A Guidance Framework for Operationalizing Machine Learning for AI

Published: 24 October 2018 **ID:** G00366587

Analyst(s): Soyeb Barot

As AI and ML initiatives mature, the biggest challenge faced by technical professionals is operationalizing ML for effective management and delivery of models. This guidance document provides a framework to build and deploy ML models in production for successful delivery of AI-based systems.

Key Findings

- Transitioning ML models to real-world production applications is becoming more critical as artificial intelligence (AI) and machine learning (ML) initiatives mature, and the emphasis has shifted from development toward deployment and continuous delivery.
- The operationalization process is still seen as an art. Organizations are at a loss to systematically productionalize ML and face significant scalability and maintenance challenges.
- ML projects are inherently different from traditional IT projects in that they are significantly more heuristic and experimental, requiring skills spanning multiple domains — statistical analysis, data analysis, platform engineering and application development.
- Defining an operational framework and identifying key supporting components like model management systems are critical toward building and delivering effective AI-based systems.

Recommendations

Tech professionals responsible for implementing and managing analytics and BI strategies should:

- Leverage ML, DevOps and DataOps expertize to help establish a machine learning development life cycle (MLDLC). Use a data-architecture-first approach to support multiple ML pipelines.
- Create an integrated ML platform that can blend disparate functions, data processing, model training, model management and model monitoring to support sophisticated AI use cases. This will allow for continuous delivery of AI-based systems within IoT, natural language processing (NLP), machine vision or predictive models within analytical platforms.

- Focus on managing and monitoring models alongside maintaining the associated metadata to help reduce the anxiety for scaling AI-based solutions.
- Reduce the technical complexity by using a simplified architecture and framework for deploying ML models by monitoring and constantly re-evaluating their business value.

Table of Contents

List of Tables

List of Figures

Problem Statement

By 2021, at least 50% of machine learning projects will not be fully deployed. This is due to the lack of a framework and architecture to support model building, deployment and

> monitoring, which inherently will affect the continuous delivery required for AI projects.

The field of ML and AI is exploding due to the advent of big data and the need for analytical insights in real time from data that has become multimodal and heterogeneous (such as videos, text, images, social interactions and comments). AI has become part of our everyday lives, from self-driving autonomous cars, mobile banking and machine translation services to virtual private assistants and creating personalized content. AI as a technology or a system helps emulate human behavior typically by learning, appearing to understand complex content, but understanding the components and/or process for delivering AI capabilities is not as straightforward. Machine learning helps build models and algorithms to process large amounts of data and learn on its own, with minimal supervision assisting with building AI-based systems.

On their journey to build AI-based systems, most organizations have defined the process to build, train and test ML models. The challenge has been what to do once the model is built. Integration, deployment and monitoring are essential aspects to provide for continuous feedback once the models are in production. This is where the entire process of building ML models aligns more closely with the application development life cycle or software development life cycle (SDLC) than with an endpoint analytics project.

Democratization of machine learning platforms is proliferating analytical assets and models. The challenge now is to deploy and operationalize at scale. For technical professionals, understanding the proper machine learning apparatus plays a major role in managing and deploying enterprisewide ML products that can support implementation of AI-based systems.

This is not the case when it comes to machine learning, the initial steps remain the same where we collect the data, curate it and establish a data pipeline to build and train the model. However, after testing and validation, the model needs to be integrated within an application or analytics platform, which we will discuss later in the document.

> Once deployed, the model needs to be managed and monitored to ensure it performs optimally within the thresholds defined by the business.

There is a clear need for a feedback loop to ensure there is no model drift or degradation that would require continuous tweaking of the model and periodically retraining it (see Note 1). You need to establish an analytics- or model-building life cycle in order to build, deploy and manage the ML or statistical models for optimal performance of the application and/or consistency in the results from an analytics platform. Hence, there is a need for you to operationalize the ML model development process in order to establish a continuous delivery cycle of models that form the basis for AI-based systems.

Traditionally, the focus of analytics and data science teams has been on developing reports, dashboards and models, while dealing with the operationalization of these analytical assets has been an afterthought. The reasons for this oversight are:

- Lack of a process-centric understanding from the data science and ML teams
- Communication gap between the machine learning/data science team, the IT operations, DevOps, the application team and the line of business
- Unwillingness to deal with the operational aspect of machine learning and AI projects, along with a misunderstanding of their practical business impact once in production
- Lack of a formal operationalization methodology due to the absence of a development and management framework for ML models

> Operationalization of machine learning is an imperative step in aligning AI investments with strategic business objectives — the "last mile" toward delivering business value.

The goal of this document is to provide solution architects and machine learning architects an optimal framework, architecture and set of tools and technologies needed to start and maintain an enterprisewide ML initiative. Within this guidance framework, we will address the following questions:

- How can we build, test, deploy and manage ML models?
- How do we design an architecture to operationalize the ML model development process?
- How do we leverage the architecture for building AI-based applications or model-driven analytics projects?

The target audience for this document includes solution architects, ML engineers, data engineers, AI application developers and business intelligence developers. This research can also be leveraged by CDOs, CAOs and project managers to better understand the operationalization of ML, which is critical for building AI-based systems.

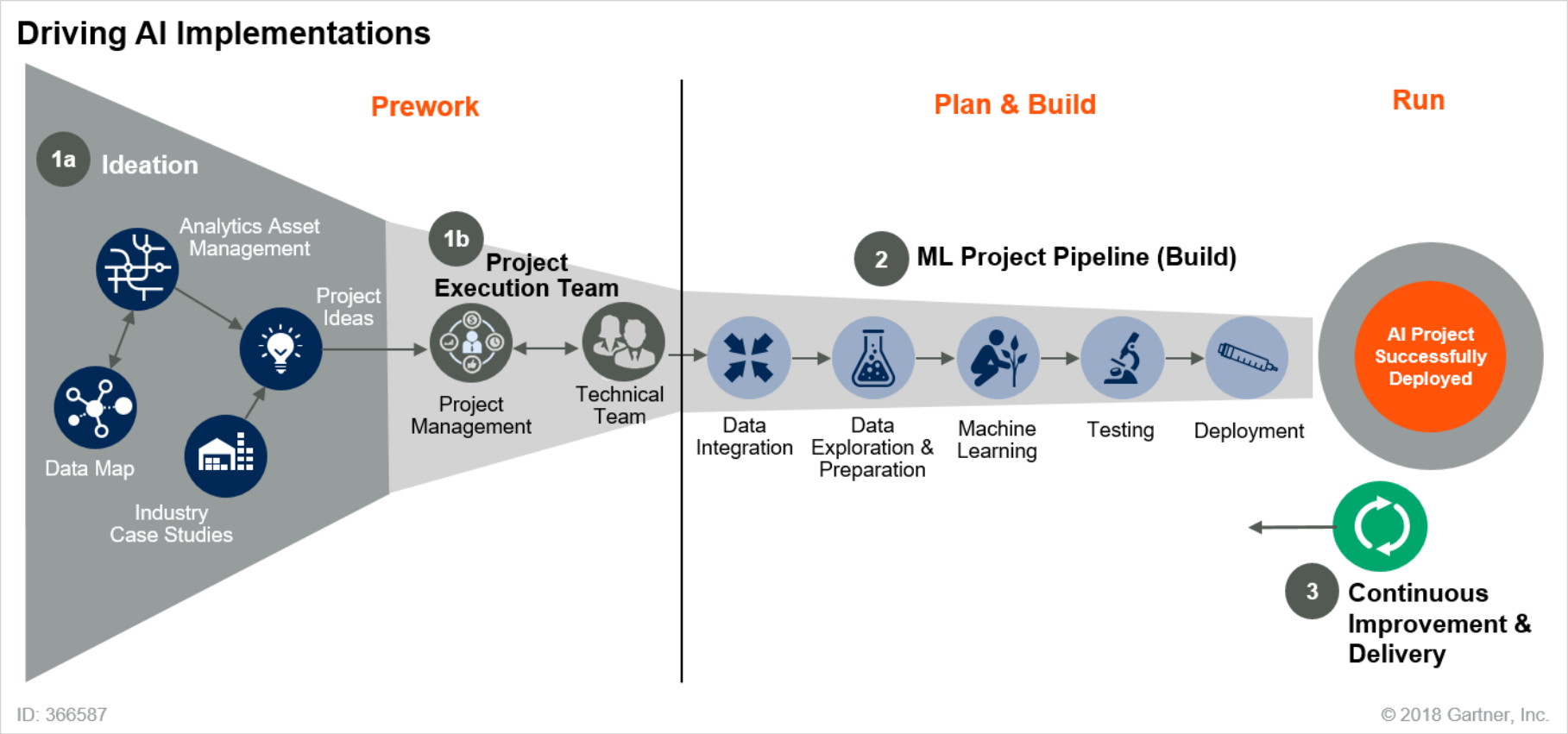
The Gartner Approach

ML is at the core of an AI project. Therefore, ML projects need to have a strong data science foundation at their core. In most cases, organizations struggle with establishing a strong data science foundation for their ML projects, which inadvertently affects the delivery of AI-based systems. This is due to a lack of a framework, a well-defined architecture and an implementation model that establishes an ML model development life cycle. For this reason, this research concentrates on AI projects that have a significant data science and ML component.

ML is more heuristic as compared to traditional BI projects, with lots of trial and error needed, and it is being applied to solve business problems for which no exact solution exists or automate business processes with intelligent AI-based systems. Also, the debugging of ML systems is quite different from traditional analytics projects or applications, which is also the case with testing. Given the special nature of ML, the complexities range from finding appropriate use cases to finding the data, understanding different implementation approaches and integrating it thereafter within the business application.

The best practices outlined within the Gartner Approach are reflected in Figure 1, conceptualized in the form of a project funnel. Nodes 1a and 1b cover the Prework section that lays the foundation for building an AI-based system, while node 2 illustrates the Plan and Build portions of the guidance framework. Finally, within node 3 we will cover the Run portion. These nodes have been expanded and covered in greater detail within The Guidance Framework section.

Figure 1. Driving AI Implementations With Continuous Delivery of ML Projects



Source: Gartner (October 2018)

The Prework involves:

- **(1a) Building a cache of ideas for ML and AI:** These could originate from industry case studies or working closely with business partners by understanding existing analytics processes. An existing data map can also serve as a pool of ideas to further drive existing project ideas (for example, bringing together data from transactional systems, semi and unstructured data [speech, image, video] and leveraging natural-language processing for analysis).
- **(1b) Putting together the project execution team:** After having generated your first wave of "project ideas," you need to formally convert them into project proposals that are tracked and managed efficiently by a project management team. Have the appropriate cast of characters, with skill sets from varied computer science and engineering disciplines that are accessible and collaborative. Some can be permanent members of the team, while some can be sourced in an ad hoc basis according to the project's requirements. Understand that finding the right balance depends significantly on the actual and ongoing project portfolio. Avoid making just one job role (data scientists, for example) responsible for too many things. This will be not only a cause of frustration, but also counterproductive. We will look at the different roles within the Prework section later in the document. This is the stage where you fully define the solution, build a project plan and assemble a team with the best-suited skills together.
- **(2) Designing and building a machine learning delivery pipeline:** This we where we further break down the architecture into components supporting data integration to:
  - Bring the required datasets within an LDW, exploration
  - Identify the data and feature engineering variables required for model building and model evaluation
  - Test the model against a testing dataset
  - Deploy the model integrated within an application or analytics platform
- **(3) Establishing a continuous improvement and delivery process:** An essential component of the architecture that helps create a feedback loop. This ensures there is a continuous improvement process for the models deployed in production alongside a delivery mechanism. Since most AI applications are built using an ensemble of models, they require a versioning, monitoring and management framework.

