

Artificial Intelligence and Cyber Intelligence

An Implementation Guide

Artificial Intelligence and Cyber Intelligence: An Implementation Guide

Authors

April Galyardt
Ritwik Gupta
Dan DeCapria
Eliezer Kanal
Jared Ettinger

Design

David Biber
Alexandrea Van Deusen
Todd Loizes

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-0447

Table of Contents

Introduction.....	1
Conditions for Success in ML	2
Use Cases for ML in Cyber Intelligence	5
The ML Pipeline	7
Decision Maker Reporting and Feedback	17
Organizational Structure for ML	18

Machine Learning and Cyber Intelligence

INTRODUCTION

Machine learning (ML) tools and techniques have been demonstrated to effectively improve cyber intelligence workflows across environment, data gathering, threat analysis, strategic analysis, and feedback to decision makers. Simply adding ML capability to existing organizational toolsets, procedures, and workflows will not solve all cyber intelligence challenges. These technologies work in concert with experienced and qualified personnel who know how to understand, integrate, and even improve ML processes in the context of cyber intelligence challenges. Only by combining modern tooling with personnel knowledgeable about its use and procedures can organizations begin to realize the significant benefits ML can provide.

KEY JUDGMENTS

- Setting up an effective ML–cyber intelligence collaboration will require proper consideration, preparation, and communication.
- Introducing operationally effective ML into the cyber intelligence workflow requires a repeatable, consistent, and well-defined process.
- Prior to using ML, it is essential to walk through the ML checklist to answer relevant questions such as “Does ML help with this?” and “Have we considered the broader context?” Any doubts that arise when completing this checklist highlight gaps in analytical understanding that must be discussed with the cyber intelligence team.
- There are important ethical and data-use dilemmas associated with ML, especially when paired with the world of intelligence. Enumerate, weigh, and address these dilemmas to the fullest extent possible before proceeding with ML capabilities.
- In ML, the biggest performance improvements result from higher-quality data, not more sophisticated algorithms. Expect to spend the majority of your time and effort on data acquisition, curation, and processing.
- A large variety of people are needed to make an effective ML and cyber intelligence effort; namely, talented cyber intelligence analysts, ML scientists, and ML and/or data engineers. While expertise may initially be divided into silos in each member’s domain, the team must work together to nurture domain expertise in a cross-functional manner and maintain open lines of communication.
- Be creative and have fun with the data sources you have. The reason to use ML is to tap the hidden knowledge potential within those data sources, so think critically about what new things can be extracted from the data that already exists and how to use it effectively. Don’t forget to balance this creativity with operational and engineering considerations.

DEFINITIONS

ML is a field at the intersection of statistics and computer science. Fundamentally, it is about learning from data: summarizing patterns, making predictions, and identifying key characteristics of a group of interest (among many other tasks). The term **artificial intelligence (AI)** has many definitions, but broadly speaking it refers to systems that understand the world and independently make smart decisions based on that understanding.¹ If an AI system can interact with and learn from interactions with the surrounding world, it must be learning from data. To that extent, ML is an integral part AI. Unfortunately, the language around AI and ML is further muddied by the fact that some ML algorithms, particularly neural networks, are often referred to as “AI” by the general public. In this guide, we focus on ML and the practice of learning from data.

Within ML, a **model** refers to a set of equations that could describe the data. When we **train** or **fit** a model, we search over a family of models to find the single model that best fits the data. This **trained model** is often referred to as simply **the model**. Within this context, an **algorithm** is a specific process for fitting the model to data. **Features** or **variables** refer to the different kinds of information recorded in the data; for example, if our data is a set of documents, one feature might be the author’s name and another might be the number of words in the document.

The work of designing a model, fitting a model, and extracting information is generally performed by an **analyst**. However, within a cyber intelligence framework, we must disambiguate this work from the work of a cyber intelligence analyst.

We use **ML scientist** to refer to people who carry out the ML analysis and **data engineer** to refer to people who collect and prepare the data for analysis.

CONDITIONS FOR SUCCESS IN ML

ML is a powerful tool, and it has spurred tremendous leaps in capability and productivity across many areas. However, just as hammers work well with nails but poorly with screws, ML is ideally suited to some tasks and poorly suited to others. This section is designed to help identify problems for which setting up an ML pipeline would justify the investment.

In popular conceptualizations, ML focuses on algorithmic capabilities; for instance, recommender systems in shopping carts (“You may like

ARE YOU READY TO USE ML FOR CYBER INTELLIGENCE?

1. Can you state your problem as either:
 - a. I would like to use ___ data to predict __, or
 - b. I would like to understand the structure of the features recorded in ___ data?
2. Is it a large-scale problem?
3. Have you already done exploratory analysis?
4. Have you considered the broader context?

¹ <https://ai.cs.cmu.edu/about>

this”) or automated labeling of people, places, or objects in images. However, less-visible aspects of ML are just as critical to the process as algorithm choice: data storage, processing, and cleaning; the augmentation of data (“feature engineering”); reporting and visualization; and the development of software and hardware to support the entire ML pipeline (among others). Many organizations already perform these tasks to support other business needs, and ML is often added to existing analysis pipelines to take advantage of existing tools that perform some of these duties.

To help determine whether your organization is ML ready, we’ve developed a checklist of necessary capabilities. If you aren’t yet performing these data analytic practices, we recommend that you incorporate these items into your analytics pipeline and familiarize yourself with their output and value before adding ML capabilities.

PROBLEM STRUCTURE

ML algorithms can be broadly divided into two categories: supervised learning algorithms and unsupervised learning algorithms. A supervised learning problem can be framed as “I would like to use some set of data to predict an unknown.” For example, you might want to identify the actors responsible for constructing and deploying a particular piece of malware. Statistically, this is a prediction problem and can be reframed as “I would like to use hashes of malware files to predict who made the malware.” As another example, cyber intelligence analysts often have more information coming in than they are able to process. The supervised learning problem can be framed as “We might want to use data about what information cyber intelligence analysts have previously found most useful to predict which new information is most important for the analysts to consume.” These two examples will use vastly different data, but because they can both be framed as a problem of making a prediction using data, supervised learning algorithms can be applied to these problems.

