of maintenance or replacement, but no additional detail. To be useful for fault diagnosis or predictive maintenance, more detail is required about the specific component(s) in the equipment that experienced the fault, the fault itself, what corrective actions were taken, and the times that events occurred. In CSS, this information is generally unavailable. Timestamps are included to document when the ticket was created and when work was completed, but they are not necessarily reliable indicators of the actual timing of events; for example, in the first row of Table 11, the ticket created date is 9/8/2017, while the work completed date is 5/10/2017, roughly four months earlier. This suggests that the ticket was created retrospectively for recordkeeping and accounting purposes, rather than to track maintenance events as they occurred. More broadly, it implies that CSS was not designed with the intent of providing data inputs to ML models in mind, but for other purposes that are important to the USAR. This point bears further discussion, because it applies to other systems in this demonstration as well.

Misalignment of System Functional Design and ML Objectives

At the outset of the demonstration, the project team expected to encounter challenges with data quality and availability based on prior analyses of MDMS data. Data integration across multiple disparate sources was another anticipated challenge. One lesson learned since, however, is that it is critical to understand how the functional design of a system can drive what and how data are captured by the system and ultimately how useful they are for the purposes of ML modeling. In the case of MDMS, this is straightforward; the system was designed for the purpose of capturing energy and water use data at Army buildings and making that data available to energy managers for analysis and further action. The functional design of the system aligns closely with the objectives of this demonstration, and accordingly the data captured by the system were useful for developing baseline prediction and fault detection models. In the case of CSS, however, the functional design never anticipated that the system would be used to provide inputs to fault diagnosis or predictive maintenance models, and accordingly the data do not support those use cases.

EBCS represents a more complicated case. EBCS provides a standardized Niagara-based platform connecting USAR BCSs throughout the nation. Collecting standardized building operation data to support advanced applications such as ML models is a key EBCS function. Historical data stored at the EBCS server level heavily rely on the connectivity to the integrated BCSs. In the EBCS functional design, point histories are required to be temporarily saved at the local integration controller for up to 14 calendar days; however, in practice, the Tridium Niagara N4 software on which EBCS runs periodically stops recording point histories, and recording must be manually reinitiated (see Figure 30). If the connection is lost beyond the 2 weeks when data are temporarily stored at the local integration controller, building operation data gaps occur. During the course of this ESTCP project, the EBCS program experienced some extended disconnectivity attributed to a lack of timely responses to broken integration controllers and a system upgrade from Niagara AX to Niagara N4. Further, as discussed above, some inconsistent point naming conventions were implemented across buildings that affect ML model scalability. The data reliability, accuracy, and consistency gaps exposed by this demonstration will inform the USAR EBCS program. The USAR EBCS program is addressing these data gaps to optimize the EBCS's intended data applications.

9.2 CYBERSECURITY ISSUES

When the demonstration began, USAR was planning to transition a number of ARNet-hosted systems to the Microsoft Azure cloud platform. Among the systems planned for transition was EBCS. To date, however, that transition has not occurred; consequently, throughout the demonstration there has been uncertainty regarding the ultimate deployment environment and the cybersecurity requirements associated with deployment. Because of the ongoing uncertainty about deployment requirements, the ML tools developed during this demonstration were not deployed to EBCS directly but instead are hosted on the PNNL network. The team will continue to work with USAR after the demonstration concludes to plan a future EBCS deployment.

Cybersecurity is a central pillar of the DoD's information systems, and ARNet has strict cybersecurity requirements. These requirements present challenges to deploying a new application on the network. Some of these challenges are described below. Although the project team is actively working to address these obstacles, they prevented a deployment of the

application on the ARNet. Section 8.0 describes the demonstration deployment on PNNL's network.

USAR requires any IT system to be approved through an application process before being granted an ATO on their network. The ML application developed in this project is written in the Python programming language, but there is no current ATO for the Python programming language.

Access to ARNet is also tightly controlled. A DoD Common Access Card (CAC) is required to access any system on the network. While the project team included several team members with CACs and an embedded staff member, the primary data scientists and software developers do not have CACs, which complicates development on a USAR system.

