Achieve Production Optimization with AWS Machine Learning

October 2020

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

During the last decade, artificial intelligence (AI) and machine learning (ML) have been introduced into the manufacturing sector, enabling efficiency in processes, driving sustainability, supporting safer working environments, and increasing quality and productivity. AI and ML–enabled manufacturing can offer innovation and optimizations by providing manufacturing systems fault detection and prediction, optimal use of raw materials, parts and resources, exploiting heterogeneous big data analysis and an interconnected manufacturing enterprise. This paper provides an introduction of how machine learning can be used to optimize OEE through the identification, prediction, and prevention of unplanned downtime, fewer quality issues, and improved productivity.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems on AWS. Using the Framework allows you to learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

Introduction

Within the manufacturing industry, manufacturers are constantly looking to optimize production within factories to improve efficiency and reduce cost. In today's digital world, the AWS Cloud, with its nearly unlimited storage and compute consumed with a pay-as-you-use model, has provided a platform for manufacturers to innovate and trial ideas that were only historically available to academic or research organizations. This has opened new opportunities in the constant drive for optimization in real-time with cloud-enabled services, such as the Industrial Internet of Things (IIoT), data lakes, analytics, and machine learning (ML).

For manufacturers that are executing their digital transformation and integrating operational technologies (OT) and information technology (IT), the application of machine learning is a relatively new concept. This whitepaper provides an introduction to machine learning and how it can be used in manufacturing by providing example use cases of optimizing production through the variable of Overall Equipment Effectiveness (OEE).

Overall Equipment Effectiveness

Across discrete and process manufacturing, one of the main metrics used to gauge manufacturing productivity is Overall Equipment Effectiveness (OEE). There are a multitude of books and websites that explain and give examples of OEE but in summary OEE is a percentage that represents the productivity of a plant, line or manufacturing machine. The OEE percentage is made up of the percentages of quality, throughput or performance, and the availability or non-stoppage time as shown in the following formula:

$$OEE = Availability \ x \ Performance \ x \ Quality$$

With the detailed calculation being:

$$OEE = \frac{(Planned\ Prod.Time - Stoppage\ Time)}{Planned\ Production\ Time} \ x \ \frac{(Cycle\ Time\ x\ Cycles)}{Run\ Time} \ x \ \frac{Good\ Count}{Total\ Count}$$

By focusing on the factors that influence the variables of availability, performance, and quality, we can improve OEE. It is generally accepted that OEE greater than 85% is considered world class, with most manufacturers operating in the 60% range.[1]

ML and OEE

If we could measure and reduce downtime (improve availability), reduce defects (improve quality), and improve the productivity of a factory, line, or machine, then we can increase OEE. To accomplish this, you need process and production data to measure and analyze the manufacturing processes so that insights can be drawn for improvements. Manufacturing operational data sources can be factory machine telemetry data, operational MES (Manufacturing Execution System) data, maintenance records from asset management systems and production planning data from the ERP (Enterprise Resource Planning) system as examples or even manually collected data in logbooks. This is where the concept of a manufacturing data lakes comes in as it provides a cloud-based platform which scales and enables the central storage and analysis of these data sets. By having the manufacturing data in a data lake allows the data scientist to correlate, analyze and train the machine learning models for anomaly detection and predictions which can then be used to monitor and control the three OEE variables.

The application of these machine learning models into the manufacturing process where processes can be automated is where the value comes, and this is generally referred to as the artificial intelligence (AI) piece of the process. This is not an overnight exercise but a gradual journey of learning which will enable manufacturers to optimize their physical manufacturing process through the predictions of the machine learning models. For some manufacturers the ultimate goal is to be a ‘lights out’ manufacturing enterprise using AI and machine learning predictions to automate decisions or augment manual process steps and upon gaining confidence on the ML model, the models can directly invoke control points changes in the manufacturing process.

Manufacturing use cases

There are many use cases for the application of AI and ML in manufacturing, from autonomous robots for material movement to digital twins, but for this whitepaper we focus on the cases that have a direct correlation to OEE improvements. These use cases can be grouped into two main categories: operational planning and manufacturing operations.

Operational planning

For operational planning use cases, the application of AI and ML is around simulation and forecasting for manufacturing processes and demand planning. By using historical data and external data sets as inputs the use cases are:

- Process simulation, determining the required capacity to improve cycle times and reduced production line interruptions
- Operational forecasting, forward planning on labor and material to support the next X months of manufacturing
- The application of ML in supply chain is justifiable as a topic on its own and therefore this paper won't focus on it in regards to OEE improvements. But as a side note, supply chain as an input can have a variance on production planning and material supply and therefore OEE.

Manufacturing operations

When it comes to manufacturing operations use cases for AI and ML the application is on the factory floor and using machine telemetry, process sensors and augmented object data like images to:

- Improving machine availability by:
    - Reducing manual machine intervention.
    - Implementing prescriptive maintenance.
- Improving machine performance by:
    - Providing automatic setting/recipe management control.
    - Optimizing processes through simulations.
    - Fine tuning variables to optimize machine speeds, process parameters, energy use as examples.
    - Identification of shortstops and microstoppages and eliminating them
- Improving output quality by:
    - Detecting potential defects through process measurement monitoring.
    - Automated visual inspection.
    - Anomaly detection and defect analysis of inline test.

Analytics and data

Before we jump into the depths of machine learning and the application of it to improve OEE, let's talk about data and analytics. If you have already started experimenting with machine learning, or if you are about to start, you should start with analytics and data. To train a machine learning model to make a prediction with a high level of accuracy, you need a training data set. This data set is built using raw data from different systems and sources that has been cleansed, analyzed, and contextualized before it can be used. The phrase "you need to learn to walk before you can run" applies heavily. Before ML is applied to the data, invest time and effort to understand the business context and confirm that the correct data is available and understood through analysis. However, don't strive for perfect data before starting any sort of analysis and ML—your data sets will grow and mature overtime.

Analytics

Let's work backwards and start with analytics. Data is the key component and it's important that we dive a little deeper into it. There are four different types of analytics: descriptive, diagnostic, predictive, and prescriptive.

Descriptive

Descriptive analytics provide the answer to "what happened?" and provides insights into the past through data analysis to determine what might happen in the future.