This guidance framework focuses on providing a systematic approach to build and operationalize the machine learning model development process, hence building the foundation for delivering on AI initiatives. Gartner recommends that you follow the sequence laid out within the framework in the next section to establish a platform that can support a continuous delivery cycle of ML models.

The Guidance Framework

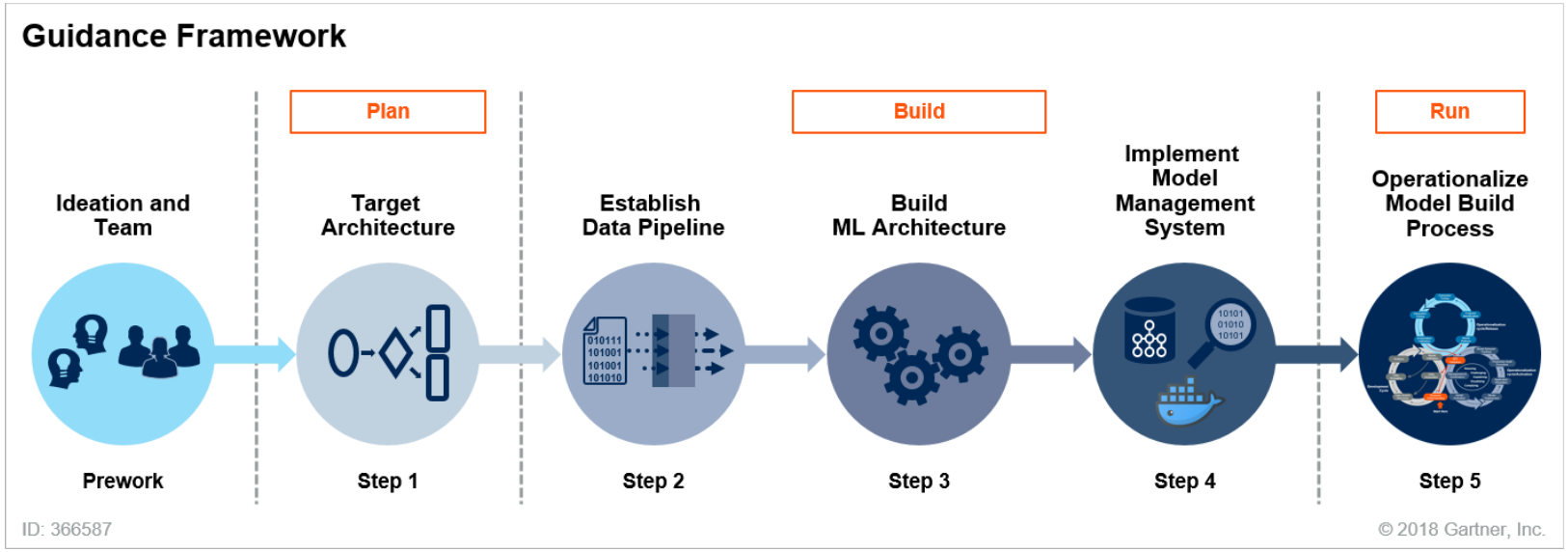
The implementation of the machine learning development life cycle (MLDLC) architecture follows Gartner's familiar guidance framework to Plan, Build and Run the platform as part of operationalizing the development of AI-based systems. Within the previous section, the ideation of

use cases for building the AI-based system and prototyping cover the sections of Prework and Plan. When looking for use cases and how to build an AI strategy, please refer to the following Gartner research:

- “Driving an Effective AI Strategy”
- “Laying the Foundation for Artificial Intelligence and Machine Learning: A Gartner Trend Insight Report”
- “A Framework for Applying AI in the Enterprise”

Within this document we will focus on the Plan, Build and Run stages of the guidance framework. For simplification, these stages have been stretched across multiple steps. The complete course of action is shown below in Figure 2.

Figure 2. Operationalizing ML — Guidance Framework



Source: Gartner (October 2018)

We will focus on ideation and putting together an execution team as part of the Prework step. Within Step 1 is where we will Plan and Design the target architecture to operationalize the model-building process. In Steps 2 through 4 we will cover the building blocks of the target architecture, and finally we will bring it all together in Step 5 to complete the Run stage of the guidance framework.

- Prework: Ideation and Team
- Step 1: Design Target Architecture
- Step 2: Establish Data Pipeline
- Step 3: Build Machine Learning Architecture
- Step 4: Implement Model Management System
- Step 5: Operationalize Model Build Process

Prework: Ideation and Team

Ideation to Set Platform Objectives

Creativity is defined by our ability to bring change that is driven by innovative life. The addition to modern machines and technology can help us bring those ideas to life. We have machines that can "learn" and help us realize those ideas through a design of complex and distributed programs ciphering through vast amounts of data — more than is possible by humans. ML and AI provide us with the tools to bring those new and innovative ideas to fruition.

Unlike other IT projects, the appropriate moment for seeding a project portfolio approach is during the ideation phase, and should start with obtaining a solid set of project ideas:

- AI and ML implementation use cases may come through industry case studies (see node 1a in Figure 1). Gartner's hundreds of industry and business function analysts can be a rich source of ideas as well (see "A Framework for Applying AI in the Enterprise").
- Gartner recommends creating a "data map," where we suggest you look for existing analytical and data assets — historical and external market data sources that have remained untapped. Business processes that can be automated, to further drive existing business ideas, and uncover the need for new sets of data should be included within the analytics process (for example, further transactional data, natural-language processing [NLP], speech and image data).
- Bring together the lines of business, as well as the wider IT organization, to best determine which applications generate lots of data, what additional sources could be incorporated and whether there exist underutilized or untapped data and processes that can be automated by leveraging ML and AI. Involve IT and the business, data scientists, and digital/innovation teams in brainstorming sessions to address business problems and ideate new solutions to maintain the competitive edge.

Recommendations:

- Collectively develop candidate use cases of AI and ML, while identifying the quantifiable business outcomes from each of these use cases. Agree on measurable business impacts before gathering the first set of data and begin the journey to build the framework.
- Prioritize the selected AI projects based on their technical feasibility and measurable economic viability — that is, on their ability to be properly measured while in production, delivering significant value to business.
- Along with ideas and technology (such as ML), put together a team of individuals with the right set of skills.

> Data + Ideas + Machine Learning + Team = Continuous Innovation With AI

Technical Team

Enterprises looking to establish an ML delivery team to support the building of AI-based systems need to look beyond the realms of a traditional analytics team structure. Machine learning and AI projects require a substantial amount of infrastructure, application, data and DevOps expertise to fully support:

- Building ML models
- Framework for delivery of ML models
- Integration of models within AI-based systems
- Monitoring of models within applications and analytical platforms

This thereby operationalizes the entire ML development life cycle.

Gartner has identified the following roles, all of which may or may not be permanent within the data science team, and which support the ML development life cycle. However, it's imperative that they are accessible during the various stages of AI projects. They include:

- **Data Scientists:** Critical key staff members that are responsible to extract various insights from the data. They provide the ability to pull together analytical methodologies from a broad range of fields. Leverage existing algorithms or create new ones if need be. As all data science is geared toward actionable results, knowing the success criteria in advance — in terms of ROI, accuracy and objective — is an absolute must. Hence, these individuals provide an overview of the end-to-end ML development process while working very closely with business and other IT engineering teams.
- **Citizen Data Scientists (aka Data Analysts):** These individuals typically come from within the lines of business, bringing in domain expertise. They have a strong intuition about regression, testing hypotheses and exploring data. They are also able to pursue a good stretch of data-science-oriented business scenarios. They are not formally trained as data scientists, but can

still execute a variety of “simpler” data science tasks, especially with the help of new-generation data science tools. Use your core data scientists to mentor and train as many citizen data scientists as possible. These data-innovation evangelists play a critical role in presenting insights gathered by data scientists to decision makers about the potential business impact of newly built ML models. This function can help bridge a significant gap by synthesizing complex algorithms, their outcomes and applicability within AI-based systems.

- **DevOps/DataOps Engineer:** This is a key engineer function within the framework to operationalize ML for AI. The DevOps/DataOps engineer identifies and optimizes the interdependencies between data stores, applications, development and execution of models and infrastructure operations. Apart from making the appropriate data accessible and available for data and citizen data scientists, they play an instrumental role in providing optimized infrastructure to build and train the models. They are required to collaborate with data engineering, development, QA, release, operations and infrastructure teams to maintain high-quality artifacts as they support underlying architecture for ML and AI-based system development.
- **Machine Learning Architect (aka Artificial Intelligence Architect):** Define, evaluate and recommend machine learning architectures and frameworks that support ML and AI initiatives. These individuals help develop machine learning prototypes utilizing ML frameworks, orchestrate the deployment and management of models in production environments, and have a working understanding of the applicability of ML models within the various disciplines of AI, such as:
  - Natural language processing — Chatbots, translation services and sentiment analysis services.
  - Computer vision — Training and implementation of image recognition models and support for streaming video analysis
  - IoT use cases — Deployment of IoT edge devices and models within sensors

  ML architects are responsible for the continuous evaluation and improvement in the performance of various algorithms, models and strategies based on the available datasets. Therefore, they work very closely with the data scientists and business stakeholders overseeing the process of managing, integrating models within production AI-based systems and monitoring them. ML engineers play an important role as part of the feedback loop that supports the fine-tuning and retraining of ML models.
- **Solution Architect:** Provides technical leadership and a deep understanding of business goals, processes and the overall solution architecture. They help design the overall architecture and support the development of ML models, selection of tools and development of AI-based technology solutions aligning with enterprise architectural standards. The architect leads the evaluation, design and analysis of the enterprisewide ML and AI architecture, translating business requirements into architectural blueprints. Alongside collaborating with the data science and ML team, they work with the enterprise architecture, data management architecture, information security, application, and infrastructure teams, producing technical documentation of systems and architectures. They act as subject matter experts on a broad

range of technologies, products and vendor offerings to meet performance and other SLA metrics.

- **Application Developer:** Even though this position requires sporadic custom coding, the primary function is to act as an application integration specialist for deploying ML models within applications built within NLP, computer vision or IoT-based use cases. These individuals provide the deep technical expertise, understanding and integration of technologies and architecture. They are responsible for defining standards and guidelines and governing the integrated processes that support the organization's application environment. Working with other IT teams and business analysts, they help understand and maintain the integration needs of the target-state application. They partner with product owners to align the product roadmap, work on technical design, identify gaps and work on remediation strategies to minimize the impact on AI-based systems deployed in production.
- **BI Engineer/Developer:** Provides technical expertise into the integration of predictive models within BI and reporting tools and, in some cases, recommendation engines. Working closely with business stakeholders and data analysts, they ensure the development of reports and dashboards with the right set of visualizations for storytelling. These dashboards help deliver a compelling story about the data, as well as the insight derived by the data and citizen data scientists, and are responsible for bringing about a standardization to the entire analytics delivery process. These individuals develop conceptual, logical and at times physical data models supporting the data analysis and business intelligence process with the incorporation of predictive models built using ML. They are primarily responsible for overseeing the analytics development life cycle, which is a derivative of the ML model development life cycle. Working closely with the ML development team, business analysts and data architecture teams, they develop analytics solutions based on business data consumer needs.