The "Dockerized" ML application can be easily deployed in a wide range of systems, but it is not an isolated application; it requires a connection to the EBCS database hosted on the same network. The connection needs to be allowed through the various firewalls in place between the machine hosting the database and the one hosting the application. Without access to the deployment environment and close collaboration with the system operators, this configuration is extremely difficult to establish remotely.

The Army routes its network traffic through a proxy server, which implements aggressive packet inspection. The particular configuration of the proxy server complicates programmatic access to external sources, such as the NOAA weather data FTP site or even the Python repositories, necessary for the ML application to be installed and function properly. Through initial testing, the project team found that many of the external sources do not allow a programmatic connection because of the packet inspection feature.

10.0 REFERENCES

Army Directive 2014-10, Advanced Metering of Utilities, June 2014.

Army Directive 2020-03, Installation Energy and Water Resilience Policy, March 2020.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142. http://yann.lecun.com/exdb/mnist/

DoD Utilities Meter Policy. January 2021.

Department of the Army, Office of the Assistant Secretary of the Army for Installations, Energy and Environment. February 2022. United States Army Climate Strategy. Washington, DC.

E.O. 13834 of May 17, 2018, Efficient Federal Operations. 83 FR 23771.

E.O. 14057. of Dec 8, 2021, Catalyzing Clean Energy Industries and Jobs Through Federal Sustainability. 86 FR 70935.

Energy Act of 2020, Public Law 116-260, December 2020.

Energy Policy Act of 2005, Public Law 109-58, August 2005.

Energy Independence and Security Act of 2007, Public Law 110-140, December 2007.

Ester, M., H. P. Kriegel, J. Sander, and X. Xu. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226–231.

Kim, Jeeyung & Jin, Mengtian & Homma, Youkow & Sim, Alex & Kroeger, Wilko & Wu, Kesheng. (2022). Extract Dynamic Information To Improve Time Series Modeling: a Case Study with Scientific Workflow. 10.48550/arXiv.2205.09703.

Kim W. and S. Katipamula. 2018. "A Review of Fault Detection and Diagnostics Methods for Building Systems." *Science and Technology for the Built Environment* 24(1):3–21. doi: 10.1080/23744731.2017.1318008.

Koehler T.M., J.K. Goddard, W.D. Chvala, C.J. Anderson, and X. Duan. 2017. *Implementation of the U.S. Army Reserve Enterprise Building Control System (EBCS) Initiative – Pilot Phase*. PNNL-27154, Pacific Northwest National Laboratory, Richland, WA.

Li C., Z. Ding, D. Zhao, J. Yi, and G. Zhang. 2017. "Building Energy Consumption Prediction: An Extreme Deep Learning Approach." *Energies* 10(10):1525.

Mocanu E., D.C. Mocanu, P.H. Nguyen, A. Liotta, M.E. Webber, M. Gibescu, and J.G. Slootweg. 2018. "On-line Building Energy Optimization using Deep Reinforcement Learning." *IEEE Transactions on Smart Grid* 10(4):3698–3708. doi: 10.1109/TSG.2018.2834219

Office of the Assistant Secretary of Defense, Utilities Meter Policy, January 2021

Office of the Under Secretary of Defense for Intelligence and Security. DoD Instruction 5200.48 Controlled Unclassified Information (CUI). March 2020

Parker, Steven A., Hunt, W. D., McMordie Stoughton, Kate, Boyd, Brian K., Fowler, Kimberly M., Koehler, Theresa M., Sandusky, William F., Sullivan, Greg P., and Pugh, Ray. 2015. *Metering Best Practices: A Guide to Achieving Utility Resource Efficiency, Release 3.0*. United States. https://doi.org/10.2172/1178500.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12 (2011): 2825–2830.

U.S. Army Corps of Engineers (USACE) - Engineering and Support Center, Huntsville Utility Monitoring and Control System Mandatory Center of Expertise (UMCS-MCX), Army Metering Program Guidance for Advance Meters, Revision 8, 26 May 2016

Zinkevich, M. 2019. Rules of Machine Learning. Google. Available: https://developers.google.com/machine-learning/guides/rules-of-ml.