Unsupervised learning problems can be framed as “I would like to understand the structure of the features recorded in a given set of data.” The structure we’re looking for could be very different depending on context. One common kind of structure we might look for comprises subgroups and clusters; for example, we might analyze resolved incident tickets collected over the past year by looking for clusters of related tickets. A second kind of structure comprises groups of closely related features. For example, if we are collecting data on insider threat indicators, we might want to examine which features are highly correlated with each other. If we identify 10 to 12 features that are all closely related and effectively measuring the same thing, then we may be able to reduce our data collection burden and only collect the 5 or 6 most useful. Note that in the section below, “Examples of ML for Cyber Intel,” we refer to these two problem structures so you can see how they are applied in practice. Later, in the section “The ML Pipeline,” we discuss the requirements and conditions under which you can apply both supervised and unsupervised learning.

SCALE

ML will frequently reap the largest return on investment when it is applied to a task at a large scale. This scale could take many different forms. A common problem of scale is a task that needs to be executed repetitively. For example, satellites collect images faster than humans can label them. However, an ML algorithm can label the petabytes of images collected each day and flag anomalous images for human review. In other situations, an analysis might only be needed once, but the data available is of large scale—more than can be handled by a single person. For example, analysts might have a large amount of information on a particular set of network intrusions. In such a case, ML algorithms could find patterns in the data. This information then increases the capability and speed of the human cyber analyst.

The second way ML can address a problem of scale is to provide greater consistency across repetitions. For example, if we are looking at data and deciding whether or not to open an insider threat investigation, two humans might reasonably disagree. Supplementing the human analyst with information from an ML algorithm can foster greater consistency in decision making.

EXPLORATORY ANALYSIS

A good rule of thumb is to always run simple analyses first. This includes basic information about data, such as how much data is missing, lists of the most frequently observed data for different datatypes, and data visualizations, such as histograms and time series charts. Statisticians frequently refer to this as exploratory data analysis. You should be able to answer basic questions about your data, such as “How many?”, “How often?”, and “What kind?” before attempting to apply ML techniques.

There are two reasons for addressing these questions. First, you can gain tremendous insights from your answers. How many of your incident tickets contain the words “technical debt?” How many contain the word “malware?” Simply identifying the “top 10” lists for different types of data frequently uncovers significant trends you may not be aware of. This type of analysis is very straightforward to perform using data analysis tools, and taking advantage of this low-hanging fruit can be a very cost-effective way to make use of existing data.

The second reason for addressing these questions is that the data cannot be put into an ML algorithm until it has already been sufficiently processed. The ability to answer these basic questions indicates that the data is processed enough for use in an ML algorithm. Furthermore, often what you find when conducting the simple exploratory analysis provides insight that will help shape the ML analysis. For example, you might discover that one sensor is, essentially, replicating the information from another sensor. Therefore, only one of those sensors should be used by the ML algorithm. Another common discovery is that, from date X to date Y, a sensor was misfiring and therefore should be omitted for those dates. When you try to apply an ML algorithm without first acquiring this basic understanding of the data, errors will happen.

BROADER CONTEXT

Every ML analysis takes place within a broader context that includes ethical and legal ramifications. These issues will vary with context, but there are two that every ML analysis will share in common and which should be addressed when deciding whether to implement an ML algorithm: 1) What are the consequences of a data breach? and 2) What are the consequences of an incorrect decision based on the ML algorithm?

The large amounts of data required to make ML efficient also make data breaches more problematic. The Pennsylvania Supreme Court recently ruled that businesses can be held legally responsible for not taking adequate safeguards with sensitive information.² We also know that many algorithms, particularly neural networks, “leak” information: if someone has access to any of the decisions or output of an algorithm, they can make strong inferences about the information that was used to train the algorithm. The consequences of such a breach vary from case to case, ranging from an increased risk of identity theft for consumers to national security issues.

ML models are probability based. There is always a level of irreducible error in the output of an ML algorithm. For instance, if we are looking at predicting insider threat, there will always be cases in which the algorithm indicates someone is a threat (but they are not) and cases in which someone is a threat (but the algorithm misses it). This is true in every single application. As you implement an ML model, you must develop the procedures and responses to situate the output within your organization. Is this a case in which the response to the ML output can be automated? Or, do you have a case in which the response must be escalated to a human decision maker?

USE CASES FOR ML IN CYBER INTELLIGENCE

There are many different types of ML, each best suited to solving a particular set of challenges. To provide a better understanding of how these tools can augment your cyber intelligence analysis, the following section describes a number of use cases demonstrating these capabilities in a variety of common scenarios.

ENVIRONMENTAL CONTEXT: INSIDER THREAT ANALYSIS

One increasingly common application of ML is to predict which individuals within an organization might represent insider threats. This use case is usually a supervised learning problem: Collect as much relevant behavioral computer activity as possible on users (web browsing, network share access, logon/logoff logs) and use this data to predict the extent to which an individual is, or has the potential to be, an insider threat. This problem could also be framed as an unsupervised learning problem of anomaly detection: Most employees will exhibit relatively consistent usage patterns (e.g., logging on at a consistent time of day). A starkly anomalous usage pattern may be a red flag. The statistical problem is then to identify the anomalies in the data.

² <https://www.jdsupra.com/legalnews/pennsylvania-supreme-court-recognizes-34420/>

There are two points to highlight in this use case. First, this application of ML would require proactive coordination between the insider threat team, the cyber intelligence team, and the ML team. Second, models and data collection should be updated regularly: Attack surfaces change constantly, and gaps in coverage are continually relevant.

DATA GATHERING: IDENTIFYING REDUNDANT INFORMATION

When dealing with large, disparate datasets, there is frequently significant redundancy in the available data. Since this redundancy can substantially increase analyst workload, especially as the scale of the data increases, ML can help identify which information is redundant to save time and storage requirements. Unsupervised learning methods (including clustering and the feature reduction algorithms discussed below), combined with simple comparison metrics, can be used to group the data and flag redundancy. On large datasets, this can result in a significant reduction in data the analyst needs to examine and in greatly reduced data storage needs.