Diagnostic

Diagnostic analytics is the next layer and provides the answer to "why did it happen?" This is where analysis of data is used to identify the root cause and understand cause and effect.

Predictive

Predictive analytics answer the question "what will happen?" based on the use of statistical analysis and modeling of historical data, in addition to more advanced machine learning algorithms.

Prescriptive

Prescriptive analytics answer the question "what should I do?" and uses simulation algorithms on the data to answer questions around possible outcomes or events.

Prescriptive analytics is one method that can be used for decision making for artificial intelligence.

In some use cases, the application of predictive analytics might be a simpler, quicker, and lower cost option to using machine learning. With any problem, half the solution is to find the right tool to solve it, and machine learning should be seen as one of many tools, not the only tool in the tool box.

Data

Without data, we would not be able to implement machine learning. In fact, you generally need large amounts of data to be able to apply machine learning. The data set needs to contain enough data for the algorithm to detect the decision-making logic in multiple scenarios. If this decision-making logic is easy and the scenarios are constrained, then less data is required. But if the decision is very difficult to make or there are many scenarios, then substantial amounts of data might be required. For example, a defect detection model requires less data to detect the difference between a scratch and a dent but needs more data to detect the difference between a scratch and a scrape. In addition, the model might require less data if the images are well lit and stable, but more data if the lighting is inconsistent or blurry.

Two other factors are also important in determining if the model can detect the decision-making logic from the dataset: data quality and data context. First, high quality data is necessary, otherwise machine learning algorithms will mistakenly learn to mimic and then produce inconsistent data results. This results in weak performance at the least and harmful outputs at the worst. As such, the importance of a high quality “golden dataset” that is manually verified by humans cannot be understated.

Second, data capturing the context of the problem is important. Data scientists often work very closely with SMEs (subject matter experts) to make sure the contextual data is also provided to the model. For example, to perform automated quality control, it is not sufficient to only know the final quality label. Relevant supporting evidence, such as color, weight, dimensions, hardness, and other testing parameters, also need to be provided to the data, which normally comes from the manufacturing engineers or quality inspectors.

To achieve higher data quality and more data context, it’s recommended that you invest the time and effort upfront in establishing a manufacturing data lake platform with data security and lifecycle management strategy and policies. A cloud-based data lake provides a central platform to bring together siloed data sources, for example, ingesting industrial IoT data from factory automation equipment (PLCs, DCS, and Historians), and

ETL (Extract Transform Load) data from ERP systems. The data is stored in the relevant data stores, such as a time series database, object store, or structure SQL database, as shown in Figure 1 with the AWS Manufacturing Data Lake Reference Architecture as an example. In addition to providing a platform to store data, a data lake also provides the services to transform and processing data so it can be filtered, aggregated, labelled, and transformed into useable data for the data scientist to use as a data set.

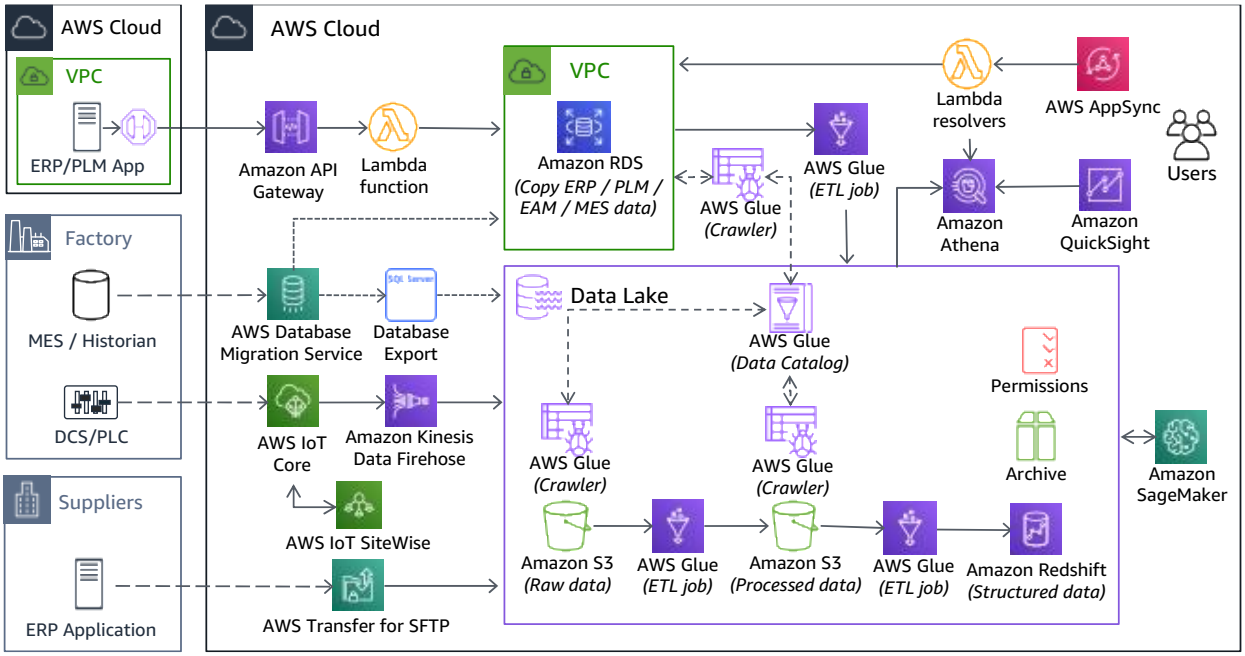

Figure 1. Manufacturing Data Lake Reference Architecture

Introduction to AI & ML

Machine learning is not new, it has been around since the 1950's and most of the common machine learning techniques we use today were invented years ago. What has changed is the barrier of entry has been significantly lowered due to the resources available in the cloud and the pay as you use model, which allows experimentation and innovation with minimal cost and risk. Machine learning is no longer constrained to only research or academic institutions, individuals and business of all sizes can experiment and apply machine learning.

Artificial Intelligence

Artificial Intelligence (AI) is the general array of research and applications that enables computers to demonstrate human-level intelligence. From simple if-then statements to expert systems to today's ML algorithms, the goal of these techniques is to emulate human intelligence to achieve increasing degrees of accuracy for wider breadths of applications. However, in the context of this whitepaper, we focus our discussion specifically to "Narrow AI", or the ability of computers to accomplish straight-forward, well-defined tasks, such as defect classification or anomaly detection. It is within this focus where AI applications can demonstrate immediate impact.