APPENDIX A: POINTS OF CONTACT

Point of Contact	*Organization*	*Phone & E-mail*	*Role in Project*
Paul Wirt	ARIMD	703-806-6757 paul.g.wirt.civ@mail.mil	Chief, Sustainment and Resiliency Division
Emily Wendel	PNNL	206-528-3011 emily.wendel@pnnl.gov	Principal Investigator
Ben Ford	PNNL	206-528-3212 benjamin.ford@pnnl.gov	Project Manager
Bill Chvala	PNNL	509-372-4558 william.chvala@pnnl.gov	Senior Adviser

APPENDIX B: MACHINE LEARNING METHODS

This appendix presents an overview of several key methodological concepts involved in Machine Learning (ML).

B.1 Regression

Regression is an instance of supervised ML that can be used to predict a continuous value. A classic example of regression is to predict the prices of a house given the features of the house such as size, cost, and location. The prediction is performed based on training labeled data such as a database of house prices as a function of size, cost, and location. Several classification algorithms have been developed over previous years, such as linear regression, tree-based regression, support vector regression, and random forest regression. Of particular interest are more advanced algorithms that have recently gained popularity, referred to as deep learning algorithms. In this category, recurrent neural networks (RNNs) are one of the most popular deep learning regression algorithms that have been successfully applied to problems such a language modeling and prediction, speech recognition, and language translation. In the context of Figure 1 of the main report, regression is applied to the problem of predicting baseline prediction, as described in Section 2.1-546169424.388.

B.2 Classification

Classification is an instance of supervised ML in which a training set of correctly identified observations is available. Classification in particular identifies which categories (sub-populations) a new observation belongs to, based on a training set of data observations containing observations (or instances) whose category membership is known. A classic example of classification is to take an image and label it as belonging to a particular type (e.g., a dog image or a cat image) based on training that was performed on a set of pre-labeled images (training set). In the context of Figure 1 of the main report, classification is applied in two places—fault detection and fault —as described in Section 2.1. Several different classification algorithms have been developed over the last few years and successfully applied to problems such as image classification. Examples include decision trees, rule-based learning, discriminant analysis, principal component analysis, support vector machines, and multi-layer perceptron.

B.3 Recurrent Neural Networks

RNNs would be an appropriate choice for development of the baseline prediction engine in Figure 1 of the main report, because they are especially suitable to address prediction and classification problems in which the inputs are expressed as time series. In particular, they appear to be promising in the context of modeling a complex dynamic system such as a building because of their superior ability to capture nonlinear and dynamic dependencies compared to other ML methods. Their structure is such that information that belongs to a time stamp is fed back to the neural network, and thus this information is accounted for when updating the weights of the neural network (Goodfellow et al. 2016). This makes the model learn about the temporal dependence between the inputs and the outputs (see Figure B.1).

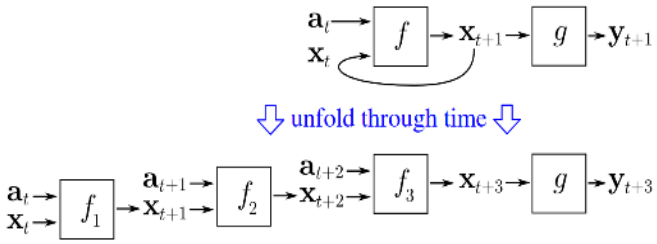

Figure B.1. Diagram of a Recurrent Neural Network

B.4 Model Predictive Control

Model predictive control (MPC) is a form of control in which control action at the current time is obtained by solving a finite time horizon, open-loop, optimal control problem, using the current state of the plant as the initial state. The optimization yields an optimal sequence of inputs and the first element in the sequence is applied to the plant while the rest are discarded. This procedure is repeated for each time instance (Figure B.2). An important advantage of this control methodology, which renders it very useful, is its ability to explicitly account for hard constraints on controls and states. Therefore, MPC has been widely applied in the petrochemical and related industries where satisfaction of constraints is very important because the most efficient operating points typically lie within or close to the intersection of such constraints. In the context of Figure 1 of the main report, MPC can be applied to the problem of optimizing control decisions, as described in Section 2.1.