Apart from the roles and characters identified to support the ML operationalization process to build AI-based systems, you would require a product owner (business stake-older), a business analysts and a project manager to support the overall ML and AI initiative and continuous delivery of solutions that provide significant ROI to the enterprise.

Details on the roles for building data science and ML teams can be found within "Must-Have Roles for Data and Analytics, 2018."

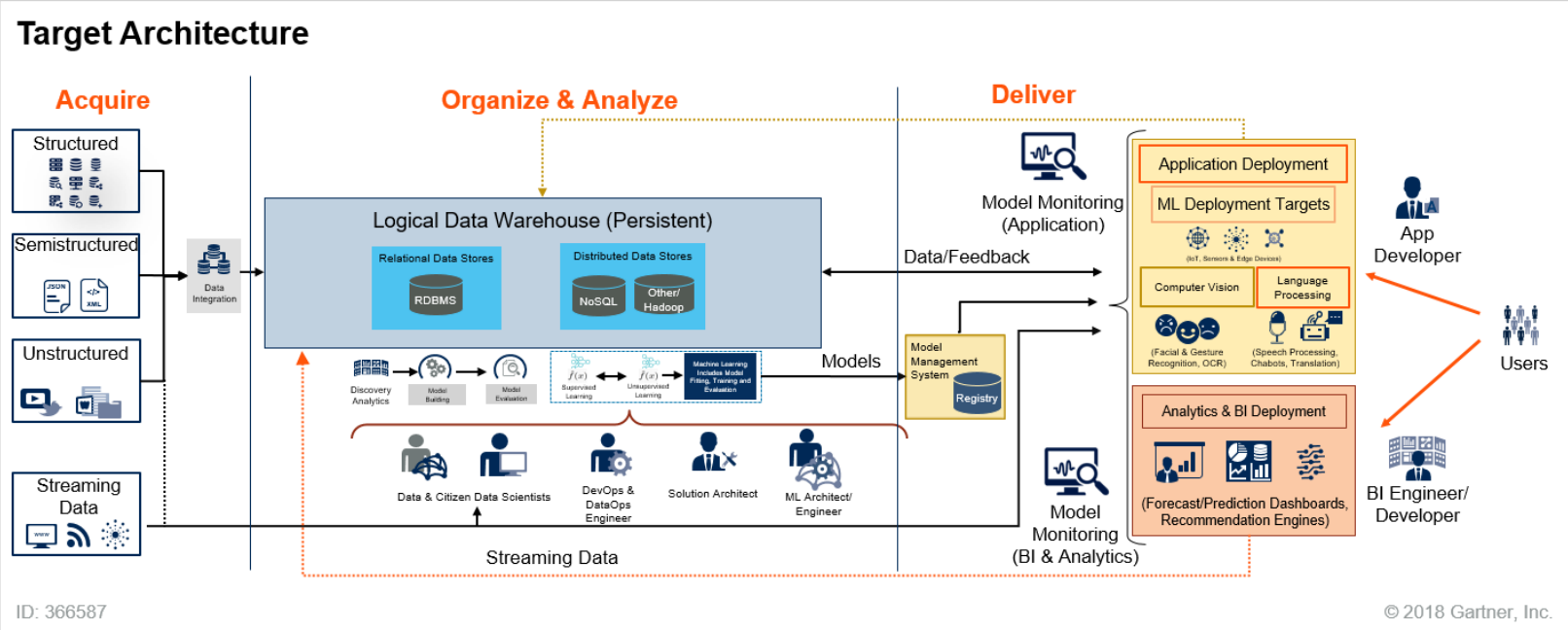
Having completed the prework, identified use cases through ideation and assembled the technical implementation team, you can now proceed with the steps laid out within the guidance framework.

Step 1: Design the Target Architecture

As part of the first step, the solutions architect and the ML development team need to collaborate to design the target architecture that will support the ML model development, management and deployment process. They need to factor in all of the components seen typically within a data management and analytics architecture. For more guidance on implementing a comprehensive data and analytics architecture, refer to "Solution Path for Implementing a Comprehensive Architecture for Data and Analytics Strategies."

Figure 3 represents a typical implementation architecture for operationalizing the ML model development life cycle in line with the solution path identified above.

Figure 3. Target Architecture to Operationalize ML for AI



Source: Gartner (October 2018)

The comprehensive, end-to-end ML operationalization architecture (shown in Figure 3) supports the full spectrum of projects outlined within this research. It provides a logical view meant to guide the technical professional toward designing and building a comprehensive ML model development architecture that can be operationalized for the delivery of AI-based systems. This architecture works as a guide regardless of whether the models are deployed within applications supported by language processing systems, computer vision applications, IoT edge systems or embedded within analytical platforms, with their outcomes delivered in the form of predictions on a dashboard or in a forecast report.

While evaluating the individual components or modules of the architecture, focus on the capabilities that can then be translated into specific tools or technologies. It will also present a unique opportunity to look at overlapping/existing capabilities that can be leveraged to facilitate the operationalization efforts within any existing ML and AI architecture.

The logical architecture is divided into four major sections: Acquire, Organize, Analyze and Deliver. The Organize and Analyze sections have been combined, since we will leverage the LDW's infrastructure to support the compute, analysis and training of the ML models.

Acquire

Within this section, you need to build a data acquisition and integration platform from all of the identified data sources:

- **Structured** data sources are composed of the ones generated by enterprise applications and ERPs. Common examples include inventory control, sales transactions, ATM activity and airline reservation systems.
- **Semistructured** data usually consists of semantic tags and does not conform to the structure associated within typical relational databases. Examples include email, XML and JSON files generated typically from customer relationship management (CRM) systems.
- **Unstructured** data cannot be organized in any discernable manner and has no associated data model. This type of information is text-heavy and often includes multiple types of data (audio, images and video). Examples of unstructured data include books, medical records, satellite images, Adobe PDF files, notes in a web form, blogs and text messages. These files are difficult to organize and can be best utilized for analysis by storing them within a file system or an object store.
- **Streaming** data includes a wide variety of data, such as log files generated from a mobile or web application, e-commerce purchases, in-game player activity, social networks, financial trading systems, geospatial services and telemetry from IoT and other connected devices. Based on the use cases, support for the streaming data pipeline is vital for the model-building process by data and citizen data scientists alongside applications and analytical platforms that host the ML models in AI-based systems. Therefore, we see a dotted line (recommended but not absolutely necessary) to the LDW, with a solid data pipeline to analysis by data and citizen data scientists, making it available for applications in a production environment.

Another key architectural component of the Acquire section is the support for data integration, be it extract and load within the LDW or the transformation of the data packets while in motion.

- **LDW** implementation is recommended, since it provides the most efficient way of holding the data, and permits the reuse of data and functions. It allows you to hold combinations of structured and unstructured data, and support real-time and batch processing with a minimum number of elements, thus avoiding unnecessary duplication. The LDW can support the curation and analysis of the data by data scientists and citizen data scientists. It prevents data movement between various storage systems and individual user desktops/laptops. This process lays down the foundation to support both discovery analytics and the ML model training process, thereby inherently enabling a self-service shared data platform to support applications and analytical functions. It is one of the critical components of the architecture, and we will see how to design an effective LDW in Step 2 of the guidance framework.

DataOps engineers can help facilitate the orchestration of the data integration function required to support the acquisition of data. We will look at these in further detail in Step 2.

Organize and Analyze

You should leverage the LDW to organize varied data formats and build the basic components of an ML model development architecture that would support:

- **Feature Engineering,** where steps such as preprocessing, sample selection and the generation of the training dataset take place. This is essential as part of the execution of the ML routines. Feature analysis helps describe the structures inherent within your data being analyzed and selected as part of training variables (see Note 2).
- **Model Engineering,** which includes the data model designs and machine algorithms used in ML data processing (including clustering and training algorithms). Model fitting is one subcomponent, where a set of training data is assigned to a model to make reliable predictions on new or untrained data. This also includes a model evaluation component, where the models are evaluated based on performance and efficacy.
- **Model Execution,** which refers to the environment where the processed and trained data is forwarded for use in the execution of ML routines (such as experimentation, testing and tuning).

We will cover building the ML architecture in Step 3 in further detail. The function of designing the ML architecture is facilitated by the ML architect working closely with the DevOps engineer to build the underlying infrastructure and ML execution framework. Their roles have been covered within the Prework section earlier in this research.

Deliver Models for Inference

Finally, once the models have been built, they need to be integrated within an AI-based system application or an analytics platform, and deployed in production. This is a critical juncture within the framework and where things get a little complicated. Therefore, we will look at it in greater detail within Steps 4 and 5.

Before you deploy the model, you need to build a model management system or solution that can support the following:

- **Model registry** can support versioning of the machine learning models, maintain lineage information on the API calls made to the models stored within the model management system and track model versions in production.
- **Model manifestation** helps create a manifest that encompasses model, dependencies, inference script (aka scoring script), sample data, testing and training data, and the schema. This manifest acts as a recipe to create a container image or provision an API management system on top of it. These manifests could be autogenerated, assist with creating different versions and manage their manifests.
- **Containerization** is the process by which you can use the manifests to build containers like Docker container images for the respective environments. The images provide you with the flexibility to run these images in varied physical and logical compute environments — dev, test, model, prod and disaster recovery (DR).
- **Deployment** is where business-usable results of the ML models or insights are deployed within an enterprise AI-based application, IoT systems or data stores (for example, for analytical reporting and dashboards).
- **Model monitoring** is an essential part of the operationalization of ML model development. As discussed in the earlier sections, this is where ML differs from traditional analytics projects. You need to constantly track the performance of the models created by your teams, take snapshots periodically and find the right ones to build on, retrain, and deploy in production. This would be only possible by capturing the model telemetry and turning them into actionable insights. You will require a monitoring tool that makes it easy to assess the model performance and capture the deployment metrics. We see the feedback lines of the model telemetry from the AI-based system applications and analytical platforms as depicted within Figure 3.

Having a robust model management and monitoring system in place facilitates the operationalizing of the MLDLC, which we will cover in greater detail in the Deliver section of Step 5.