Machine Learning

Machine learning is a subset of AI that uses machines (computers) to search for patterns in data to build logic models automatically. Within machine learning, there are different algorithmic methods which can be applied to the data sets for specific applications such as forecasting, anomaly detection, or classification. The appendix provides a deeper dive into the different ML algorithms for reference.

Deep learning

Deep learning is a form of machine learning which was initially used in the application of computer vision and natural language process. It is based on artificial neural networks with feature learning which is used in the detection or classification from the data. Deep learning has since shown good applications beyond perception tasks, such as for forecasting, recommendation systems, and graph/network analysis.

Machine Learning on AWS

At AWS, we have machine learning services at three different layers in the service stack (Figure 2) which can be used and applied depending on the level of machine learning practitioner's expertise or use case complexity.

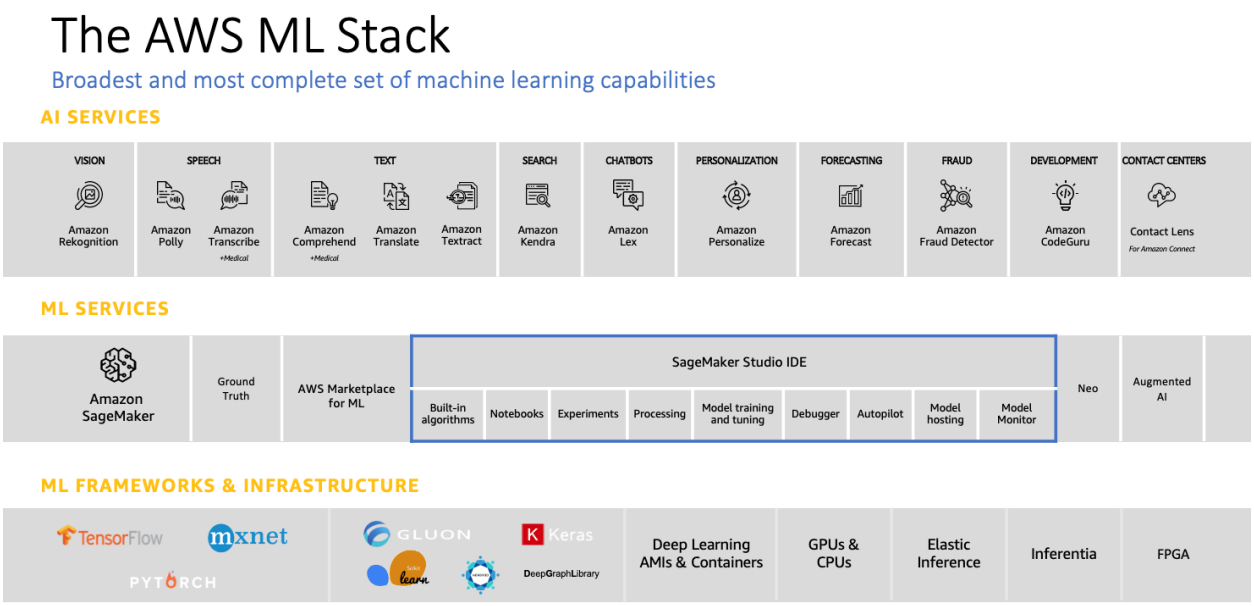

Figure 2. The AWS ML Stack

ML frameworks and infrastructure services

The services in the bottom layer of the stack are generally used by expert ML practitioners. These are people who are comfortable designing their own tools and workflows to build, train, tune, and deploy models. They're comfortable operating at the framework and infrastructure level.

ML services

The middle layer of the stack is for ML developers and data scientists where we've simplified the workflow to make it easier for them to build, train and deploy models. For example, Amazon SageMaker is a managed ML service that makes it easier to build, train, tune, and deploy ML models. SageMaker has algorithms and frameworks built-in to enable the data scientist to focus on training the model rather than managing the underlying infrastructure. SageMaker has native integrations with PyTorch, MXNet, and TensorFlow, but enables the use of any framework and algorithm.

In addition to SageMaker Studio being an IDE (Integrated Development Environment) for machine learning, Amazon SageMaker Ground Truth can be used to automatically label your data or to facilitate the use of human labelers in building your training data set.

AI services

The top layer is a set of AI services that allows the effective application of ML in everyday activities with natural language, vision, and text. For example, for object detection in images, use Amazon Rekognition and for natural language processing, use Amazon Comprehend.

The Machine Learning process

When it comes to applying machine learning to address manufacturing problems, such as increasing OEE by reducing defects or downtime as mentioned in the previous section, it is important to have cleansed data with samples in it before starting the process of training a model. The training and evaluating of the model's predictions is an iterative process as shown in Figure 3.