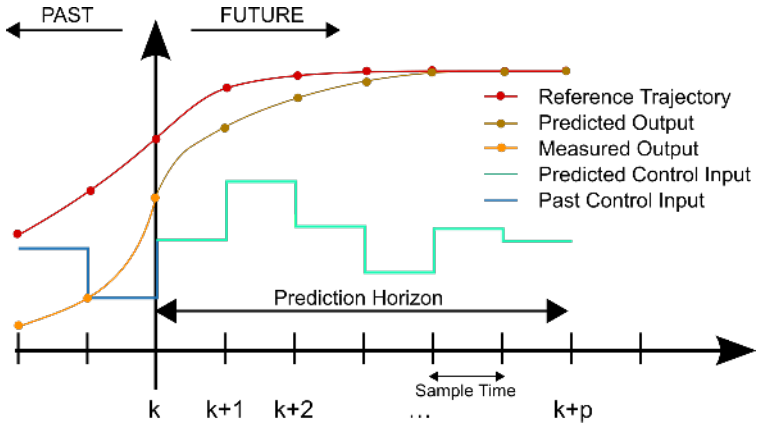

Figure B.2. General Scheme of MPC

B.5 Deep Reinforcement Learning

Reinforcement learning is a category of ML where the agent learns through trial and error. This experience-driven nature, combined with recent advancements in deep neural networks, formed a new branch in deep learning science, which has shown promising results in areas that were intractable before (Arulkumaran et al. 2017). DeepMind's AlphaGo Zero uses deep reinforcement learning (DRL) to reach superhuman performance without any human knowledge (Silver et al. 2017). In robotics (Kalashnikov et al. 2018), autonomous driving (El Sallab et al. 2017), and team-playing strategic gaming (OpenAI 2018), DRL agents and bots achieved promising results, where the environment is only partially observable, the tasks last over a long time horizon, and state and action spaces are highly dimensional. The deep neural network in

DRL is used to learn the environmental states and value of performing a set of control action at each state. Once trained, DRL could be used to suggest control actions that lead to optimal outcomes (see Figures B.3 and B.4). In the context of Figure 1 of the main report, DRL can be applied to the problem of optimizing control decisions (as an alternative to MPC), as described in Section 2.1.

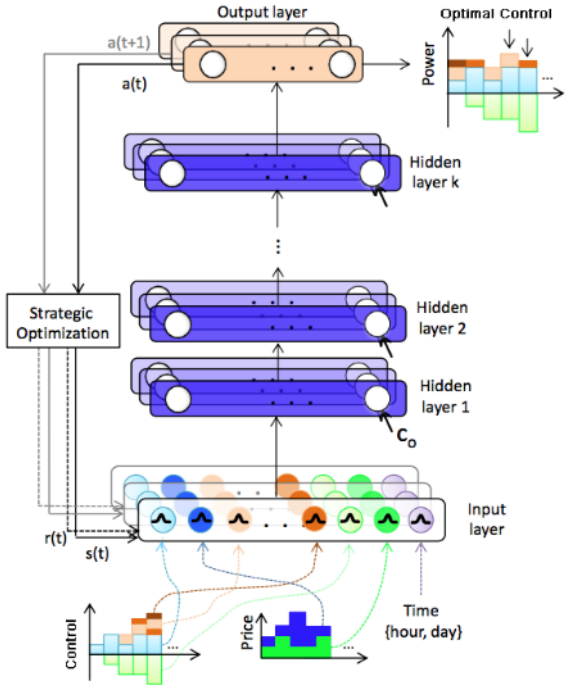

Figure B.3. General Architecture of DRL

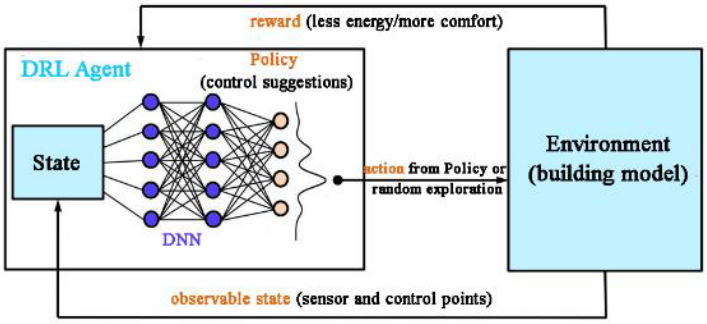

Figure B.4. Process Flow for a DRL Model

B.6 References

Arulkumaran K., M.P. Deisenroth, M. Brundage, and A.A. Bharath. 2017. "A Brief Survey of Deep Reinforcement Learning." arXiv preprint arXiv:1708.05866.