THREAT ANALYSIS

Malware Attribution

Given a set of executable files that are known to be malware, we may be interested in identifying the sources of the malware to identify a threat actor. If we have access to a labeled dataset in which different pieces of previously collected malware have been tagged with their source, we can use that data to build a supervised learning model to predict which of our new malware files has come from each source.

Sorting and Prioritizing Information for Cyber Intelligence Analysts

Not all information has equal importance or priority when running through a cyber intelligence pipeline. Under normal circumstances, it is only after an intelligence analyst has reviewed the information that it can be given a priority rating; however, ML methods may be able to predict these priority ratings, and the historical relevance of similar data, from the task at hand. Assuming that we have access to past data that has already been given priority ratings, we can use supervised learning methods to sort incoming data by priority using a trained model.

STRATEGIC ANALYSIS: IDENTIFYING COMMONALITIES IN ATTACKS

Strategic analysis is the work of conducting holistic analysis on both threats and opportunities. Holistic assessment of threats is based on analysis of threat actor potential, organizational exposure, and the impact the threat has on an organization. One might also perform strategic analysis to provide deep clarity on the who and why behind threats and threat actors. Strategic analysis goes beyond threat analysis to incorporate analysis of emerging technologies and geopolitics that may impact and/or provide opportunities for the organization now and in the future.

When reviewing data for strategic analysis, analysts often search for associations between actors, events, or activities. We can rephrase this as an unsupervised learning question: “Given a dataset consisting of resolved incident tickets from the past year, can we find clusters of tickets that relate to similar threats or threat actors?” Identifying commonalities would ease the discovery of a modus operandi or positive threat actor identification.

Clustering analysis could be augmented using active learning techniques, which help analysts identify where more data is required to make proper decisions. For example, consider an analyst attempting to identify the threat actor or actors behind a series of seemingly unrelated, discrete threats. In the course of her analysis, she classifies different threats into discrete buckets. This allows her to apply a clustering technique, which can automatically label new data as it comes in by comparing it to existing buckets. Furthermore, by applying active learning to her data, she can understand where her own model is strongest and weakest. When new data comes in and is automatically labeled, she can quickly know the extent to which the new label requires further manual analysis.

REPORTING: VISUALIZATIONS AND AUTOMATIC REPORT GENERATION

Pairing ML with other automation provides an additional advantage. When an ML analysis is repeated regularly, the reports and graphs based on that analysis can be automatically updated and sent to cyber intelligence analysts or leadership for human review. There are several commercially available tools that streamline automatic report generation even further.

THE ML PIPELINE

The fuel powering the entire ML process is data. This data must be processed, cleaned, and prepared before being put through the algorithm. The output of the algorithm is some useful result that the analyst can use. However, just as a car isn't very useful without seats, the ML process requires significant external tooling to make the whole thing useful.

In this section, we outline the main steps for performing analysis:

5. requirements definition
6. data input and processing
7. model design, development, and execution
8. reporting and feedback

The overall model is visible in Figure 1. Note that we do not show intelligence requirements (IR) in the figure since they are too high-level for the purposes of ML.

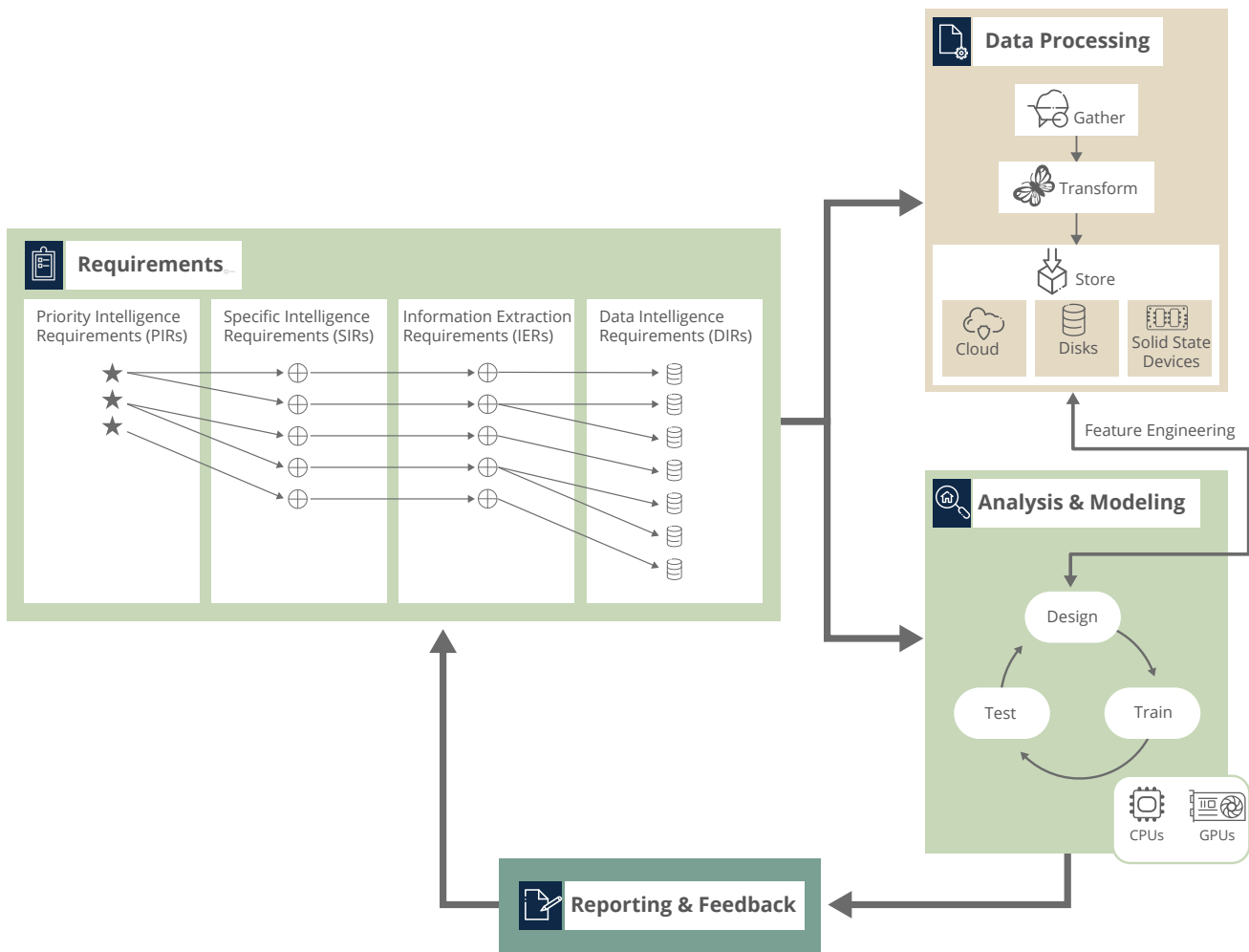


Figure 1. The machine learning pipeline for strategic analysis.