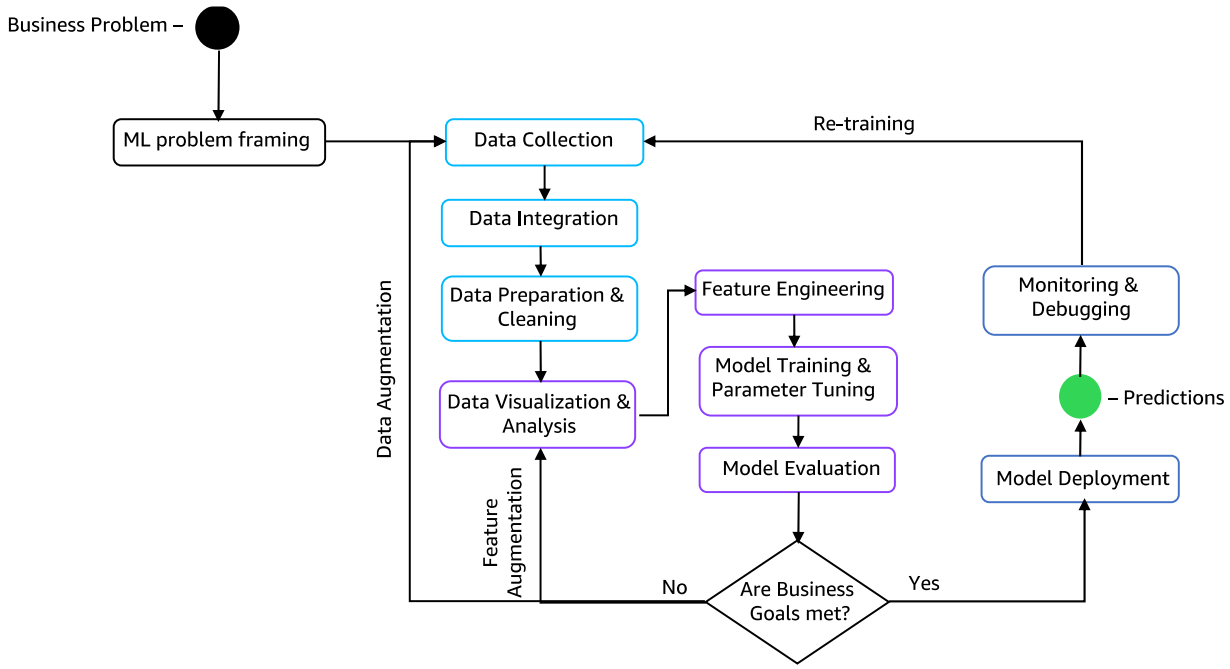

Figure 3. Machine Learning Process

It is also important to ensure the business goal and expectation is set for the model's performance, such as accuracy. A model accuracy of 99% is unlikely to be achieved from the start. That being said, accuracy is generally not the best metric in manufacturing use cases and we turn to *precision* and *recall*. This is because accuracy

is not useful unless the number of predicted classes are similar, which is not the case in manufacturing since a line is expected to produce drastically more passing parts than failing parts. The precision and recall metrics originally come from document search and are easiest to explain in this context. First is precision, which is measured by *search results that are relevant* divided by *all search results*. Precision can be thought of as a "cleanliness" metric. Second is recall, which is measured by (search results that are relevant) divided by (all relevant documents). Recall can be thought of as a "completeness" metric.

As an example for predictive maintenance: Let's say there are 15 machines that failed in the past. A predictive maintenance algorithm is being back tested (that is, tested on past and historical records) that predicted 10 of these machines would fail but only 5 actually failed. In this case, the precision of the predictive maintenance model would only be 5/10 and the recall of the predictive maintenance model would be 5/15.

However, maximum precision and maximum recall is difficult to achieve simultaneously. In the context of predictive maintenance, to catch more failures before they occur, some working machines will also be flagged as failing, thus lowering precision. In other words, higher recall means that more machines would receive unnecessary maintenance, while higher precision means some machines would fail before maintenance was performed. There is not a one size fits all metric, and selecting the right metric to optimize the model's performance depends on the business problem and context.

Continuing with the previous example, a cost analysis would need to be made to balance the cost of performing maintenance vs letting the machine fail. If the maintenance effort is expensive, then a higher precision in the predictive maintenance model is recommended, since unnecessary maintenance should be minimized. On the other hand, if the machine in question is a critical piece of equipment and the maintenance is cheap, then a higher recall in the model is recommended, since an actual failure would be detrimental.

Application of ML in manufacturing

When it comes to applying machine learning to your manufacturing use cases to improve OEE, there are many ways to address the business problem and many algorithms to apply. This section breaks down the OEE metric and looks at how ML is currently used by manufacturers to improve availability, performance, and quality.

Improved availability

Recall that the OEE calculation for availability is:

$$Availability = \frac{(Planned\ Prod.Time - Stoppage\ Time)}{Planned\ Production\ Time}$$

To maximize the availability metric, you must either decrease *Stoppage Time* or be precise in predicting *Planned Production Time*. In this section, we cover both predictions to decrease stoppage time and forecasting to improve production time planning.

Decreasing stoppage time with predictions

Stoppage time within the OEE calculation can be planned (machine setup time) and unplanned (either a machine or asset failure, or a failure during the manufacturing operation with the part which requires resolution or a reset) as an example. In this section, we focus on the unplanned stoppage time, as this is often the biggest variable and where machine learning can be used to provide prediction. As a working example, we walk through predictive maintenance as a mechanism to prevent unplanned downtime of machines, but the same methodology can be applied to predicting failures within manufacturing processes.

Predictive maintenance decreases unforeseen stoppage time by enabling proactive planning for, and executing of, machine or asset maintenance shortly before the likely failure event. To achieve predictive maintenance, two approaches can be taken depending on the rarity of failure events. To add some context, downtime of a CNC machine due to the catastrophic failure of a component in the hydraulic pump might be hard to predict if historically sensor data cannot show leading indicators that it will fail or that it has failed in the past. But on the flip side, if current spikes have shown in the past that a motor is close to failure, future failures can be predicted. In this case, the current spike is the proxy event. A point to keep in mind is that other events might be used as a proxy for failures, if the failure event itself is not or cannot be captured. For example, maintenance logs, repair invoices, and machine operator run logs could be used, assuming that the proxy event is close enough to the actual failure event to achieve the prediction.

If the failure event is not rare, then supervised learning can be used. In these cases, a model is trained on historical records of maintenance events paired with corresponding time-series telemetry records of leading indicators, such as vibration, temperature, and pressure. The first model family we recommend trying is tabular binary classifiers including XGBoost[2]. The primary downside of these models is that feature engineering

will be required and will be different for every device. For example, relationships between indicators must be manually confirmed by SMEs, such as maintenance engineers, and added to the dataset. However, results can be achieved with less data, model training is faster, and the model will be (relatively) easier to understand. The second model family would be deep learning architectures with sequential input, such as recurrent neural networks (RNNs). The primary downside of these models is the larger dataset required, training times are longer, and the model is difficult to understand. Some level of feature engineering is often still required even for deep learning.

If the failure event is rare, then unsupervised learning can be used. In these cases, real-time outlier detection can be performed on the leading indicator themselves, providing engineers advance warning on potential device failures. For example, if a new sensor is installed then historical data for this sensor is unavailable. However, if domain expertise shows that heavy vibration is correlated with failures in the target device, then outlier events in the new vibration sensor should be brought to the engineer's attention, potentially preemptively catching a failure. In terms of models, the Random Cut Forest algorithm[3] is usually one of the first outlier detection models to be tried on time-series datasets.