El Sallab A., M. Abdou, E. Perot, and S. Yogamani, S. April 2017. "Deep Reinforcement Learning framework for Autonomous Driving." *IS&T Electronic Imaging, Autonomous Vehicles and Machines 2017*, AVM-023, 70-76. arXiv:1704.02532.

Goodfellow I., Y. Bengio, A. Courville, and Y. Bengio. 2016. *Deep Learning* (Vol. 1). Cambridge: MIT Press.

Kalashnikov D., A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. 2018. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-based Robotic Manipulation." 2nd Conference on Robot Learning (CoRL 2018), Zurich, Switzerland. arXiv:1806.10293.

OpenAI. 2018. OpenAI Five, June 25, 2018. https://blog.openai.com/openai-five/.

Silver D., J. Schrittwieser, K. Ioannis Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. 2017. "Mastering the Game of Go without Human Knowledge." *Nature* 550:354–359.

C.1 Use Case Prioritization and Machine Learning for Buildings

In this appendix, we describe some common use cases for machine learning (ML) in the operation of commercial buildings.

1. **Baseline consumption modeling:** Predict power consumption (under business-as-usual conditions/without optimization of controls) at the building level for a specified time period in future.
2. **Unsupervised fault/anomaly detection:** Use a baseline consumption model to predict baseline consumption. Then compare the baseline prediction with real/measured consumption. If the deviation is more than a certain use-prescribed threshold, classify the performance as anomalous.
3. **Labeling consumption data as anomalous/non anomalous:** Use cluster analysis to segregate (training data only) into anomalous and non-anomalous. However this is only applied on training data for the purpose of autonomously labeling data as anomalous and non-anomalous (in the absence of expert opinion). Such labeled data are needed as an input for the use case below.
4. **Supervised fault/anomaly detection:** Use data labeled as anomalous vs. non-anomalous (either labeled by an expert) or by using the methodology in the above use case) as training data to train classification ML algorithms for fault detection.
5. **Baseline control-oriented modeling:** Train regression ML models that predict energy use and indoor environment variables (e.g., temperature). The inputs are control knobs and exogenous variables.
6. **Model-based control optimization:** Use models trained in the above use case to optimize control settings to achieve a prescribed objective (reduce energy/cost) over a prescribed period of time. Can be implemented as a lookup table.
7. **Model-free control optimization**: Train control policies directly (without the aid of a model, hence different from the model-based control optimization use case) to optimize them to achieve a prescribed objective (reduce energy/cost) over a prescribed period of time. Implementation on an actual building is outside the scope of the project; therefore, the model developed in above use case or a pre-existing simulation model can be used as a proxy to demonstrate the proof-of-concept.
8. **Transfer Learning**: Transfer models learned on one "source" building to another "similar" "target" building. Both baseline consumption model and baseline control-oriented models can be examined for transferability.
9. **Supervised Fault Diagnosi**s: Use data labeled in two layers: (1) anomalous vs. (2) non-anomalous. Anomalous data is labeled with the cause of the fault. Train classification ML algorithms on such data to diagnose (specify) the type of fault. Data can be generated from simulation models if field data required for the analysis are challenging to obtain.
10. **Predictive Maintenance**: Analyze the health of the subsystems in a building using data from equipment and estimate probability of failure.

The availability of high-resolution datasets from buildings within an enterprise provides a unique opportunity to the owners/operators of the enterprise to potentially leverage them to satisfy

several applications/use cases such as the ones described previously. However, due time and resource constraints, it is often not feasible to address all use cases at once. The owner/operator might need to down-select a few of the use cases or might want to adopt a stage gated approach where a few use cases are initially targeted, and more are brought on later, depending on resource availability.