Step 2: Establish the Data Pipeline

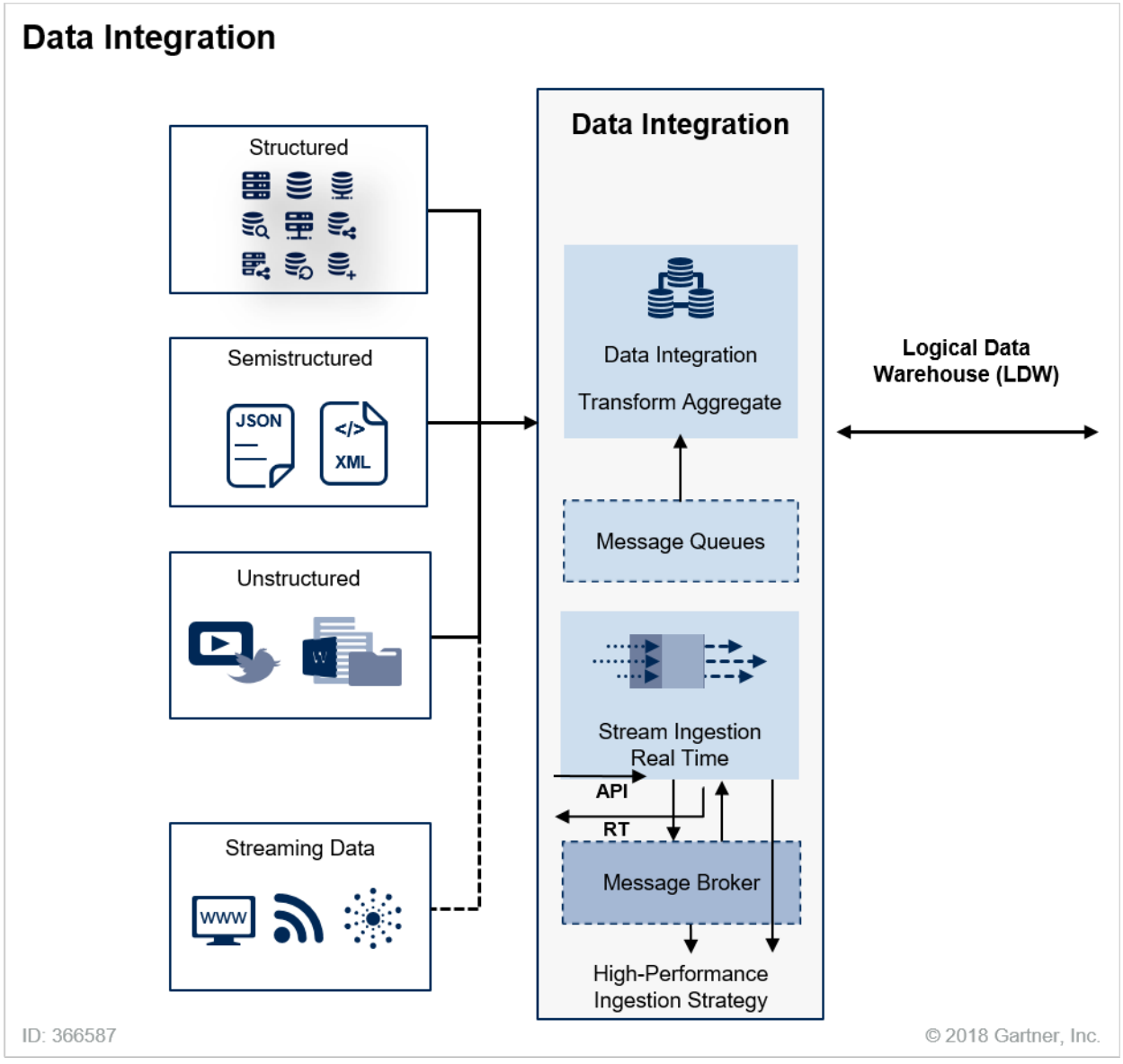
Data is the foundation of almost every AI-based system developed using machine learning. Organizations that are just beginning to explore opportunities to apply advanced analytics, ML or AI should develop a data-centric view of potential applications. It is essential that you build a data pipeline to ingest data from various sources with varying structures into an LDW. In this section, we will focus on developing the first building blocks to acquire the data from various sources and organizing it with the LDW using a data integration tool that can support acquiring, importing, transferring or loading data for model building using machine learning.

Data Integration

The data integration component should facilitate easy access to ML pipelines, supporting real-time, near-real-time, and batch data streams. These pipelines will need to be continuously evaluated and

enhanced to support future requirements. One way to retrofit ML pipelines for the system is to leverage processing engines, such as Apache Spark, or deploying event-based streaming architectures using Apache Kafka to help drive the data ingestion strategy. Apache Spark allows for the development of ML pipelines that provide high-level APIs to help users create and tune those ML data pipelines. This allows developers to execute multiple algorithms in a single pipeline or workflow. For more information on Apache Spark, see The Details section of Gartner's "An Introduction to and Evaluation of Apache Spark for Big Data Architectures." For building streaming architecture platforms, see "Enabling Streaming Architectures for Continuous Data and Events With Kafka" (see Figure 4).

Figure 4. Data Integration



Source: Gartner (October 2018)

The architectural components and logical implementation supporting the ingestion and integration of data for ML is depicted within Figure 4:

- **Batch processing** via an asynchronous method that enables the effective processing of high volumes of data at regular intervals and allows for the computation of arbitrary functions on the dataset, which, in turn, are output in batches.
- **Stream processing** using message brokers like Apache Kafka, Solace and Flume to ingest an infinite, constantly growing data stream that includes unbounded, unordered datasets produced at high velocity. Discrete portions of the stream, known as windows, can be captured through some characteristic of the data, such as event time stamps, and processed using Apache Storm, Spark Streaming, Apache Flink or Kafka Streams, to name a few.
- **Message-oriented** patterns associated with service-oriented architecture (SOA) that require capturing, transforming and delivering data through messages requiring near-real-time processing for event-driven systems can be supported via message queues, RabbitMQ, Apache ActiveMQ and Solace.

Please refer to “Agile Data Quality to Maximize Your Business Results” for further guidance on data quality tools.

Table 1 provides a representative list of some of the data integration tools to consider for implementation.

Table 1. Data Integration Tools

Vendor	Products
Actian	■ DataConnect
Adeptia	■ Adeptia Connect
AWS, Amazon	■ Glue ■ Kinesis
Attunity	■ Compose
Denodo	■ Denodo
Hitachi	■ Vantara
IBM	■ InfoSphere DataStage
Informatica	■ PowerCenter
Information Builders	■ Data Management Platform
Microsoft	■ SQL Server Integration Services (SSIS) ■ Data Factory (Azure Cloud) ■ PolyBase (Azure)
Oracle	■ Data Integrator
SAP	■ SAP
Syncsort	■ DMX
Talend	■ Open Studio
TIBCO Software	■ Data Virtualization

Source: Gartner (October 2018)

Please note that iPaaS and cloud vendors also provide data integration tools, such as AWS Glue and Microsoft Azure Data Factory. For a complete list of data integration and iPaaS tools, please refer to “Magic Quadrant for Data Integration Tools” and “Magic Quadrant for Enterprise Integration Platform as a Service.”

Logical Data Warehouse (LDW)

The LDW can help store active raw data and historical processed and raw datasets alongside curated datasets in the form of cubes and data marts. The key here is to reduce data movement and support the ongoing analysis and model-building process.

For larger companies, Gartner recommends using an LDW to support the ML model development, while for smaller organizations or simpler processing this may not be a necessity. Subsets of the components of the LDW can be used. The full LDW implements a series of components or engines that can handle all of the various data storage and processing needs. This includes structured data using a DBMS, unstructured data using distributed processing (such as Hadoop or Spark). It also provides engines for data virtualization to support agile access to data, as well as streaming. The key is that any requirement can be met by a single engine or a combination of them which are readily available and integrating, thus speeding up development. However, by using the minimum number of engines, you can avoid unnecessary server sprawl and complexity. The organization may begin with a subset the components, sufficient for their immediate needs. Architecting for an LDW and anticipating expansion makes that expansion easier while keeping costs under control, thus providing a better ROI. The LDW is built using a set of relational, NoSQL and data virtualization products that complement each other, thereby providing a centralized data hub for the enterprise to store all of its assets in a governed fashion.

Depending on the number of use cases being considered as part of building the AI system, it may or may not be necessary to bring streaming data sources within the LDW. Hence, it is represented by the reference architecture with a dotted line. We will look at the usability to stream data sources in following sections for discovery analytics and feeding data directly into AI systems.

To implement an LDW, please refer to “Solution Path for Planning and Implementing the Logical Data Warehouse” for further guidance. Depending on the scale of the implementation, the LDW can be built using a single persistent data store as well. You can refer to “Identifying and Selecting the Optimal Persistent Data Store for Big Data Initiatives” for further guidance on selecting the optimal data store to build the LDW.

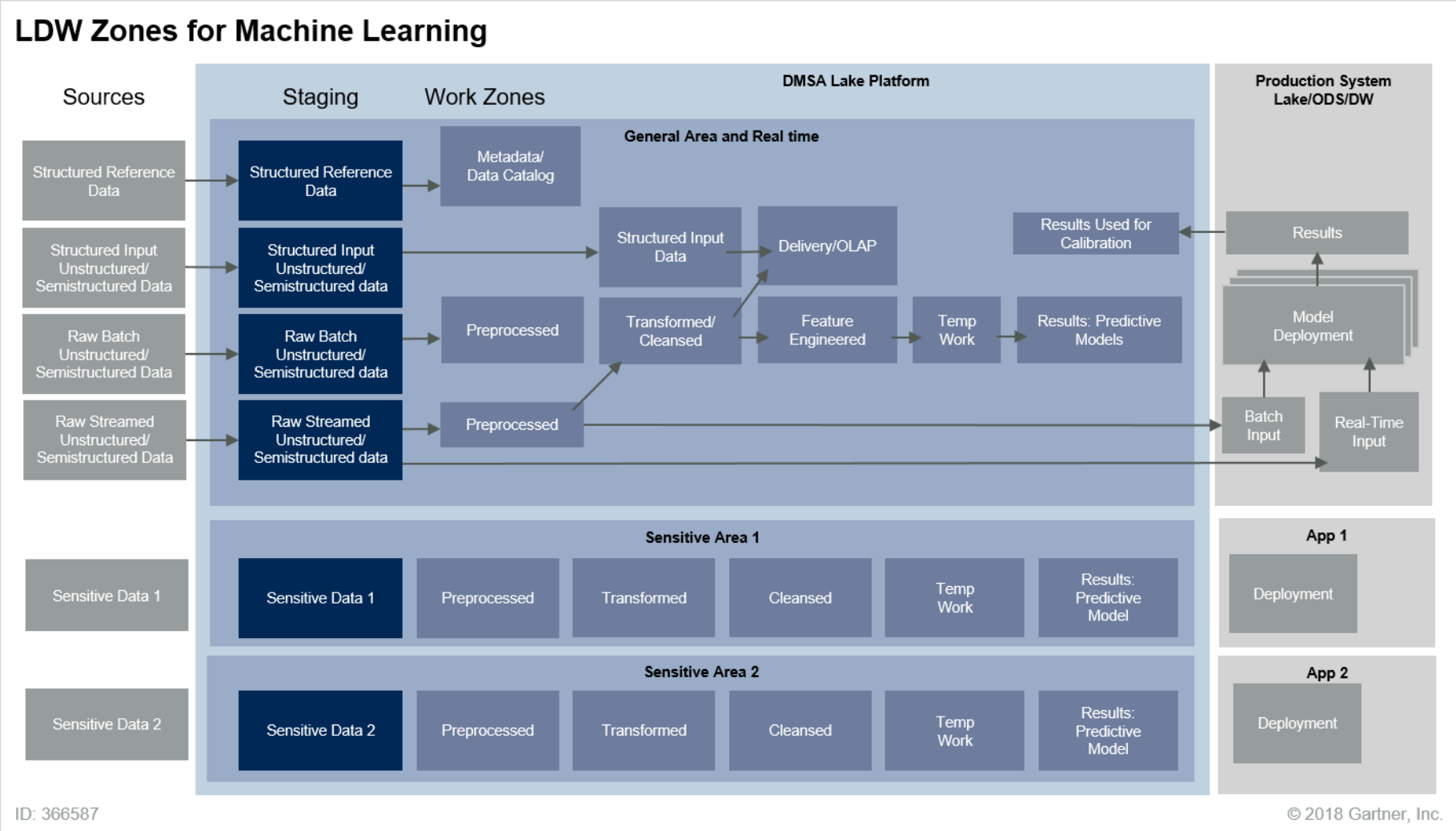
You should store the data in its raw original format. This builds lineage and allows you to go back to the source data for audit purposes, especially in case of the training datasets used to build the models. We hear more and more these days of some models having inherent biases, and this is introduced by the dataset used to build and train the models. Therefore, maintaining a copy of the training dataset is becoming an essential requirement as part of governance of AI-based systems.