Improved production time planning with demand forecasting

Demand forecasting enables more accurate estimations of planned production time by predicting the amount of a product that should be created. To perform demand forecasting, forecast models are produced using historical time-series datasets of product demand. Many time-series forecasting models exist and trade-offs between them need to be considered. Classical methods, such as autoregressive integrated moving average (ARIMA), require less data because they only model a single time-series at a time (for example, a single SKU at a single customer). However, forecast accuracy might be lower, since the model does not take into account other related time-series datasets, such as holidays.

Cutting-edge deep learning based models, such as DeepAR[4], draw correlations across multiple time-series (for example, across product lines and across customers) to produce higher forecast accuracy but requires large amounts of data for the algorithm to discover these correlations. Additional considerations from a business standpoint are also considered such as forecast horizon, product resolution (SKU vs product line), customer resolution (vendor vs channel vs region), etc., to achieve balance between forecast granularity and performance.

Improve performance

As mentioned in the introduction, the OEE calculation for performance is:

$$Performance = \frac{(Cycle\ Time\ x\ Cycles)}{Run\ Time}$$

Therefore, to maximize the performance metric, you must either decrease total *Run Time* or increase *Cycles*. *Cycle Time* itself will not change directly unless the process is optimized or changed.

Process optimization

Process optimization increases performance by optimizing the process variables to maximize production speed while keeping defects in control. There are multiple stages to achieving the goal of fully automated process optimization. Each stage has advantages and disadvantages from a business standpoint that must be considered and balanced. The ideal goal might be a machine that functions with zero human interaction, but below we discuss each step along the way to achieving that goal.

In advance of any discussions of using machine learning for process optimization, we have to mention that machine learning cannot directly substitute simulations for this task. Parameters calculated from first principles should be used for the initial optimization, as the calculations they produce are based on physical or chemical properties. However, where machine learning might be more applicable is in its ability to account for errors that differentiate the results in physical simulations versus the results in real world operating conditions. The two should be used hand-in-hand.

With that said, let's begin our discussion of process optimization. Before we optimize the parameters of the process, we need to establish guardrails on the process itself. For example, before the model recommends increasing the line speed of a machine, it first needs to make sure the proposed line speed will not result in defective parts or unplanned stoppages due to poor quality. This quality prediction application will be discussed more in the quality metric below. Put simply, a quality prediction model should first be built on the historical failures paired with supporting data, such as set-points and ambient sensor readings.

With this quality prediction model in place, we can begin optimizing some processes manually. For example, an operator can first input a setting that they believe will increase operating speeds while still meeting the quality measurements. The model can then determine if this proposed setting, when combined with other settings and sensor

readings, will result in a failure. If the chance of failure is high, then the operator is notified and can manually try again with a lower setting.

Next, regression models can be trained to predict the cycle time of the process. The data will be the same to before but the target will now be a number (for example, parts per, throughput or flow) rather than a class (for example, pass/fail). This regression model serves as the foundation for the process optimizer. With the two models running in tandem, the operator can validate new settings to increase the process or machine speed while staying within the process guardrails. For example, the operator can propose an increase in a setting and the machine learning models might indicate that this setting will likely result in a higher line speed while producing a quality part. The operator can then test this setting.

However, one caveat is that because the models are trained via supervised learning, they will only know about certain scenarios if it has seen the scenario in the past through the historical data.

In certain edge scenarios or where exploration is costly, historical defect data might not exist or be limited in quantity. Likewise, historical data on machine performance and the corresponding settings might be conservative, and the data for the actual maximum speed might not exist. Because the model has never seen these examples before, it will not be able to foresee some failures or be able to prescribe the optimum speed.

To be able to apply machine learning models for automated process control, a data set has to be built which covers nearly all scenarios—to the point that there is a high degree of confidence that the model can identify and make the correct prediction. Until you are at that point, the models can be used by the SMEs (manufacturing engineers and operators) to safely fine tune and explore settings, thus combining the best of both human knowledge and machine learning predictions.

Over time, the regression model will contain information about how each setting contributes to the cycle time. The SMEs can manually experiment with the most mathematically impactful settings while making sure the experiments make sense given their previous experiences. Over time, the process optimization algorithms, such as Bayesian Optimization or the reinforcement learning model, will see more scenarios and its prescriptions of optimum settings will increase in accuracy.

When there is confidence that the model has seen enough cases, it can be deployed in a process optimization solution that, given a set of input variables, can automatically push settings to the machine that will result in the optimal machine or line performance.

Improve quality

As with the other OEE variables, quality is calculated with:

$$Quality = \frac{Good\ Count}{Total\ Count}$$

Before we get into the specifics of quality predictions to clarify, improving quality is not the same as identifying defects post operation. The outcome of improving quality is to ensure that the variable controls of the manufacturing processes are set so that poor quality parts are not manufactured in the first place.

Automated quality testing

Automated quality testing decreases overall run time by offloading one or more inspection tasks to automated systems, thus reducing the time spent performing slower manual inspections. Automated quality testing falls into the realm of human augmentation in manufacturing, where tedious or simple tasks once performed by teams of highly trained humans can be offloaded to an algorithm, freeing these experts to perform more impactful tasks. To perform any augmented quality testing, the pass/fail decisions made by humans must first be captured, as well as supporting indicators the quality inspectors used to make the decision (for example, color, weight). A classification model is then trained on this dataset to automatically classify future parts/products as pass or fail. These classification models can come in many forms depending on how the dataset is formatted. In this section we will cover two formats commonly found in quality testing use cases: tabular and images.

Tabular is the most common format of dataset used in quality testing because it is the easiest to capture. If available, relational database tables containing past pass/fail records are joined with corresponding supporting indicators tables and exported as `.csv` files into a data lake for model development. In the case when quality databases are not available, spreadsheets with manually-tracked records can be put into the data lake as exported files (though this is not recommended as it's prone to manual errors and is not scalable). The resulting `.csv` files should contain a minimum of three columns: timestamp, device ID, and pass/failure record. Other columns should include supporting indicators that the quality inspectors used to make the pass/fail decision. These indicators are different for every product but could be factors, such as weight, dimensions, hardness, etc. Once this dataset is captured, feature engineering is usually performed to better match the inspectors manual process, such as converting weight and dimensions to density. A tabular classification model, such as XGBoost, can then be trained on these historical records.