DEFINING REQUIREMENTS

Consistent with current cyber intelligence best practices, an ML planning process should begin with specific intelligence requirements (SIR) that map to priority intelligence requirements (PIR) from your collection management team. These cyber intelligence requirements are your north star. Without an intelligence requirement stated in a clear problem definition, analysts can veer into questions such as “What will happen if I try this fancy new algorithm on this data?” While that kind of exploration can be useful for getting to know the data and trying out whether a new algorithm works as advertised, it probably won’t solve the big picture problems for the organization.

A common complaint from many organizations is, “I have more data than I know what to do with.” This happens because data was collected without any particular use in mind. Consequently, the organization often does not record information that would be useful to answer any particular question. Not all data is equal, and for most companies enormous stores of inert data provide no incremental value. The planning stage is critical to avoiding this problem.

We suggest creating explicit information extraction requirements (IERs) and data intelligence requirements (DIRs) that map directly to the organization’s intelligence priorities held by the collection management team. These requirements will help guide the ML analysis and data collection and provide an explicit mechanism for tracking how data is used. The IERs and DIRs need to be developed in tandem, because the kinds of ML analysis desired will have different data needs and the data available will shape what ML analyses are possible. The IERs need to answer questions such as

- What kind of data science method should we be using?
- What metrics will we use for success?
- Are there any other criteria necessary to make the results useful?

The IERs should be developed by the ML scientist and the cyber intelligence analysts. The consumers of the ML output are there to ensure that the ML scientists understand their needs. The ML scientist must be there to translate those needs into properties of the analysis. When an ML scientist first meets with a client, she will listen and ask a series of questions in order to understand the client’s needs. The IERs simply make these needs explicit and directly link them to the organizations’ PIRs and SIRs.

To develop the IER, it’s helpful to begin by asking questions such as

- What does a minimally sufficient solution look like?
- How good does an ML model have to be to be useful?

The answers very much depend on the use context for a problem. If all conclusions must be strong enough to present as evidence in a court of law, that is a very different threshold than one needed to simply ask analysts to investigate a suspicious anomaly. When we are deciding what criteria our model should optimize and what thresholds it must meet, we must take this context into account.

BEST PRACTICES

SETTING THRESHOLDS

While a 5% error rate has been a standard threshold in statistics and data science for about a century, it may be a bad threshold for your application. A popular social media platform recently deployed an ML algorithm to detect adult content published on its platform. This is a prediction algorithm designed to detect whether a given image contains particular types of nudity. Like any prediction algorithm, it will make mistakes. However, for this large platform, a 5% false positive rate constitutes approximately 500 million misclassified images and a corresponding number of users unhappy that their images are being blocked for no apparent reason. The default 5% error rate is not low enough for this context. It is worth spending the time and effort to determine what kind of criteria must be met for a solution to be useful for your problem.

DIRs follow from IERs. Once IRs, PIRs, SIRs, and IERs are specified, DIRs address concerns such as

- What data do I need to fulfill these IERs?
- How much data do I need?
- Are there any data collection methodologies that need to be followed?
- What are potential sources of the data that is needed for analysis?

Different analyses require different amounts of data. Simple analyses might require hundreds of data points, while more complex analyses, such as neural networks, require tens of thousands. The DIRs make these kinds of requirements explicit. In the section “Modeling” below, we discuss when labeled data is necessary and why it is imperative that the collection conditions match exactly the conditions in which an ML model will be used. The DIRs specify these requirements.

Defining the collection conditions is an essential part of a documented and repeatable process of requirements generation. It is also important to frequently verify DIRs. As the nature of any cyber intelligence analysis changes, the types, amount, and sources of data also change. Consequently, ensure that DIRs are not carried over from previous tasks simply because previous tasks seem similar.

One last point regarding IERs and DIRs: They should all be designed with specific expiration dates. The questions of interest to an organization will necessarily change as the cyber threat landscape changes, and an organization should adapt its analysis to meet the new questions. Design the requirements with this in mind.

Gap Analysis

Gap analysis can be useful in defining requirements. This is a formalized process for answering

- Where are we now?
- Where do we want to be?
- How do we close the gap between here and there?”

Through this gap analysis process, you may discover that the data you currently collect does not actually contain the necessary information to address your problem.

For example

- Where are we now?
 - ♦ We analyze netflow data to determine whether our network is under attack.
- Where do we want to be?
 - ♦ We want to learn the identity of our attacker.
- Identify the gap.
 - ♦ Netflow data is too coarse to identify how the attack is being executed.
- How are we going to close the gap?
 - ♦ We need to collect data directly on the computer being attacked and, possibly, examine logs and memory dumps.

This netflow example highlights an important lesson: Just because you have data, that doesn't mean it has the right information to answer your current question. Netflow data is perfectly adequate to provide evidence that an attack is occurring. But it's completely insufficient to answer how the attackers got in, how they are moving through your network, or what they're doing inside. Gap analysis is a useful tool for identifying requirements in general, including which new data needs to be collected.

QUESTIONS YOU SHOULD BE ABLE TO ANSWER AT THE PLANNING STAGE

- What is the relevant IR/PIR/SIR?
- What does a “minimally sufficient solution” look like?
- Create information extraction requirements (IERs).
 - ♦ What kind of analysis is required (e.g., classification, anomaly detection, clustering...)?
 - ♦ What metrics will be used to measure success?
 - ♦ Are there other criteria that must be met (e.g., specific run time, processing limitations)?