When implementing a data lake, it is likely that you will have multiple versions of the data, since the technology for the data lake is usually batch-oriented. However, you are also likely to want to access the data in some kind of a transformed form as well, thereby making the processing easier or helping connect the data to more structured data. These other forms are likely to be a second physical representation of the data produced by either a batch or streaming operation. Hence, the design should support feeding specific areas or zones by using streaming processes as well as batching. Alternatively, zones can be defined using schema on read. Here, data is read and transformed on the fly to create a different version of the data for further consumption. Following

these processes may result in different versions of the data, transformed in different ways. These are different logical zones.

See Figure 5 for an example of designing an LDW for machine learning. Note that the number and size of the zones vary. For example, there might be one formal zone for machine learning. Within this, the data scientists and citizen data scientists may define their own subdivisions, which are not necessarily visible to the outside world.

Figure 5. LDW Zones for Machine Learning



Source: Gartner (October 2018)

The LDW lays down the foundation to support the ML model-building process, and the underlying infrastructure can support the computational processing requirements for building and training the model. See the A Data Science Lab section within “Use Design Patterns to Increase the Value of Your Data Lake” for further information.

Data Cleansing and Metadata

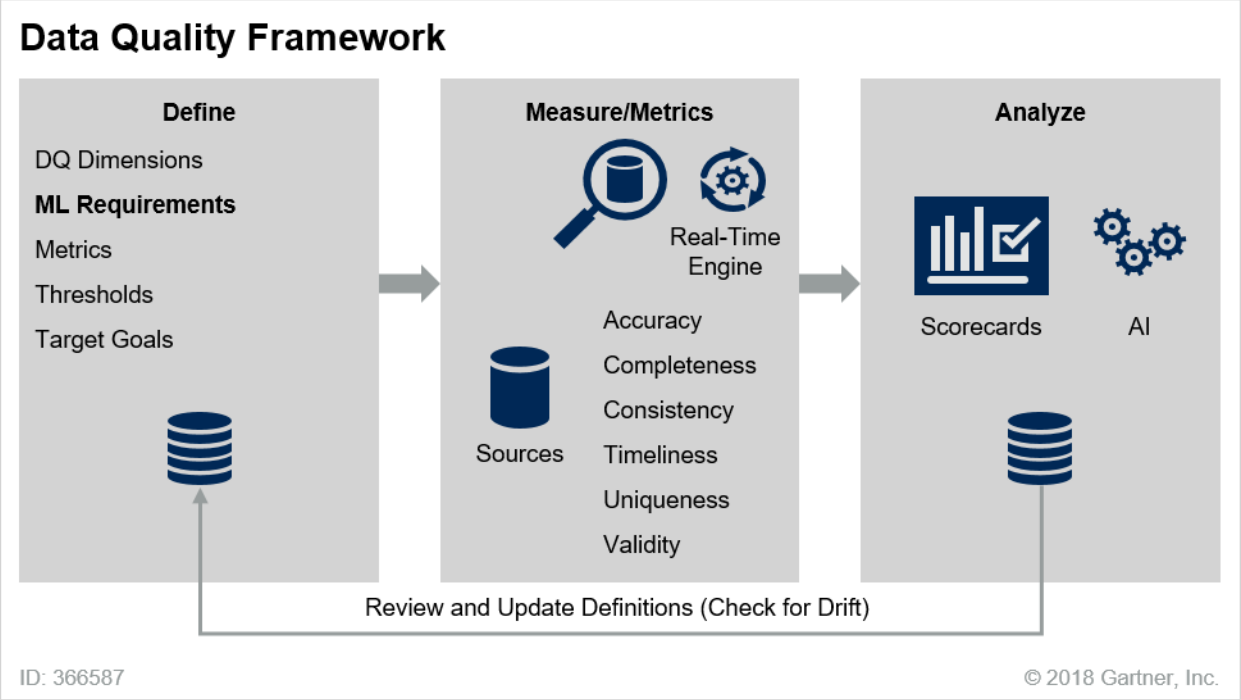
Data Quality

ML models within AI-based systems suffer due to poor data quality caused by disparate datasets, altered raw data and stringent data governance practices. Data quality affects the accuracy of ML processes in multiple ways (for example, using bad data to train supervised models leads to inaccurate outcomes). Also, lack of access to unaltered, unfiltered raw data can impact the efficacy of feature analysis and unsupervised ML models. Hence, it is recommended that technical professionals should analyze and optimize data quality within ML workflows to capture and amend variances within the datasets used for model building.

Implement a process to capture metrics to help identify data drift over time to minimize the impact on deployed AI-based applications or analytical platforms with embedded predictive models. You can leverage existing data quality systems to perform ML preprocessing functions like cleansing and normalization. Creating new data quality pipelines for ML preprocessing can offload work from data scientists and support more citizen data science roles.

The task of effectively implementing and operationalizing data quality initiatives in general is a challenge in the majority of organizations. In the midst of struggling data quality programs, new AI and ML projects are pushing the demands of these data quality processes even further. Hence, ML and AI data quality requirements — specifically, data preprocessing functions like cleansing and normalization — should be identified and incorporated into existing data quality systems. These functions can support both the ML process pipeline and the deployed AI model (see Figure 6).

Figure 6. Data Quality Framework



Source: Gartner (October 2018)

Organizations should target the following key areas to support ML and AI architectures as identified within Figure 6:

- **Define data quality dimensions** to implement data quality functions, such as normalization and other cleansing functions.
- **Measure the thresholds** of data quality metrics as an operational component of ML, which in some cases require cleansing data in near real time.
- **Analyze the scorecards** to validate how well data quality frameworks are cleansing and processing data required within ML pipelines. For deployed models within AI-based systems, more complex scorecards may be required to determine how well the AI model is functioning.

Refer to “Enabling Data Quality for Machine Learning and Artificial Intelligence” for further information on implementing a data quality framework for ML and AI.

Metadata

Metadata is a must for data scientists developing ML models. Otherwise, they would not have context or semantics and wouldn’t know what data to match upon or correlate for hypothesis validation and model building. It can also help compensate the information overload of enterprise data by leveraging the metadata needed to support building AI-based systems. Gartner recommends that you categorize the relevant metadata for ML and AI initiatives as a core

component for operationalizing machine learning. As shown in Figure 7, metadata is generated during multiple phases of the data life cycle (acquire, organize, analyze and deliver). It is also generated during the movement of data between phases as part of the model building, training, evaluation and deployment.

Different users may find only certain subsets of this metadata to be useful. For example, data scientists and citizen data scientists may be interested in metadata generated from within the data lakes or the LDW as part of the hypothesis validation, model building and for the models deployed when retraining/tuning the models. The DevOps and DataOps engineer will gain key insights from metadata generated during AI/ML processes carried out during the discovery, transformation and aggregation activities associated with the organize and analyze phase (see Figure 2). The ML architect and engineer can help derive insights from the metadata associated with model performance and logs generated by the model monitoring tools once they have been deployed in production. All of this metadata in the broader context becomes essential not only to the individuals and roles involved within the operationalization of ML but also to support the EIM and data governance functions within the organization.

To make productive use of this role-based approach, technical professionals must implement an enterprise metadata management discipline, supported and built around the deployment of an enterprise metadata management solution. There are three components that are critical to such an initiative:

- **Capture metadata** as part of building the ML data pipeline.
- **Categorize metadata** into technical, operational, social or business realms for the individual roles involved in the operationalization process.
- **Use the metadata** to link them to specific model use cases and maintain the lineage to enable governance and EIM functions as part of building the AI-based systems.

“Deploying Effective Metadata Management Solutions” covers how to implement a metadata management solution that can facilitate the ML model-building process.

Having designed the core foundational component of a data pipeline to support the ML model development, we now look at building the machine learning architecture.

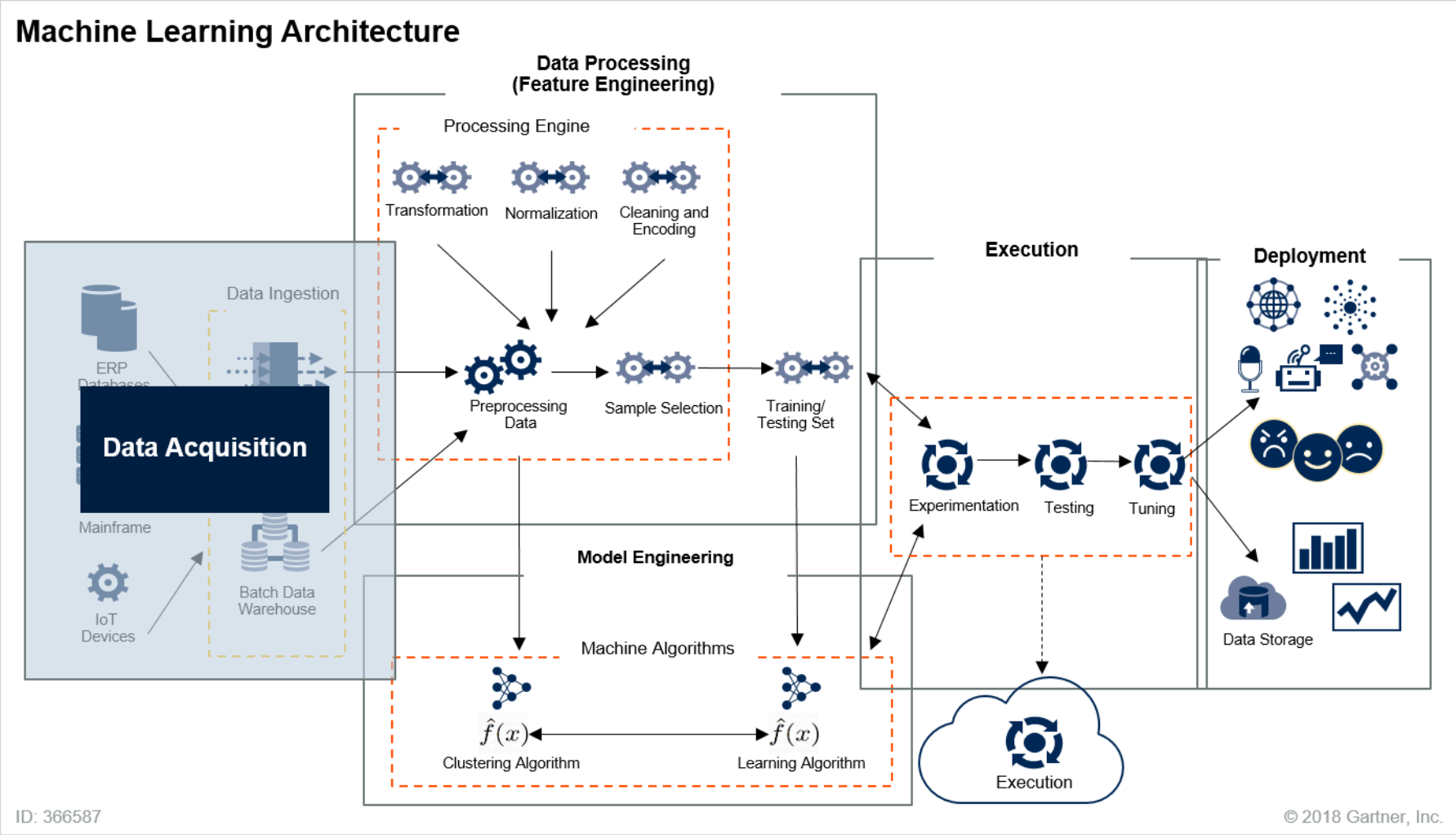
Step 3: Build the Machine Learning Architecture

A typical machine learning architecture includes five functional stages: acquisition, processing, modeling, execution and deployment. Data acquisition and processing functions have been taken care of within Steps 1 and 2, where data is ingested using data integration tools, organized and curated within the LDW. The feature engineering, model engineering and execution is what we will focus on implementing within Step 3. This is where data scientists or citizen data scientists can build models using algorithms that support clustering, regression and random forest with the help of data science platforms.