Image data is not as common as tabular data, but is receiving significant attention these days due to the recent research and applicability of computer vision. In these use cases, historical human-labeled pass/fail records are paired with captured images that the quality inspectors reviewed during inspection. These labels can be at a part level (for example, the whole circuit-board) or at a defect level (for example, scratches and dents). Feature engineering is often performed on the raw images to enhance the features that the inspectors use during their decision-making process. These steps could include removing unnecessary parts of the image, enhancing edges, enhancing colors, and more. Image augmentations, such as flipping, rotation, and brightness/contrast modifications, are also often performed to help the model generalize better.

After these pre-processing steps are completed, a model is then trained on the processed images. For part-level classifications, deep learning-based image classification architectures are used including pre-trained models, such as VGG or Resnet families. For defect-level classifications, object detection architectures are used including pre-trained models, such as SSD (Single Shot Detection) or Yolo. However, be aware that pre-trained models are trained using everyday objects, such as cats and dogs, which might not carry over to your specific images of manufacturing parts or assemblies.

When it comes to quality of a part or product, the application of computer vision for quality is not specific to pass/fail. Computer vision can also be applied to augmented quality classification or sorting of product. This exercise can decrease run time by offloading one or more classification to automated systems. This use case is the same as augmented quality testing, but instead of a simple pass/fail decision, one or more classes can be assigned, such as grades, sizes, or downstream treatments. The same technology that applies to augmented quality testing can be applied here. The only difference is that the labels are now multi-class instead of binary and could be used to sort the quality of raw product, such as timber or food produce, into the different categories that are used in the manufacturing processes for the end product.

Quality processes prediction

Quality prediction increases the good item count by detecting and remediating potentially defective parts from the upstream production pipeline by monitoring operational process KPI. For example, monitoring the pressure and temperature of an injection molding machine mold and providing notification if bad parts will be made unless remediation actions are not taken to the molds recipe. The application of quality predictions is very similar to predictive maintenance if the prediction was on the part

rather than on the machine and the manufacturing operation process is remediated rather than the machine being maintained. Otherwise, the techniques are the same as for predictive maintenance.

If possible, a labeled time-series dataset of pass/fail inspections across multiple stages of production, along with leading indicators supporting this pass/fail inspection, is preferred. The existence of this labeled dataset supervised learning to take place using ML algorithms, such as XGBoost or RNNs. If a labeled time-series dataset does not exist, for example if upstream defects are rare, then an unsupervised approach can be taken. Just like with predictive maintenance, leading indicators are tracked over time and anomalies in these indicators are detected and reported to engineers. Random Cut Forest would also be an algorithm that could be utilized for this anomaly detection case.

Culture and organization

Applying machine learning in manufacturing to improve OEE is not just a technology implementation, it also requires a culture change and organization mind shift. To become a data-driven manufacturer with the application of ML prediction not only requires the trust of the manufacturing organization but also the dedicated time to train and integrate the models into the manufacturing processes. The adoption of ML to drive efficiencies should be a top down strategy with key stakeholder buy-in. Otherwise, it is unlikely to be successful.

With an organization, the application of ML also requires a new skill set, which manufacturers have historically not possessed, that is, the role of the data scientist. The data scientist's role is to interpret the data and train the models but they cannot do this in isolation and need support from the experts in the manufacturing business and IT to be successful in providing them the business context support to understand the data and the IT tools to get to the data and train the models.

A successful implementation of machine learning will only come about by having an organization that is ready to embrace a new culture. Get VP or C-level buy-in, have the willingness to experiment, show value, and then scale out for success.

A great example of this is INVISTA and how they changed their culture to move from business intelligence (BI) to AI.[5]

Getting started

Hopefully from the previous sections of this paper you have a good foundational understanding of machine learning concepts and algorithms, and how they can be applied to improve OEE. When it comes to getting started with implementing machine learning for your OEE use cases, there are some structured steps that you should take on your journey as outlined in Figure 4e 4.

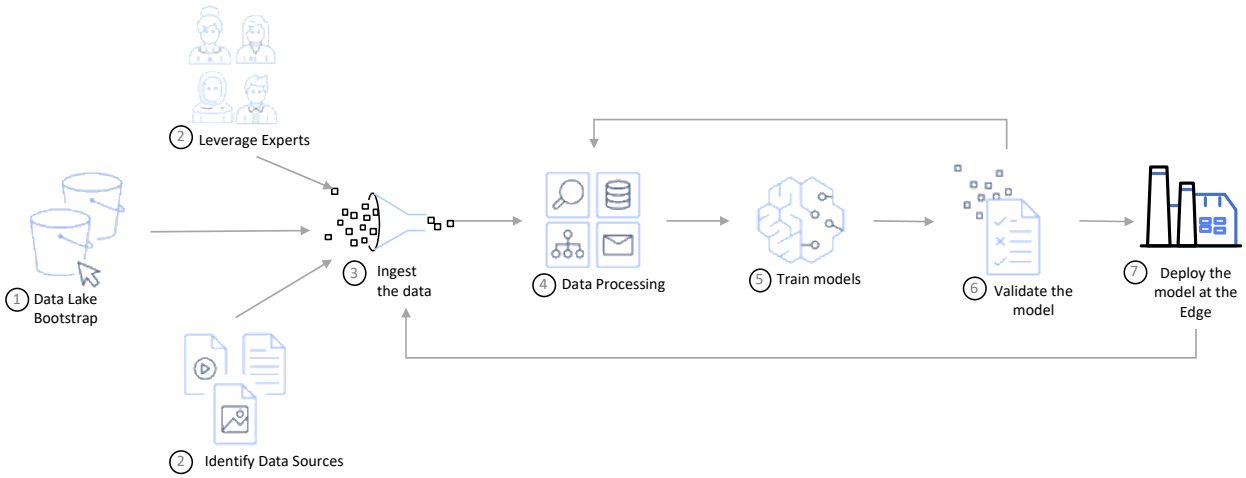

Figure 4. Getting started with the machine learning process.