Note: You should not settle on any particular algorithm at this stage, simply identify the needed metrics and criteria
- Create data intelligence requirements (DIRs).
 - ♦ What data do I need to answer this question?
 - ♦ How much of that data do I need?
 - ♦ Are there any specific collection requirements (e.g., random sample)?
 - ♦ You might have a couple of data sources in mind, but it's too early to commit to a specific one.

DATA PROCESSING

Data gathering and data processing are where you should expect to spend the majority of your time and effort. It should be noted that, within the field of ML, the biggest improvements in performance come from a foundation of better, higher-quality data.³ This is true in cyber intelligence as well. For example, in one study the authors tried seven different algorithms to predict, from three sets of features, whether or not a file was malware. The differences among the algorithms' performances was minimal. The differences among the prediction accuracy for different features were pronounced (Table 1). This example highlights two things that are almost always true about ML: 1) Better data generally makes more of a difference than algorithm choice, and 2) There is generally a different cost associated with different features.

Feature of Executable File	Ease of Extraction	Prediction Accuracy
n-grams of bytes	Cheap and easy to extract	60-80%
Opcodes	Requires disassembling the file, medium cost and effort	85-95%
API calls used by executable	High effort and computational time to extract	90-95%

Table 1. Prediction accuracy in malware detection algorithm study

3 <https://arxiv.org/pdf/1808.01201.pdf>

The majority of the work in ML takes place in the data preparation stage. Some estimates suggest you'll need as many as five data engineers for each ML scientist. In our experience with customers, this is not unrealistic; the amount of work required to prepare data for ML should not be underestimated.

One of the best practices we observed was to automate data gathering and processing as much as possible. Anything you do more than once should be automated. In fact, one team noted that achieving any sort of scale without automation is impossible. Having an automated data collection and processing system, on the other hand, allows teams to get more and different data, allows for continuous improvement, and results in direct savings in labor costs. Furthermore, the automation provides time for the analysts to work on more pressing issues.

GATHERING DATA

First and foremost, identify data sources in your DIRs that meet your needs rather than collecting and storing data using an ad-hoc approach. We also recommend tracking the sources and making that information part of how you store the data. When gathering data, it is already common practice among database experts to create an extensive data dictionary describing what each data element is and how it was generated. However, in the context of ML, consider adding the source of each data element to the dictionary. Doing so will not only help you track where data elements come from, reducing technical debt later on as models are updated, but also allow you to assess the usefulness of each data source in the future. This assessment can help you decide whether or not to continue to collect particular data. In addition, an ML algorithm can be adapted to weight each source based on prior knowledge about source quality or the algorithm's assessment of the data's value in making predictions.

It should also be noted that manual data entry is highly error prone, expensive, and often infeasible at scale. Avoid manual entry as much as possible. We reiterate that automation in all aspects of data gathering should be pursued to the extent possible.

DATA TRANSFORMATION

Once data sources have been obtained, the data must be prepared for storage and subsequent analysis. This typically entails at least three steps: cleaning, storage, and feature engineering. In this section, we will discuss the first two steps. Feature engineering will be discussed in greater detail at the end of the section "Modeling."

Data cleaning entails ensuring the data is of proper quality for future analysis. This work includes tasks such as handling missing data, outliers, correct-but-highly-unlikely values, and similar problems. While each of these tasks can be handled in any number of ways, the key here is consistency. Mishandling these data corner cases can result in the loss of significant data, incorrect conclusions, or missing entire swaths of data for a variety of reasons. Given that data is the fuel that powers all analysis, the intelligence analyst, the ML specialists, and any subject matter experts should all agree on how such corner cases are handled.

Over the past two decades, the technology for storing data has become incredibly sophisticated: It is now possible to store almost any type of data using commercial-off-the-shelf products. Unfortunately, this can make life difficult for the ML practitioner, because it's now often far easier to simply toss

data into a database than it is to prepare it for subsequent analysis. Consequently, storage should be performed with the end goal in mind; in this case, subsequent usage in an ML algorithm. So, when considering a data storage solutions such as relational databases, time-series databases, NoSQL engines, binary blobs, graph entities, and document-based storage, it is critical to consider how the stored data will be consumed. It may be, for instance, that the data should be stored in multiple formats—columnar tables as well as JavaScript Object Notation (JSON)—to enable different types of analytics. Because the nature of the analysis will vary tremendously based on the use case, analysts should ensure that the data storage format is working to their advantage rather than posing a hindrance.

MODELING

In this section, we will discuss the two broad categories of ML problems (supervised and unsupervised learning) with the goal of enabling leadership and cyber intelligence analysts to work effectively with their ML team. The type of modeling you need is determined by your PIR and/or SIR. It should be specified in the IER, which you established based on your PIR. Categorizing an ML problem this way can help constrain it, and it can help you begin the process of translating an organizational need into a tractable data science problem.

We emphasize that basic exploratory descriptive statistics should always be completed before beginning a more complex ML analysis. Descriptive summaries are some of the most basic tools, but also the most useful. Such summaries include visualizations, averages, standard deviation, correlations, and proportions. Every project will need them, and many problems can be solved with these tools alone. They answer the question, “What does normal look like in this dataset?” For example, knowing how much traffic your network usually carries on Monday morning as opposed to Thursday afternoon is very useful in planning infrastructure and scheduling system updates. This information could also be used to provide a baseline for an ML algorithm against which network traffic anomalies could be detected. Moreover, creating a simple visualization is often the fastest way to reveal errors in data collection; for example, a negative number denoting the number of logons on a particular day would be a clear indicator that the data collection process needs to be checked. Descriptive summaries provide a necessary foundation for interpreting the results of more complicated analyses.



PREDICTION AND SUPERVISED ML

A supervised learning problem can be framed as “I would like to use _____ data to predict _____.” The key requirement for supervised learning is labeled data. For instance, if you want to use the hash of a binary executable file to predict who made the malware, then you need to have the hashes of other binary files that have already been labeled by source. Likewise, if you want to predict which new pieces of information are most important for human cyber intelligence analysts, then you must have data on prior information that the analysts have labeled as useful or not useful.