We need an architecture to build ML models that is flexible and can accommodate the elastic learning patterns as part of building an AI-based system. The underlying infrastructure components

of the LDW can help support the computation. In the case of Hadoop, its distributed computing framework alongside yet another resource manager (YARN) can help support execution engines like Spark and TensorFlow. Cloud-based machine learning as a service (MLaaS) platforms are also an excellent alternative to build the ML workflow due to the elasticity to scale processing and handle high data volumes as needed (see Figure 7).

Figure 7. Architecture of ML Platform



Source: Gartner (October 2018)

Figure 7 shows Gartner's suggested reference architecture for ML. We will focus on the infrastructure areas for processing, modeling, execution (training) and deploying the models as part of the ML workflow.

Data Processing

Data processing includes the steps of preprocessing, sample selection and the training of datasets, in preparation for the execution of ML routines.

This is where the ingested data within the LDW is picked up for advanced integration and processing steps needed to prepare the data for ML execution. This may include modules to perform any upfront data transformation, normalization, cleaning and encoding steps that are necessary for building the model based on the specific AI use case. In addition, if supervised learning is being used, you will need to create a sample dataset that will be used toward training the model.

Feature analysis, or feature engineering (a subset of the data processing component), is a process during which features that describe the structures inherent in your data are analyzed and selected. Much of the data ingested for processing may include variables (referred to as features) that are redundant or irrelevant. Therefore, technical professionals must enable the ability to select and analyze a subset of the data to reduce training time or to simplify the model. In many cases, feature analysis is a part of the sample selection process. To combat privacy and ethical concerns, tech professionals should focus on removing potentially biased features from being used in the model. Hence, feature engineering is an important subcomponent that assists with filtering the data that may violate privacy conditions or promote unethical predictions. Feature engineering is often an underestimated process within the ML workflow because it requires significant human intervention and computational resources. However, new techniques and data science platforms aid in the development of this module. Firms like Feature Labs enable the acceleration of the ML workflow to help data scientists deliver faster training datasets for ML model development.

Model Engineering

Model engineering, or data modeling**,** includes the data model designs and machine algorithms used in ML data processing (including clustering and training algorithms).

The modeling portion of the architecture is where algorithms are selected and adapted to address the problem that will be examined in the execution phase. For example, if the learning application will involve cluster analysis, data clustering algorithms will be part of the ML data model used here. If the learning to be performed is supervised, data training algorithms will be involved as well. Note that algorithms do not need to be built from scratch by your data science team. Many useful libraries of extensible algorithms are available as part of Apache MXNet, Apache Spark MLlib, Caffe, Keras, PyTorch and TensorFlow that can be extended and adapted for your own use. When starting out with ML, experience can be gained by obtaining a few common algorithms — either supervised or unsupervised — from the marketplace and deploying them in the cloud with some data to perform experiments. These may uncover promising avenues for future business value, and may eventually be expanded into formal ML deployments.

As part of the model engineering, you may be need to provide the capability to do model fitting — where a set of training data is assigned to a model to make reliable predictions on new or untrained data and model evaluation. During model fitting, models are evaluated based on performance and efficacy.

Model Execution

Execution is the environment where the processed and training data is forwarded for use in the execution of ML routines (such as A/B testing and tuning).

Depending on how advanced the ML routines are, the performance needed for execution may be significant. Hence, one key consideration in this area is the amount of processing power that will be needed to effectively execute ML routines — whether that infrastructure is hosted on-premises or obtained as a service from a cloud provider. For example, for a relatively simple neural net with only four or five inputs (or "features") in it, the processing could be handled using a regular CPU on a desktop server or laptop computer. However, a net that has numerous features designed to perform advanced, "deep learning" routines will likely need high-throughput computing power on the execution platform. This will take the form of high-performance computing (HPC) clusters or compute kernels executing on high-powered graphics processing units (GPUs). ML and data science teams will often look to test and debug ML models or algorithms prior to deployment. The testing of ML models is typically multidimensional — that is, developers must test for data, proper model fit and proper execution. This can be nontrivial. To combat this challenge, Gartner recommends designing testing environments that mimic production as closely as possible to avoid issues when operationalizing the entire workflow.

Model Deployment

Deployment is where the business-usable results of the ML process — such as models or insights — are deployed to AI-based enterprise applications (such as image processing systems, NLP-based systems, conversational platforms or data stores) and embedded within analytical platforms.

ML output is considered to be similar to any other software application output, and it can be persisted to storage, file, memory or application or looped back to the processing component to be reprocessed, as in the case of IoT implementations. In many cases, ML output is persisted to dashboards that can alert a decision maker and/or recommend the next course of action via a recommendation engine. When operationalizing ML programs, note that the learner becomes an analytics program, similar to any other analytic program you might run in production. In production, the ML system becomes an advanced deterministic query that relies on compute power for execution. Understanding that the deployment of the resulting information, tools or new functionality generated by the ML routine will vary depending on what type of ML model is being used and what value it is intended to generate is critical. Deployed outputs could take the form of reported insights, new models to supplement data analytics applications, or information to be stored or fed into other systems.

Another factor to consider is that the models built using R and Python generally need to be recoded before deployment within production environments built using Java or .NET application frameworks.

PMML, an XML standard for representing analytic models, has made things easier to encode and deploy models (see Note 3).

In the case of AI-based systems you may not be dealing with just one model, rather it could be an ensemble of multiple models. This could get complicated as you operationalize the entire ML model development workflow as part of building an AI-based system. Hence, you need to build a model management system to keep a track of the various models and version history, and maintain the associated metadata to scale this at an enterprise level. We will take a look at this in Step 4.

To dive deeper into the implementation of the ML architecture, please refer to “Preparing and Architecting for Machine Learning: 2018 Update.”

Data scientists and citizen data scientists can be provided with multiple options when it comes to products, technologies and computing frameworks based on their preference. Some of the most popular programming languages, frameworks and techniques used for model building are R, Python, Caffe2, PyTorch, Keras, CNTK, DL4J, TensorFlow and Apache Spark. Most data science platforms support these technologies in a managed software as a service (SaaS) or platform as a service (PaaS) implementation format. These data science platforms consist of machine libraries that support model building, training, testing evaluation and deployment capabilities, and some even provide prebuilt algorithms and models. Table 2 provides a representative list of some of the data science platforms.

Table 2. Data Science and ML Platforms

Vendor	Product Names
Alteryx	■ Alteryx
AWS (cloud only)	■ Amazon Machine Learning ■ SageMaker
Anaconda	■ Anaconda
Databricks	■ Unified Analytics Platform
Dataiku	■ Dataiku
DataRobot	■ DataRobot
Domino Data Lab	■ Domino Data Lab
Google	■ Cloud AutoML ■ Cloud Machine Learning Engine
H2O.ai	■ H20.ai
IBM	■ SPSS ■ Data Science Experience (DSX) ■ Watson Studio
KNIME	■ KNIME
Microsoft — Azure (cloud only)	■ Azure Machine Learning Workbench
SAP	■ Leonardo
SAS	■ SAS
TIBCO	■ Alpine

Source: Gartner (October 2018)

For a complete list of data science and ML platforms, please refer to “Magic Quadrant for Data Science and Machine-Learning Platforms.”

Step 4: Implement a Model Management System

The model management system becomes the key interface module between the Organize and Analyze section of the target architecture and the Deliver section.

As discussed earlier, the ML model development process for building AI-based systems is fundamentally different from a typical analytics project where we deliver a report or a dashboard. Here, we are dealing with the model adding a cognitive element within an application that is making its own decisions and affecting business outcomes. Therefore, tracking the various assets, the data used for building the model, the training datasets, and testing datasets and model versions if it has gone through iterations of retuning become critical metadata for managing and governing and AI implementation. The reproducibility of the model results and outcomes helps establish the bases for audit, compliance, risk assessment and adherence to industry regulatory guidelines.

Hence, when building ML models you have to track the following components at various intervals of the development, release and activation cycles, which we will see in further detail in Step 5.

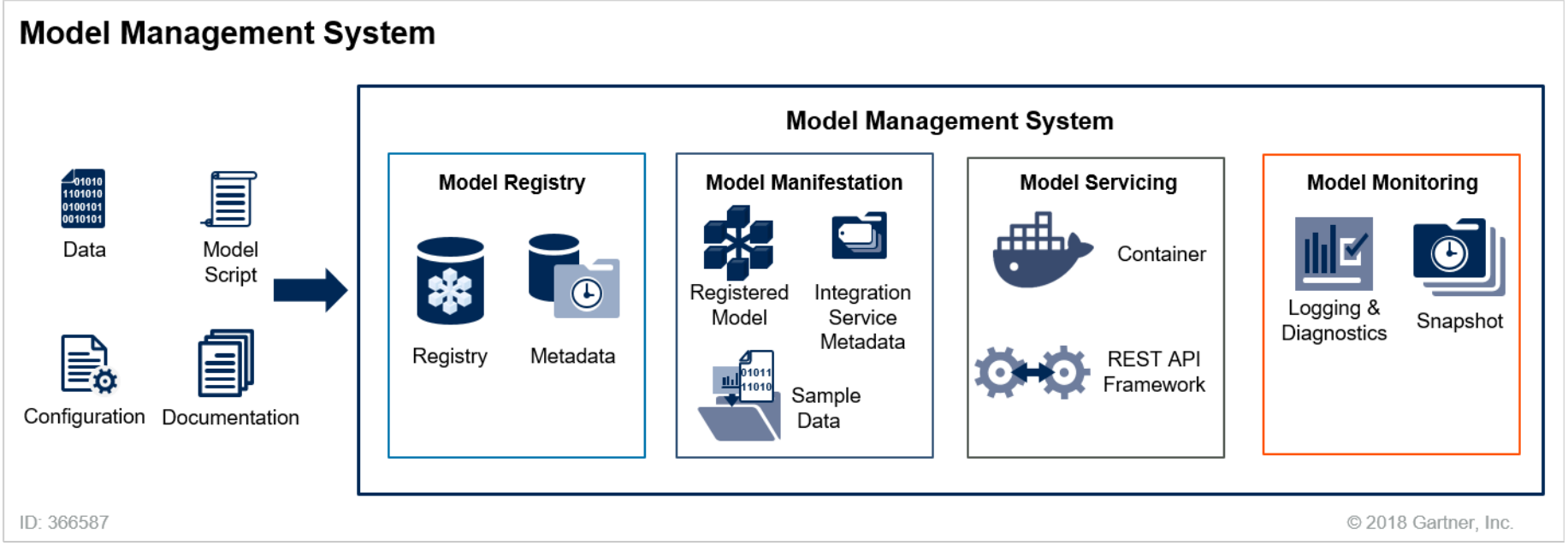
- **Inputs:**
  - Dataset (training and testing)
  - Feature variables (included as well as those excluded as part of the training dataset)
  - Hyperparameter (see Note 4)
  - Code (scripts, such as Python and R)
  - Runtime environments (variables and configurations)
- **Outputs:**
  - Model output (performance metrics against validation dataset)
  - Weights/representation of the model
  - AI-based system — business process outcome (application or analytics dashboard)

By tracking the above components during the various sections of the target architecture, you can evaluate how well the model has been trained based on the defined metrics for the AI-based system. Hence, a model management system should typically provide the following capabilities:

- Versioning of the models
- Tracking models in production
- Providing an API or containerization framework
- Monitoring the models

Let's take a look at the components part of a model management system in further detail in Figure 8 and see how they support the capabilities defined above.