1. Architect your data lake

It is not imperative that you have a data lake strategy in place from the beginning, but it is beneficial to ensure that the foundational components of a data lake (see Figure 1) are in place to enable the training of ML models on the created data sets. We recommend that you:

- Set up your data stores for storing the raw data and data sets using Amazon S3.
- Set up the catalog for metadata using AWS Glue.
- Ensure that a security policy is in place using IAM roles, policies, and Amazon S3 bucket policies.
- Ensure that a data lifecycle policy is in place for archiving using Amazon S3 lifecycle policies.

All of the above components can be built using AWS Lake Formation, which makes it easy to set up a secure data lake by simply defining data sources and what data access and security policies you want to apply. Lake Formation then helps you collect and catalog data from databases and object storage, move the data into your new Amazon S3 data lake, clean and classify your data using ML algorithms, and secure access to your sensitive data.

2. Identify the data sources and experts

This step is about ensuring that when it comes to working backwards from the business problem, resources from the business and IT organizations are working together to identify and interrupt the data needed for the training data set. Some of the initial hurdles when it comes to implementing machine learning are normally around not having sufficient training data to train the model to identify the defect or anomaly, or missing the business context to interrupt the data for the training data set.

3. Ingest the data

After the training data has been identified, the next step is to ingest, or load, the data into the data lake. This exercise will most likely be a batch upload to the data lake initially, but going forward an automated pipeline should be configured to ingest the raw data for the data sets. When it comes to operational telemetry data from machines, PLCs or sensors then AWS IoT can be used to stream messages into the data lake S3 buckets via Amazon Kinesis as shown in Figure 6. For images, a similar ingestion path can be used to copy or move the images into a S3 bucket using AWS Transfer for SFTP (Figure 5). For structured data sets from enterprise and manufacturing applications, an ETL process can be set up which exports the data set on a schedule or pulled via an API.

4. Data processing

After the data has been ingested, the next step is to condition and cleanse the data into a training data set. This can be done through a set of ETL or scripted jobs, which ensure that the data can be used to train the data. For example, all images are labeled or time gaps are removed from time series data.

After the data has been processed, consolidated, and cleansed, the next step is to perform feature engineering. During this step, data scientists work with SMEs to bring structure to the domain knowledge. This structure provides the ML algorithms with higher impact data for the decision logic to be built upon. For example, a new field might be calculated from two or more existing fields, or new images generated from existing images. This is carried out in Amazon SageMaker Notebooks.

5. Train the models

Next, a model training job is started with the notebooks. These jobs can leverage any of the built-in Amazon SageMaker algorithms (many of which have been mentioned in previous sections) or the job can call custom scripts, containers, or deep learning frameworks.

6. Validate the models

When model training is complete, the model results are validated and compared against the business objectives as measured by relevant metrics, such as precision or recall. Amazon SageMaker Experiments can be used to capture, organize, and search previous modeling runs. If the objective is not met, then further iterations are taken, repeating steps 4 through 6 until the goals are met.

7. Deploy the models to the edge

After the model is validated to meet the business objectives, the next step is to deploy the model. This can be done as part of an application in the AWS Cloud or to the edge where it can perform inference within the 4 walls of the factory. With AWS IoT Greengrass installed and configured on an edge gateway or industrial PC connected to the factory network, the ML model can be deployed and configured to perform inferences as part of the ingest data rule set in parallel with the data being ingested into the data lake, as shown in Figure 5 and Figure 6, depending on the use case. Amazon SageMaker Model Monitor can be applied as well to make sure the models stay relevant, or retraining pipelines can be added to make sure the new data ingested is constantly used to keep the models up to date.