To assist the owners/operators in making well-informed decisions about how to best allocate their resources, it is important to develop a framework that allows relative ranking or prioritization of use cases. Some of the key aspects to consider per use case are enumerated below:

1. Data needs expressed in terms of required measurements or sensor outputs, data volume and quality of data.
2. Availability of the required data.
3. Any prerequisites such as pre-trained models.
4. Candidate ML algorithms.
5. Difficulty of implementation gauged in terms of data preparation and processing effort, and ML algorithm installation and testing effort.

A comparison of the use cases across the above aspects can help in assigning a priority level per use case. The priority assignment can either be qualitative such as high/medium/low or quantitative such as a numeric score (1, 2, 3, etc.). We do not claim that the aspects listed above constitute a complete list that an organization should consider, but only represents some of the most common aspects to be considered when setting initial priorities. The priorities can be further refined based on additional criteria that might be more specific to stakeholders, such as limitations on available computing infrastructure, relative popularity of certain use cases or classes of ML algorithms (for instance deep learning might be considered more "exciting" than linear regression), "pain points" based on the experience of operating the buildings so far, etc. The use case prioritization matrix for the EBCS and MDMS datasets using this framework is shown in Tables C.1 and C.2.

Table C.1. Data Needs for ML Use Cases

Use Case Description	Use Case Category	Data Needs		
		Measurements	Data Volume	Data Quality
Baseline consumption modeling	Fault detection, energy bench-marking	Energy use data (hourly or smaller resolution): Gas consumption Electricity consumption Water consumption Outdoor environmental Data (hourly or smaller resolution)	At least 1 year. Multiple years is preferred	Should be fault-free. Sparse gaps are tolerable (a few data points missing).
Unsupervised fault /anomaly detection	Fault detection	M-2A: All measurements needed for baseline consumption modeling M-2B: Consumption measurements for time period	At least 1 year. Multiple years is preferred	Data M-2A should be fault-free. Data M-2B can have faults. Sparse gaps are

Use Case Description	Use Case Category	Data Needs		
		Measurements	Data Volume	Data Quality
		for which fault detection needs to be performed (can be a real-time stream).		tolerable (a few data points missing).
Labeling consumption data as anomalous/ non-anomalous	Fault detection	Energy use data (hourly or smaller resolution): Gas consumption Electricity consumption Water consumption	At least 1 year. Multiple years is preferred	Data should contain faults but not all of the data should be faulty. Sparse gaps are tolerable (a few data points missing).
Supervised fault/anomaly detection	Fault detection	Energy use data and outdoor environmental time-series data; these measurements should be flagged as faulty/non-faulty (either by data owner or using the labeling technique above)	At least 1 year. Multiple years is preferred	Data should contain faults but not all of the data should be faulty. Sparse gaps are tolerable (a few data points missing).
Baseline control-oriented modeling	Control optimization	Control knobs: set-point temperatures etc. Exogenous variables: outdoor environmental data, occupant data or suitable proxies. Performance data: energy use data, indoor environmental data	At least 1 year. Multiple years is preferred	Should be fault-free. Sparse gaps are tolerable (a few data points missing).
Model-based control optimization	Control optimization	Exogenous variables in baseline control-oriented modeling, for time-window for which control optimization needs to be performed. Utility rates for time-window for which control optimization needs to be performed	At least 24 hours, multiple days spanning multiple seasons	Should not have any gaps.
Model-free control optimization	Control optimization	Exogenous variables in baseline control-oriented modeling, for time-window for which control optimization needs to be performed. Utility rates for time-window for which control optimization needs to be performed	At least 1 year. Multiple years is preferred	Should not have any gaps.
Transfer learning	Energy bench-marking, Control optimization	Source building: same measurements in the baseline consumption modeling (if transferring baseline consumption model) or as in	At least 1 year. Multiple years is preferred	Should be fault-free. Sparse gaps are tolerable (a few data points missing).