Figure 8. Model Management System



Source: Gartner (October 2018)

Model Registry

The model registry provides the definition of the ML model, and what defines the model are the components it was built using — data dependencies, scripts, configurations and documentation. We also want to capture metadata on the model and its version to provide business context and model-specific information. By providing this definition and structure, we can now keep an inventory of models within the model registry.

The model definition helps with setting the expectations on its behavior in production when integrated within an AI-based system. The registry provides us with:

- Understanding and control over which version of a model is in production.
- Assets and documentation associated with the model, useful when needing to create a new one.
- Release notes explaining changes to the individual versions.

Typically, the model registry module uses the following information:

- Model file or a directory with the model files
- User-created Python/R scripts implemented as part of the model scoring function
- Runtime environment variables and file dependencies
- Schema file for API parameters

Model Manifestation

To deploy or service the results from models, you are required to maintain additional artifacts as part of integration and productionalization of the ML models. The system should provide the capability to create manifests that encompass the model, dependencies, inference script (aka scoring script), sample data and schema. This manifest essentially acts as a recipe to create an API framework or a container image. Some of the data science and MLaaS platforms provide organizations with the ability to create versions, and to autogenerate and manage their ML model manifests as part of the ML workflow.

The model manifest holds the following information:

- Registered model against the AI-based system (application or analytical platform)
- Parameters and configuration of the associated API service or container
- Versions and environment variables (dev, test, model, prod)
- Schema, sample data and dependencies of the scoring script

Model Servicing

Model servicing is where we identify the endpoint interface of the model with the AI-based system. The interface information is what we captured as part of the model's manifest in the previous section. You have multiple options on how you would like to service the model's output or result — containers or a REST API service. You might want to create multiple versions of the models and manifests, depending on the deployment environments (dev, test, model, production). Therefore, the model management system should be able to support the functionality.

The reason for creating individual environment-specific versions is to minimize the risk that the model deployed matches the model developed by:

- Running tests prior to deploying the model to production — release cycle validation (covered in Step 5)
- Capturing environment-specific parameters — versions and library dependencies for the model (for example, a Python or R script requirements .txt file)

You could build Docker-based container images for respective environments where these containerized, Docker-based images can provide you with the flexibility to run these images on the multiple compute environments — on-premises, development environments and even IoT devices. The servicing could vary significantly based on the end system where the model is being integrated.

In the case of an AI-based application, you would want to expose the results of the model via a web service, either via real-time scoring of the model or by exposing scores that were produced offline via a batch process. The model could also support forecasting reports or dashboards showing some predictions for business decision makers and hence require the model output to be stored in a data store.

In either case, without a model registry and a model manifest it can be challenging to understand where to find and consume the results of a current model running in production.

Model Monitoring

Most algorithms are trained and ML models are built using stationary data, but once the model is deployed in production within an AI-based system the application is now dealing streaming data that isn't stationary. What occurs over time is the model starts showing drift or degradation in the scoring and performance by losing its precision and prediction quality.

When it comes to monitoring ML models, even the simplest ML systems consist of multiple moving parts — data aggregation, computing framework and infrastructure, and integration framework (such as web services.) Hence, monitoring models could require gluing together many pieces of tools and technologies to capture logs (such as data, compute cluster and model telemetry), analyze the data and visualize it for performance evaluation and alerting. You should consider the following metrics as part of monitoring the ML models:

- The same parameters and metrics you used to evaluate the performance of your model during the development cycle, its accuracy, precision, recall and RMSE (see Note 5). You could plot a chart using the time series data being captured, and evaluate if it is still performing based on

the identified thresholds. Benchmark it against the performance of the training and test datasets.

- Since the model is integrated within an AI-based system, which has shared computing resources, you should monitor data aggregation runtimes, model runtimes and success rates of models runs over the past period.

It is recommended that you collect the monitoring data (whether precision, sensitivity, specificity and recall are within the thresholds) and ideally bring it back to the LDW to ensure historical storage of model performance behavior and provenance for evaluation and audit purposes. This dataset also helps with the analysis of when a particular model requires retuning to meet business KPIs and ensure there is no bias, drift or degradation over a period of time. This function of monitoring and retuning is supported by a combined effort of the ML engineer/architect, data scientist and the business application owner.

Products and Tools

Most data science platforms covered within Table 2 provide a model management system, thereby facilitating the operationalization of ML development for AI-based systems.

However, below is a list of specialized tools and products that focus purely on operationalizing ML workflows with embedded modules with model management and model monitoring (see Table 3).

Table 3. Model Management Tools

Vendor	Product Names
Aginity	Aginity
Algorithmia	Algorithmia
Anodot	AI Analytics
Domino Data Lab	Domino Data Lab
Hydrosphere.io	Hydrosphere.io
ParallelM	MCenter

Source: Gartner (October 2018)

Step 5: Operationalize the Model-Building Process

We now come to the final piece of the architecture, where we operationalize the individual architectural pieces discussed so far by following the CRISP-DM methodology.

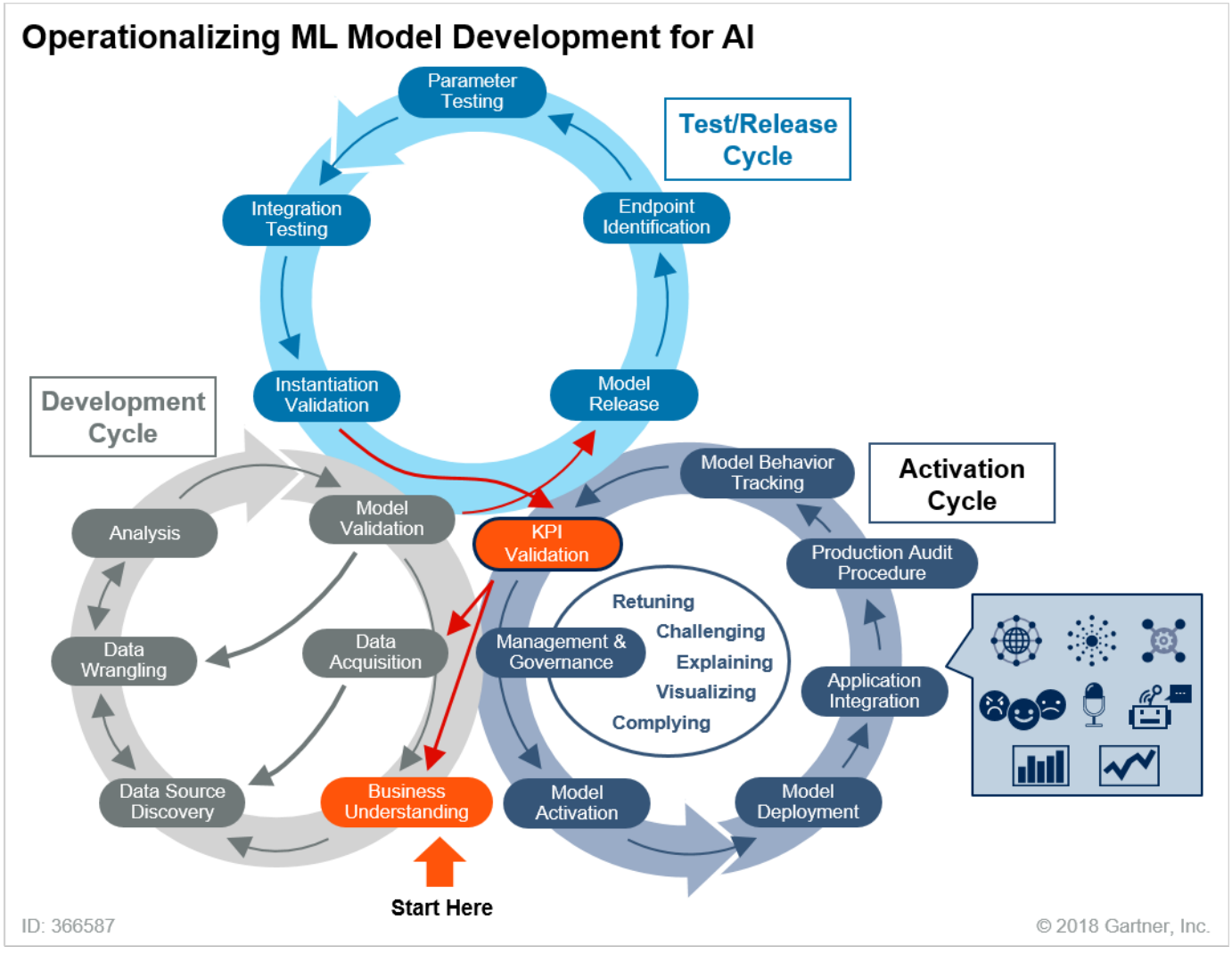
The ML model development life cycle has been derived using the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, which has been used by data science practitioners for

two decades (see Note 6). The methodology, simple and powerful, imposes a discipline that promotes integrity and robustness, resulting in establishing an ML model development life cycle that is composed of three main cycles — development, release and activation.

> ML model development is far more glamorous than model operationalization, but it is the quintessential part to delivering continuous business value with AI.

The three cycles are illustrated in Figure 9.

Figure 9. Operationalizing ML Model Development for AI



Source: Gartner (October 2018)

Development Cycle

The architectural components required to operate the MLDLC were designed and implemented within the sections of Prework and Steps 2 and 3 of the guidance framework.

We built the foundational components facilitating data acquisition and staging it within the LDW for data discovery in Step 2. This is where the data pipeline for ML model development was established. The data preparation and analysis capabilities were designed and built as part of the data processing (feature engineering) and model engineering sections within Step 3. Once the model has been validated against the testing dataset created as part of the model development process, it is introduced into the first operationalization cycle, which is release. The red arrow from model validation to model release represents the transition of the process from the development cycle to the release cycle.