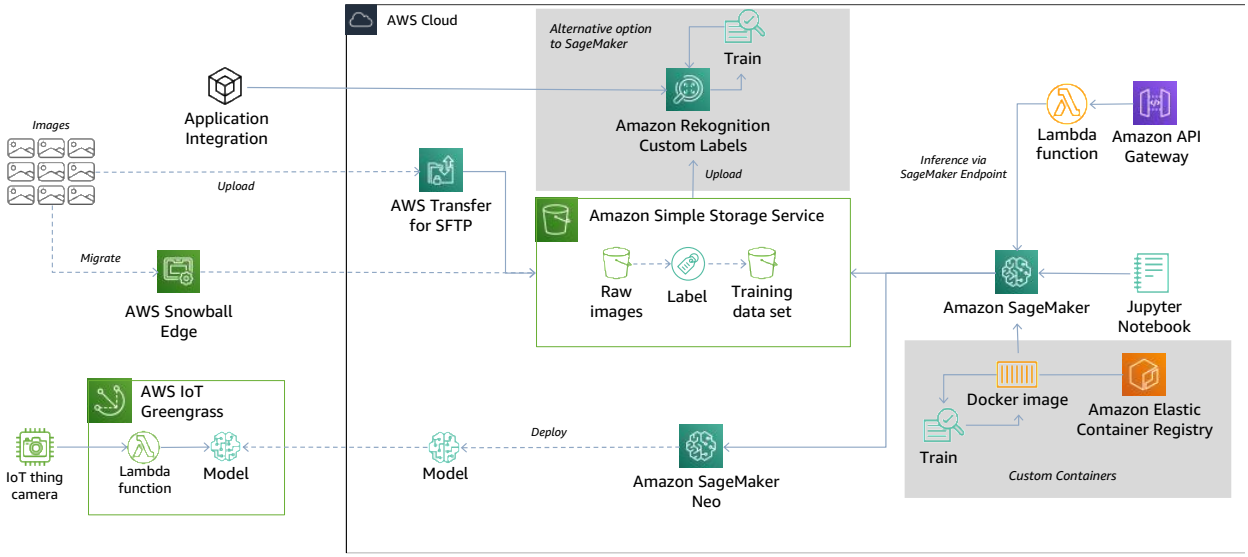

Figure 5. Manufacturing Computer Vision Reference Architecture

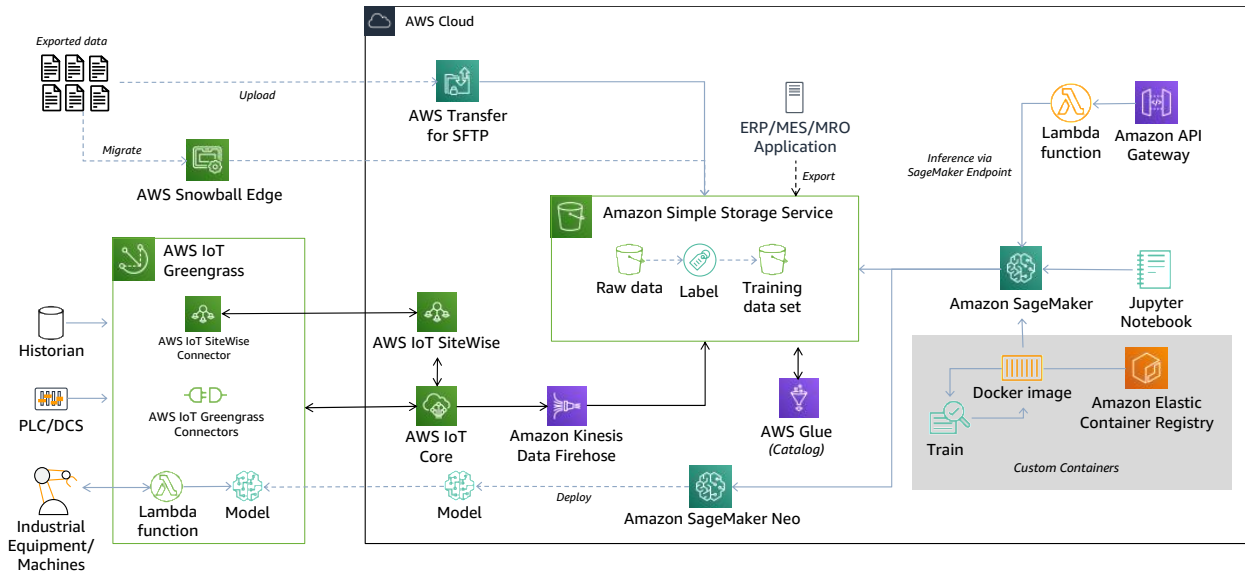

Figure 6. Production Optimization Reference Architecture

Conclusion

To successfully deploy ML to improve OEE, it is always advisable to work backward from the outcome in terms of the business value. The output results must be measurable against specific business objectives and success metrics to show the value of machine learning. Plan to succeed, and choose pilot use cases that are manageable and which have a clear path to production. There is no point choosing a use case to apply ML to identify defects if a human or measurements can't detect the defect currently. Finally, to train machine learning models, you must have reasonably high quality data for the data sets. Ensure that the training data is available and can be easily ingested into the data lake and is of a format that can be used to train a model.

Contributors

Contributors to this document include:

- Steve Blackwell, Tech Leader Manufacturing, Amazon Web Services
- Jake Chen, Senior Data Scientist, AWS Professional Services
- Scot Wlodarczak, Principal Marketing Lead, AWS

Further Reading

For additional information, see:

- Machine Learning Foundation whitepaper[6]
- Machine Learning Lens – AWS Well-Architected Framework[7]
- Managing ML Projects whitepaper[8]

For next steps, AWS Training and Certification has courses and certifications that provide hands-on experience in applying machine learning using AWS services. Additionally, AWS Prototyping and Machine Learning professional services teams can assist in developing use cases, building proof of concepts and scaling out machine learning within your manufacturing organization.

Document Revisions

Date	Description
October 2020	First publication

Appendix: Machine Learning algorithms

Machine learning algorithms are numerous but can be grouped into a few problem sets or tasks that they are designed to address. Below, we will review some common tasks and the algorithms for supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning

Supervised learning is a method of machine learning where the algorithm is trained by providing specific inputs where the outputs are known. For example, predicting machine failures using historical maintenance logs and machine data. The model is trained until a desired level of accuracy is reached using the training data or until the input to output mapping information contained in the dataset is exhausted. Example of supervised learning algorithms are:

- Linear Regression
- Logistics Regression
- Decision Tree
- Random Forest
- XGBoost
- Deep learning (LSTM, RNN)

Unsupervised learning

Unsupervised learning is when the algorithm is trained without any feedback or outcome and the algorithm finds patterns in the data set. In context with the supervised learning example for predicting machine failures in the case of unsupervised learning there are is no historical data showing failures and the model is training using leading indicators. Examples of unsupervised learning algorithms are:

- Clustering
- K-means
- Spectral
- Outlier Detection

- Random Cut Forest
- Apriori algorithm

Reinforcement learning

Reinforcement learning differs from supervised and unsupervised learning by solving sequential decision problems instead of classification and regression tasks. Training in reinforcement learning comprises an agent trading off exploration and exploitation, based on the reward it receives, to optimize the action policy in an environment.

Table 1 gives an overview of each method, the data and the main use case application.

Table 1 – Machine Learning Algorithm types.

Machine Learning Algorithms	Data Type	Main Use Case
Supervised Learning	Labelled Data	Prediction
Unsupervised Learning	Non-labelled Data	Identify Patterns
Reinforcement Learning	Decision Process	Learn Actions

Notes

[1] https://www.leanproduction.com/oee.html

[2] Gradient Boosting (e.g. XGBoost) algorithms use both regression models and decision trees for the prediction combining multiple weak or average predictors to a build strong predictor. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

[3] Random Cut Forest is an algorithm for detecting anomalous data points within a data set through a collection of decision trees which classifies an object based on attributes with each tree giving a classification and each tree voting on the classification with the forest choosing the classification with the most votes.

[4] DeepAR is the application of deep learning to produce accurate probabilistic forecasting based on training an auto regressive recurrent network model on time series data. DeepAR trains a global model to forecast multiple correlated time series which have been generated by the same process or similar processes. Based on the input dataset, the algorithm trains a model that learns an approximation of this process/processes and uses it to predict how the target time series evolves. Each target time series can be optionally associated with a vector of static (time-independent) categorical features and a vector of dynamic (time-dependent) time series. DeepAR also has a scaling mechanism to handle amplitude variation across time-series as the amplitudes in the input time series can vary drastically.

[5] https://www.youtube.com/watch?v=4WYoO52GW4M&t=33s

[6] https://d1.awsstatic.com/whitepapers/machine-learning-foundations.pdf

[7] https://d1.awsstatic.com/whitepapers/architecture/wellarchitected-Machine-Learning-Lens.pdf

[8] https://d1.awsstatic.com/whitepapers/aws-managing-ml-projects.pdf