Use Case Description	Use Case Category	Data Needs		
		Measurements	Data Volume	Data Quality
		baseline control-oriented modeling (if transferring control-oriented model). Target building: same measurements as for source building but for a smaller time period (required for validating the effectiveness of the transfer).		
Supervised fault diagnosis	Fault diagnosis	Energy use data (hourly or smaller resolution): gas, electricity, water consumption; outdoor environmental data (hourly or smaller resolution); indoor environmental data; equipment operational data. Fault labels: for each time stamp, information on whether there was a fault and if so what was the fault	At least 1 year. Multiple years is preferred	Data should contain faults but not all of the data should be faulty.
Predictive maintenance	Predictive O&M	Equipment operational data which also contains data corresponding to equipment breakdown	Multiple years of data, preferably containing equipment lifetime	Sparse gaps are okay.

Table C.2. Part B of the Use Case Prioritization Framework: ML Implementation

Use Case Description	Prerequisites	Candidate ML Algorithms	Difficulty of Implementation	
			Data Preparation and Processing	ML Implementation and Testing
Baseline consumption modeling	None	Start with: linear regression, linear SVR, random forest; if needed: RNN	Easy: Normalization	Easy: linear regression, SVR, random forest can be implemented using Scikit-Learn; RNN: can be implemented using Keras/TensorFlow.
Unsupervised Fault /anomaly detection	Trained baseline consumption model; threshold for flagging data as anomalous	Algorithms used for training baseline consumption models	Same as for baseline consumption modeling	Same as for baseline consumption modeling.
Labeling consumption data as Anomalous/non-anomalous	None	Clustering algorithms: k Mean-s/hierarchical	Easy: normalization	Easy: Scikit-learn
Supervised Fault- /anomaly detection	None	Binary classification techniques: logistic regression, linear SVM, stochastic gradient descent	Difficult: Involves obtaining labeled fault data	Easy: Scikit-learn
Baseline control- oriented modeling	None	Time-series regression approaches: LSTM, LSTM-CNN combined architectures, Linear regression, SVR, random forests.	Obtaining occupant data might be difficult, proxies might be needed	Moderate: TensorFlow/Keras; hyperparameter optimization will be needed.
Model-based control optimization	Baseline control-oriented model developed; Comfort constraints specified	Model predictive control (would need black box optimization solvers); implement them as lookup tables.	Easy	Difficult: black box optimization solvers have computational complexity issues and can have poor convergence behaviors. Trial and error with different types of solvers and different parameters settings will be needed.
	A simulation	Reinforcement	Easy	Moderate; can be

Use Case Description	Prerequisites	Candidate ML Algorithms	Difficulty of Implementation	
			Data Preparation and Processing	ML Implementation and Testing
Model-free control optimization	model of the building (can be the same model as in baseline control-oriented modeling) or access to the actual building control knobs	learning		implemented in OpenAIGym.
Transfer Learning	None	Direct transfer; off-line schemes such as inductive transfer, on-line techniques such as Generalized Online Transfer Learning (GOTL).	Easy	Moderate to difficult depending on off-line vs. online transfer learning.
Supervised Fault Diagnosis	None	Classification techniques: linear SVM, stochastic gradient descent, random forest classification	Difficult: involves obtaining labeled fault data	Easy: Scikit-learn
Predictive Maintenance	None	Stochastic techniques, e.g., Hidden Markov Models	Difficult: involves equipment life cycle data	Easy: Scikit-learn

C.2 DOD Best Practice for Applying ML to Building Systems

Modern buildings produce a constant stream of data from their building automation systems (BASs) and advanced metering infrastructure (AMI). The U.S. Department of Defense (DoD) has tens of thousands of buildings generating an ever-growing amount of data. This torrent of information quickly becomes too overwhelming for manual interpretation and evaluation. ML offers one data-driven approach for producing meaningful, actionable insights from the data.

Given enough data, ML can be an extremely powerful tool that can save building operators and managers valuable time and energy. Some examples of how ML models could be used in improving building operations include:

- Identifying sites that are consuming more energy than their peers after accounting for differences in budling size, climate, usage type, and other unique factors.
- Predicting the amount of energy a building will consume in the future.
- Alerting operators to operational issues, even before an alarm is generated.