This labeled data will serve as the “ground truth” for training the ML algorithm.⁴ The quality of labels applied to the training data will directly impact the quality of the ML output.⁵ For example, let’s assume a case in which labels from cyber intelligence analysts on different pieces of information indicate whether the information was useful or not. It is easy to imagine different analysts disagreeing about how to label the same piece of information. Moreover, certain pieces of information might meet some, but not all, of the criteria for earning the label “useful.” How should these data points be labeled? All of these small decisions will impact the utility of the results from any ML algorithm applied to the data. Indeed, when we fit a supervised algorithm, the final model is the one that makes the best predictions on a test set of the labeled data, so it is imperative that the labels be meaningful and accurate.

The second consideration in selecting a supervised learning algorithm is to identify what kinds of information you need to be able to get out of your analysis. Do you need to predict whether or not a new source, piece of data, or method of analysis will be useful? Or do you need to infer what features of the information indicate whether it will be useful? The first case could be valuable if you just need to help your intelligence analysts figure out what needs their attention most. The second case is more important if you are evaluating which information services you want to continue to collect. Many methods can be used for either prediction or inference (e.g., logistic regression), but some methods can only be used for prediction (e.g., k-nearest neighbors).

Supervised learning models work under specific assumptions and constraints. A supervised learning model is trained on one set of labeled data and then applied to new data. For this to be successful, the assumptions and conditions that held true during training must also be enforced during deployment. Consider again the malware example: There are large open source repositories of malware files that could be used to train an algorithm. But these repositories have been highly curated and contain files that are easily and distinctively identified as malware. Malware “in the wild” will not be so easy to identify. If you train your algorithm to identify malware from one of the curated repositories, it will only catch the “easy-to-identify” cases.

Moreover, the order for a supervised learning process is always

1. Collect the labeled data.
2. Train a model.
3. Apply the model.

4 Ground truth is simply defined as “the correct answer.” Labeled data provides the correct answers for every input.

5 <https://ssrn.com/abstract=2477899> or <http://dx.doi.org/10.2139/ssrn.2477899>

Therefore, the labeled data is always data from the past that we will leverage for the present. When we apply a supervised learning model, we are making a bet that tomorrow will be the same as yesterday. This is often a bet we are willing to make. For example, using the pixels of an image to predict whether or not there is a face in that image is not a task that will evolve rapidly over time. However, tomorrow's malware attack probably won't look very much like one in the past. Immutability over time is simply one example of the general immutability constraint. There can be disastrous results if the application conditions do not sufficiently match the conditions for the collection of training data.

SUMMARIZING STRUCTURE AND UNSUPERVISED ML

Unsupervised learning methods answer the question, "I would like to understand the structure of the features recorded in ____ data." The most important consideration for using unsupervised ML is "What kind of structure are you looking for?" Different algorithms will find different kinds of structure. The three most common types of structures we might be interested in are clusters, anomalies, and sets of highly correlated variables. Other kinds of structures include lower-dimensional representations, density estimates, and latent variable representations. The IERs based on the PIRs and SIRs should specify the kinds of structures we are interested in for a particular analysis. The reason you must pre-specify the structure you are interested in is twofold. First, any patterns discovered in the data have to be functionally meaningful to the intelligence analyst. Second, many of these methods will impose structure on the data, even when that structure might not actually be present; if you pre-specify the patterns of interest, it is easier to evaluate whether or not the patterns you find are actually real.

In a clustering problem, the question is "Are there meaningful subgroups in this data?" For example, we might be interested in identifying users with similar patterns of usage (e.g., early risers, night owls, and system administrators). We could be looking for groups of incident tickets that all have the same attack patterns. Common methods for extracting clusters include k-means, mixture modeling, and distance-based hierarchical clustering.

Anomaly detection tries to answer the question "Is there anything in this data that doesn't look similar to the rest?" For example, if our computer is infected with malware, it might be sending or receiving unusually large amounts of data; an anomaly detection algorithm might be useful for implementing a system to detect an infection.

Assume the question "I have a whole pile of variables—are any of them related to each other?" For this case, simple dimension reduction methods that look for sets of highly correlated variables are appropriate. For example, if you are looking at usage statistics on a website, then number of clicks and total time spent on the page are going to be highly related: The more time someone spends on the page, the more links they're likely to click. In general, dimension reduction techniques (called factor analysis in some communities of practice), focus on finding a simpler representation of the data that contains the same information as the original data. One of the most popular techniques for dimension reduction is principal components analysis (PCA), which uses basic linear algebra to find groups of features that are linearly correlated. However, PCA is most appropriate for numerical data. Other methods, such as word2vec or latent Dirichlet allocation, are more appropriate for textual data.

Measuring the success of a supervised learning method is fairly easy: How well do my predictions match the truth? In comparison, measuring the success of an unsupervised learning method is much trickier. If you ask a clustering algorithm to find five clusters in the data, it will find five clusters, but they may not be meaningful (see Figure 2). One of your criteria for success is that the patterns discovered have to be functionally meaningful to an intelligence analyst; this criterion could be measured informally or with a survey, depending on how big the team is.

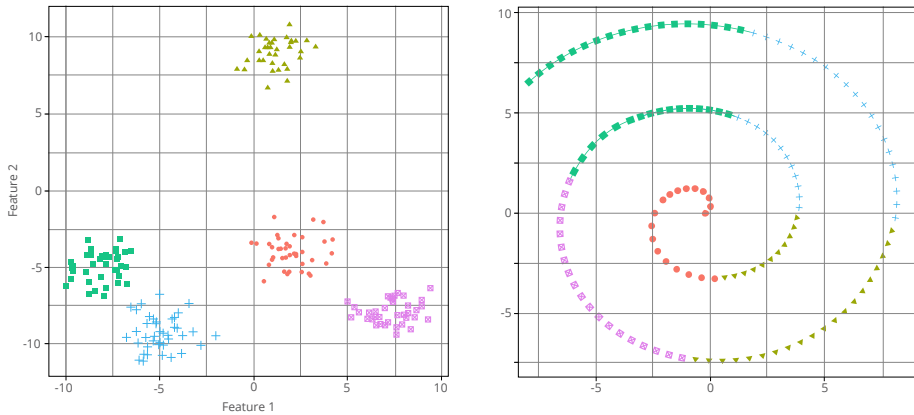


Figure 2. When clusters are present (left), a clustering algorithm will find clusters. When a different structure is present, (the spiral is a type of lower dimensional manifold), a clustering algorithm will still try to find clusters.