However, ML is not magic and not able to solve every problem. This document provides a brief overview of ML and provides some basic considerations before undertaking an ML project.

Machine Learning Primer

ML is often thought to be a complex and intimidating topic. While it certainly can be, especially in cutting-edge applications, it also can be approachable for people of all skill levels and backgrounds. Linear regression is the simplest form of ML. If you have every used Microsoft Excel to fit a trendline to data on a graph, you have done some ML! Even complex models and applications use the same basic approach and principles that Excel uses when calculating the linear regression trendline. At the most basic level, ML simply fits a model to the available data with the least amount of error possible.

ML is a diverse field with a variety of common applications and many algorithms to accomplish different goals. Only a portion of the algorithms, or model types, are applicable to facility operation and optimization. Appendix B: Machine Learning Methods contains a brief overview of some of those ML methods.

Considerations for Machine Learning Projects

There are many public resources for ML best practices. For example, Wujek et al. (2016) provides a good overview of general best practices in any ML project and Google offers a number of ML resources, including a guide featuring 43 rules of ML projects (Zinkevich 2022).

Training an ML model is the easiest part of an ML project! Most of the work in a successful project is spent on gathering sufficient data and creating the infrastructure to ingest the input data and serve the model predictions. The following sections offer some additional considerations.

Problem Definition and Use Case Selection

The first step in any project is defining the problem that you are attempting to solve. Are you attempting to predict a continuous value, classify a sample into one of a few categories, or group

similar samples together? Each of these problems has different models that work best for that use case.

The final demonstration report (Ford et al 2022) provides an overview of the common use cases for ML in the operation of commercial buildings. It also provides a Use Case Prioritization Matrix (UCPM), which can be used to set expectations for what ML use cases are and are not possible given the available data. While ML can be incredibly powerful, it is not a perfect fit for every application. Depending on the project constraints, there could be a better option than a ML model.

Data Availability and Preparation

Data are the cornerstone on which all ML projects are built. Without good input data, the model predictions will be useless. In practice, getting good input data is the most difficult part of any ML project and is the main determining factor for use case possibility. The UCPM describes the minimum data requirements needed for the common ML use cases for buildings.

At a minimum, input data sources must be aligned with the intended ML project objectives; that is, data structure and content need to be adequate for the needs of the model. For example, monthly energy use data would be unusable by a model designed to predict hourly energy use.

Even with good alignment of input data sources and ML project objectives, data almost always require at least some preparation prior to being fed into an ML model. Aside from simply having access to the required data, care must be taken to assure the data are properly cleaned and in a suitable format. This is usually a non-trivial effort, and can even involve a separate data-cleaning ML model.

Model Selection, Training, and Evaluation

While there is no single "correct" model for a given project, some model types perform better than others. The UCPM provides some guidance on what type of model to use for different use cases, but in general, the only way to determine which model performs best on the available data is to test different models! In the DoD ESTCP ML Demonstration project, we found that random forest regressors performed the best for predicting the future energy consumption of a building. Use of the appropriate evaluation metrics calculated on "held-out" data is best way to determine which model will work best for a specific project.

Going Further

This short document does not even begin to scratch the surface of ML. A great next step would be to read the full ESTCP final report (Ford et al 2022), which goes into much greater detail about the accomplishments and challenges of a real-world ML project using the U.S. Army Reserve's (USAR's) Enterprise Building Control System (EBCS) in conjunction with the Army Meter Program's Meter Data Management System (MDMS).

References

Ford, Ben, E. Wendel, T. Yoder, V. Chandan. 2022. *Final Report: Optimizing Facility Operations by Applying Machine Learning to the Army Reserve*. EW19-5300, Environmental Security Technology Certification Program, Washington, DC.

Wujek, Brett, P. Hall, and F. Günes. 2016. “Best Practices for Machine Learning Applications.” SAS2360-2016, SAS Institute Inc., Cary, NC.

Zinkevich, M. 2019. Rules of Machine Learning. Google. Available: https://developers.google.com/machine-learning/guides/rules-of-ml.