FEATURE ENGINEERING

Feature engineering is an integral part of the feedback between data processing and modeling. If we can refine and improve the features we use to train a model, then we can achieve increased operational effectiveness through greater model accuracy or reduced training time. Feature engineering can take two forms. The first is feature selection, which is used to mitigate redundant features (information is already contained in another feature, duplication) and irrelevant features (features contain no lift to an applicable ML task at hand). The second, feature extraction, is used in creating new features by combining original features under a consistent methodology. The PCA technique mentioned above is often used for automated feature extraction.

It is worth emphasizing that manual feature engineering requires more thought and effort but often produces greater rewards. Through close collaboration between the ML team and the cyber intelligence team, it is common to discover that, for example, “We’ve been using the total count of event X, but we get better results if we use the time that has passed since the last event X.” As discussed earlier, better data often provides more improvement than better algorithms. Similarly, leveraging the domain expertise of the cyber intelligence team in feature extraction will produce better ML results.

ADVERSARIAL ML

Adversarial ML is not a substantial threat yet, but it will be in the near future. Adversarial ML is still largely the domain of the research community, which is working to document the different ways in which ML systems might be vulnerable; methods for hardening AI systems against adversarial attacks are cutting edge research. The methods of attack have not permeated the script kiddie community yet; however, adversarial examples are already abundant. Most of these examples follow a pattern in which an organization trains and deploys an ML system, then the adversary builds an attack that deceives it. However, they are nonetheless powerful for demonstrating the dangers: a 3D printed turtle⁶ classified as a gun and a pair of glasses⁷ that causes facial recognition software to believe you are John Malkovich. Another attack type we can expect in the medium-term future is the injection, by an adversary, of malicious examples into your training data.⁸

DECISION MAKER REPORTING AND FEEDBACK

No matter the ML technique or the area of application, the results must be understandable to the end users of the output (the “consumer”). Even more importantly, these ML consumers must have a mechanism for providing actionable feedback to developers and analysts to ensure that the analysis is not only understandable but also valuable. The most effective way to ensure this conversation can happen is by defining a common language. Given the expected widespread adoption of ML solutions, developers and analysts cannot assume that all consumers will be literate in ML techniques, lingo, and nuances. However, a certain amount of literacy is required to ensure that useful feedback can be provided. Literacy in the following specific concepts should be enforced:

- **The concept of “probabilistic answers” is common in ML.** Many ML algorithms do not answer questions with a “yes” or “no” but rather with a likelihood that a given scenario has occurred. For example, consider an algorithm observing a large amount of network traffic coming from a known bad set of IP addresses. The algorithm may be intelligent enough to recognize the activity as a component of an attack, but may not have enough additional data to further classify the attack, or it may not be trained to recognize this specific type of activity as an attack, or any of a number of similar possibilities. In this scenario, algorithm output may indicate that an attack is occurring with 34% likelihood. While this is not something a person would say, it represents how algorithms process input.

Additionally, these outputs could be mapped to ICD 203⁹ expressions of likelihood. ICD 203 §D.6.e.2.a describes colloquial terminology ranging from “almost no chance” to “almost certainly,” mapping likelihoods to seven possible categories. It is critical for executives to understand that these terms are not chosen arbitrarily but correspond to specific likelihoods provided by the algorithm.

- **Algorithm performance depends on two factors: the training set and the currently available data.** Continuing our previous example, assume that our analyst recognizes that this type of traffic

6 <https://www.theverge.com/2017/11/2/16597276/google-ai-image-attacks-adversarial-turtle-rifle-3d-printed>

7 <https://qz.com/823820/carnegie-mellon-made-a-special-pair-of-glasses-that-lets-you-steal-a-digital-identity/>

8 <https://www.cs.umd.edu/~tomg/projects/poison/>

9 <https://www.dni.gov/files/documents/ICD/ICD%20203%20Analytic%20Standards.pdf>

is perfectly normal. Does this recognition mean that the algorithm is junk and should therefore be ignored? Of course not! The algorithm only knows what it was shown in the past and what it has available to it at the moment. Unfortunately, it is all too common for the consumer to use this data point to dismiss the AI as “junk.” Misbehavior in one scenario does not imply misbehavior elsewhere. Users of AI systems should find out what types of data the AI is best equipped to handle and be extra cautious about trusting output when feeding the system data outside its expertise. Similarly, when dealing with the AI’s “specialty” data, pay close attention to the output before dismissing a seemingly spurious result: The AI may see a pattern or trend that a human would normally miss.

Note also that some AI systems possess the ability to continuously learn new information. For example, modern spam filters are “preprogrammed” to identify generic spam. As users tag the spam they personally receive, the system learns new types of spam and classifier performance increases. Some cyber intelligence systems possess a similar capability; if this is the case, consumers should be aware that their labels are being included in the system.

- **Appropriate trust is key.** When it comes to AI, trusting the output too much or too little can be problematic. In the earliest uses of AI in aviation, there were crashes because pilots did not trust the AI system. In contrast, we know that there are systematic biases in which AI results deserve less trust. Trusting a system too much may be particularly problematic if an adversary figures out how to craft an attack specifically targeted to avoid detection by the AI—even if the human would have identified the attack without AI assistance, overreliance on an AI system may lead analysts to trust output without validating it. “Trust but verify” is a healthy motto.

Once consumers understand and internalize these concepts, they must then understand how to convey feedback to analysts. “This doesn’t make sense” is almost never considered useful feedback. Rather, we recommend that consumers try to make their feedback more actionable, focusing on the levers that analysts can tweak. The following examples demonstrate various types of actionable feedback. In all cases, “you” refers to the consumer.

ORGANIZATIONAL STRUCTURE FOR ML

There are many concerns related to creating an ML activity within your cyber intelligence organization. It is difficult to understand the team composition, how to collaborate with an ML activity, and how to best support the ML activity with proper policies and infrastructures. In this section, we outline how to organize your cyber intelligence team to achieve success with ML, and we also look at how you can incorporate some classic software engineering principles to ML to ensure high-quality output.

ORGANIZING AN ML EFFORT WITHIN AN INTEL TEAM

An understanding of the relationships among IRs, PIRs, and SIRs over time, region, and industry is maintained through individual roles and responsibilities. A team that can function effectively at the intersection of cyber, intelligence, and ML must include people with specific backgrounds, skillsets, and traits. Moreover, the team members must have a clear separation of roles and responsibilities while at the same time allowing close collaboration and effective information sharing.

A successful ML–cyber intelligence effort requires three parts:

- **domain expertise:** knowledge of cyber intelligence and other organizational context
- **ML expertise:** understanding of the underlying theory in ML and how to apply it
- **data engineering expertise:** ability to engineer systems that integrate and scale ML and cyber intelligence capabilities

Without these three kinds of expertise, an ML effort within a cyber intelligence team will find it difficult to succeed and scale.

ML scientists, cyber intelligence analysts, and data engineers must all have depth in their respective domains, but they must also be able to understand and, most importantly, communicate across their domains. These are three very large bodies of knowledge, so it is rare to be able to hire an individual with expertise in more than one of these areas. However, within a cross-domain cyber intelligence team, individual members will usually have a primary area of specialty and will develop expertise in a secondary area as they work in the intersection. This can be facilitated with formal training and collaboration sessions, but is often achieved informally via day-to-day interaction and collaboration.

Close collaboration and open communication are critical at all times and can likely be better facilitated within a fusion center, where diverse teams come together to analyze disparate information. There are complex design requirements that exist in each domain of work that require each practitioner to take the restrictions and needs of another domain into consideration at every step of their work. For example, there are fundamental limitations to what a compute resource can accomplish. The cyber intelligence analysts and ML scientists must listen and make adjustments to ensure their solutions do not grossly overestimate the fundamental assumptions that a software engineer is taking for granted. The ML and engineering personnel need to be on, or work closely with, the cyber intelligence team. Since their jobs involve directly modeling and analyzing data collected and tagged by cyber intelligence analysts, it is essential for them to be included in regular information sharing and planning meetings, especially regarding information collection practices or procedures.

SOFTWARE ENGINEERING FOR ML

While ML processes can be used to effectively monitor, assess, and model environmental events, they are not without their operational concerns. Under the best circumstances, an ML system designed today will need to be monitored and adjusted as time passes.

Stakeholders must understand, at least on a high level, that software engineering for ML looks different from software engineering elsewhere in their organization, because following good software engineering practices will enable you to adapt to changing circumstances. Traditional software engineering attributes, such as functionality, usability, reliability, performance, and supportability (FURPS), still apply to the world of ML. However, their specific implementations will look different for the ML pipeline.

Specifically, ML systems require much more verification at each step of the process than the surrounding software pipeline. The following list breaks this idea down into the specific FURPS components:

- **Functionality:** As the model is training and constantly updating, and as it is being exposed to more data, does it still achieve the task it is being created to solve? The passage of time and the application of training updates does not guarantee functionality. Consequently, verification must be woven into the functionality pipeline.
- **Usability:** Is the model consistent? Does it produce outputs humans can understand and reason about? Does the consistency change as the model gets new training updates?
- **Reliability:** Does the model provide stable outputs? If the model predicts an input as class 1 with high confidence at time t , will it still predict class 1 within the same confidence at time $t+1$? This must be verified regularly and not taken for granted.
- **Performance:** Does training the model more and more increase the runtime requirements for inference? Does the model become too large to feasibly deploy to production systems? Constant performance checks must be in place.
- **Supportability:** Is it simple to influence the behavior of the model early in the training process? How about late in the training process? How can you update a production model and ensure its veracity? A verification pipeline must be constructed here as well.

There are two places in which the differences between software engineering for ML and traditional software engineering are particularly stark. First, the mental representations a software engineer uses to think about data are fundamentally different from the mental representations an ML scientist uses to think about data and modeling. These differences can cause communication difficulties between the software engineering team and the ML team, so it is critical to have open lines of communication to resolve these issues.


Second, ML creates a tight coupling between the content of the data, the model, and the final use of the information. In fact, the model is created specifically to connect the data to the end use. This is in stark contrast to traditional software engineering, in which tight coupling is forbidden. However, the format of the data storage, the hardware, and the implementation of the algorithm can and should all be loosely coupled. Once again, close collaboration between the ML team and the software engineering team is required to address these issues.

INFRASTRUCTURE AND HARDWARE

A strong computing infrastructure is necessary to maintain a healthy and capable data science effort. Namely, the availability of computing infrastructure with networked storage and compute (CPU and GPU) capabilities is essential. Many companies will already have infrastructure for this purpose elsewhere in the organization, so partnering with those teams and growing their infrastructure is a strong possibility.

Along with computing infrastructure, quick and reliable data storage is a necessity. Data is processed repeatedly in ML R&D and production; strong data storage capabilities will augment the volume and speed of the ML effort.

With large data environments, there need to be strong measures in place to protect the transfer, usage, and availability of data. All ML and cyber intelligence personnel must follow rules set by the CISO organization. These measures also include data governance and compliance, which can apply to both on-premise and cloud infrastructures. Due to the tight data coupling in the ML domain, guidance must be created in conjunction with the CISO organization to avoid any potential issues.



About the SEI

The Software Engineering Institute is a federally funded research and development center (FFRDC) that works with defense and government organizations, industry, and academia to advance the state of the art in software engineering and cybersecurity to benefit public interest. Part of Carnegie Mellon University, the SEI is a national resource in pioneering emerging technologies, cybersecurity, software acquisition, and software lifecycle assurance.

Contact Us

CARNEGIE MELLON UNIVERSITY
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE; PITTSBURGH, PA 15213-2612

sei.cmu.edu
412.268.5800 | 888.201.4479
info@sei.cmu.edu