Test/Release Cycle

The test/release cycle, is where the published model is set free inside a guarded perimeter to verify that the assumptions made while developing the model still hold true in the real world of the actual business process. This is to ensure that the model has been built as part of the specification. Within this cycle the model is integrated within the intended AI-based application or the analytics platform and tested in a model or test environment before releasing it into production. You can draw parallels to the application release process laid out as part of the software development life cycle (SDLC).

We have identified six steps as part of the release cycle:

- **Model release.** The model is promoted to the release step, ready to be undertaken by the operationalization team and labeled as a “candidate” model (that is, development-vetted, but not yet fully production-ready). It is registered within the registry of the model management system.
- **Endpoint identification.** This is the validation of the decision points where the model will be delivering its insight. Those analytics endpoints could be within an existing application, integrated as a single model, as part of an ensemble of models and in some cases as an input to another decision modeling mechanism (like a business rule). This implementation is usually in the form of a REST API or container images, depending on the deployment. The information pertaining to the endpoint identification is captured within the model management systems manifest. These endpoints serve as integration points of the models for AI-based systems (for example, Chatbot, VPA, image classification systems, streaming video analysis applications, and IoT sensors of IoT edge systems).
- **Parameter testing.** Target business processes might be subject to technical constraints where the velocity, shape, volume and quality of the input data in the production environment might not exactly align with the data in sandboxes used to develop the models. This step is aimed at testing the alignment of the data once the model is part of a real-world application, dashboard or an analytics report.

- **Integration testing.** Provided that the expected data matches the development assumptions, integration assumptions (that is, REST APIs, microservices call and code integration) also have to be tested to ensure proper performance.
- **Instantiation validation.** As models in production are often part of model ensembles, even slight variations in those elemental models (such as a propensity to buy models instantiated across multiple states or regions in the same country) can produce radically different results.
- **KPI validation.** Model performance should be measured not only against technical parameters (such as precision) but also against the agreed-upon business KPIs (measurable business outcomes) set forth as part of the business understanding step indicated earlier within the Ideation section of the Prework Step.

Depending on the outcome of the KPI validation step, three paths are possible:

- The model fails due to insufficient data or ill-calibrated assumptions about that data. In this case, the process can proceed back to the "new data acquisition" step in the development process.
- The model fails to deliver the appropriate business outcome, and in this case it might be wise to re-evaluate the original business assumptions through the "business understanding" step back at the beginning of the development cycle.
- If you are lucky, the model delivers as promised and is ready to move to Phase 2 of the operationalization cycle: the activation phase.

The intersection points and the flow of the process based on the outcome has been denoted by red arrows in Figure 8.

Most data science platforms discussed in Table 2 support the steps laid out within the release cycle phase. The model management system discussed within Step 4 plays a quintessential role as part of managing the artifacts and the metadata relating to the ML model.

Activation Cycle

Now that the model has been tested and is performing as intended, in a test environment after integration with its intended endpoint AI-based system services, it is ready for prime time. The goal is to activate that model within existing business processes across the organization at the endpoints identified in Phase 1 of the development process.

From this point, the models remain in the operationalization cycle as long as they are performant, meet attribution thresholds and deliver business value derived from the models in production. As long as models are performing based on the KPIs defined, they keep their place in the production cycle. Otherwise, management rules are in place to re-evaluate those models in the light of changing circumstances captured in the previous operationalization steps (release and development). Those models are then sent back into the development cycle process, as illustrated with the red arrows at the intersection point.

Within the activation cycle, we have identified seven steps to fully operationalize the model development process:

- **Management and governance.** Once a model is ready for activation, it should be cataloged, documented and versioned (including the original set of variables that has been used to develop it). The model management system should act as that single-point management and governing module. We have looked at its implementation in Step 4. This model should also be subjected to the governance rules adopted by the data science team, production operations team and the business partner. The business is the owner of the model, similar to how it owns the data that was used to build and train it in the first place. Management rules should also be established to control the exchanges between the production and the development teams as part of change control (such as when to retune models, expose challenger models and levels of model explanations).
- **Model activation.** This step is the hand-off of the models that were validated during the operationalization process to the production team as activated models. At this stage, the models are "production ready," fully documented and compliant with the defined governance rules.
- **Model deployment.** Depending on how those models will be executed (on-premises, in the cloud or both, leveraging streaming infrastructures and parallel processing mechanisms like Spark), measures need to be taken to guarantee the smooth processing of the transactions leveraging the model (or the ensemble the activated model is part of).
- **Application integration.** The model is now part of an AI-Based system (NLP engine, chatbot framework, VPA, image classification application, IoT sensors, or Edge devices and/or analytical platforms embedded within dashboards or reports). Model insights are delivered in the form of a score or recommendation, the majority of which may be part of an existing application. An element of extra coding is at times necessary to properly embed a model, or its input, within an application to enhance a business outcome. This is where the application developer or BI developer comes into play to integrate the model within a production application or analytical platform. The model will finally deliver its business value, and the applications act as the end-user interface points as illustrated in Figure 3 of the target architecture.
- **Production audit procedures.** Model telemetry has to be implemented to gather the necessary data to monitor models in production. Various performance metrics in terms of accuracy, response time, input data variations and infrastructure performance instrumentation, including possible implementation of software agents, also have to be implemented. This is a crucial step within the activation cycle and as part of the operationalization process. Various instruments (such A/B testing methods, multivariate testing, multiarmed bandits (MABs) and shadow models) can be implemented to judiciously appreciate the performance of models and test them against the KPIs initially defined by the business.
- **Model behavior tracking.** Alerts and monitoring methods have to be established to track the performance of models in production. Performance thresholds and notification mechanisms are implemented in this, and the previous step helps to systematically flag any divergence or suspicious behavior.

- **KPI validation.** Extended from the development cycle and fed by the two previous steps, the KPI validation step consistently measures the business contribution of the models (or ensemble models) in production AI-based systems. The idea is to get a precise business value that can be attributed to the model. To determine that value, data from the production audit procedures step should prove invaluable.

The operationalization cycle for ML model development has never been formalized, but successful organizations have adopted various tools and technologies to provide a framework to manage operational decision services in a reliable, repeatable, scalable and secured way. Hence the framework we discussed as part of the model development life cycle above provides:

- **Catalog.** A "centralized" way to store and secure analytical assets to make it easier for analysts to collaborate and to allow them to reuse or exchange models or other assets as needed. (It could be a secured community or a collaboration space.) In our case, this is supported by the LDW (Step 2 — establish a data pipeline), storing all of the datasets in the form of a centralized data hub. This centralized data hub facilitates collaboration and the exchange of datasets within its various stages of transformation as part of the model development process. The metadata captured provides the lineage within the LDW.
- **Capabilities.** Automated versioning, fine-grained traceable model scoring and change management capabilities (including champion/challenging features) to closely test, monitor and audit analytical asset life cycles from a technical as well as a business performance perspective (through key performance indicators [KPIs]). The model management system within the target architecture (Step 4 — implement a model management system) supports the storage and manifestation of the statistical models built by various teams and individuals.
- **Coherence.** Establish simple protocols to provide functional bridges between the development and operationalization cycles (release and activation). Enhance cooperation and consultations between the development and operationalization teams. Provide efficient "translation" services between data science and lines of business, and improve the transparency of deployed analytical assets.
- **Governance.** Protocols to ensure adherence to all internal and external procedures and regulations, not just for compliance reasons but, as an increasing amount of data and models get built and deployed and address any potential privacy issues as part of the operationalization of ML for building AI-based systems.

Follow-Up

The guidance framework and target architecture shows how enterprises need to design and integrate the building blocks requirement to support the continuous delivery, monitoring and retraining of ML models for delivering AI-based systems. There are, certainly, different implementation models — on-premises, cloud or hybrid. However, the underlying architecture and operationalization framework remain the same. It is impossible to drive impactful business transformation with AI without setting a foundation and tackling challenges with data processing and establishing an ML model development life cycle.

Hence, it is important that you follow the best practices laid out within the guidance framework:

- Identify and prioritize use cases and put together a project management portfolio alongside a technical execution team.
- Define roles and responsibilities of data scientists, citizen data scientists (domain experts), DevOps/DataOps engineers, machine learning architects, solution architects, BI developers and application developers.
- Use the LDW approach to reduce data movement during the model build, train, evaluation and retraining phases. The LDW can serve as a single point hub for the model telemetry and metrics collected for optimization, audit and compliance purposes.
- Implementation of the model management system is key toward operationalizing the backbone for continuous delivery of an AI system. It is critical to ensure that there is a process in place for versioning and promoting a given model through various environments.
- Constantly monitor and measure the efficiency of the ML model and the overall application they are part of to ensure optimal performance and precision of the results.
- Establish a change control management system similar to the one followed for application development for ML models.
- Adherence to the management and governing principles of the operationalization cycle is critical for establishing a factory model for ML within the organization. This is important even from a data science perspective to support analytical functions and use cases.
- Keep the architecture modular enough to ensure the introduction of new tools and products that can assist with capabilities toward simplifying the ML model building and management process.

The key findings gathered during the design, build and run stages should be revisited to look for opportunities on further optimizing the architecture or introduction of new infrastructure, computing frameworks and/or development software development kits (SDKs).

Risks and Pitfalls

Organizations undertaking ML and AI initiatives face several risks and potential pitfalls for their project.

Spaghetti Bowl of ETL Scripts

Devoting insufficient time and resources to data preparation and building a consistent data pipeline to support ML can lead to a spaghetti bowl of ETL scripts, resulting in an operational nightmare. Too many data acquisition and transformation scripts with overlapping business rules lead to duplication of effort in the operations part of IT. There is inconsistency in training and validating datasets, and the implementation of a governance framework for audit and lineage is at risk.

The implementation of an LDW and building a consistent data pipeline is foundational to operationalizing model development using ML for AI projects. It is recommended to leverage a data integration tool for managing the ETL scripts, and to limit the number of tools used for data

acquisition. Data engineers and DataOps individuals should act as gatekeepers for building the data pipelines, while data scientists and citizen data scientists should be provided with the appropriate wrangling and data science tools, giving them the flexibility to transform the data, but in a governed self-service environment. Any production ETL data pipelines should be validated maintained and monitored by the DevOps and DataOps teams.

Incoherent Technical Team

As discussed in the Prework section, you will require a diverse set of technical skills to deal with the array of tools across multiple functional categories to build and operationalize the MLDLC. For some organizations, this may be a major challenge, but in most cases you can circumvent it by leveraging existing in-house resources. Building a consistent pipeline for model development is more about orchestrating the tasks and processes between the integration points: data acquisition, model development, training, management and integration within AI-based systems. Hence, you need a team with the right mix of technical skill sets that can manage the orchestration.

Apart from using various tools for automation, the scripting efforts require skills that are not always possessed by data scientists or data engineers. A plan to supplement the data science team with support from DevOps, ML engineers and architects, and exposure to workflow management tools, will help develop a consistent process for building AI-based systems in a consistent fashion. Plan for the right people to receive training and experience in the tools, and ensure that understanding of the overall workflow is a major part of the implementation process.

DIY Data Science

Not having a data science platform or ML architecture for the model development process in a consistent manner is a big risk to delivering and managing AI-based implementations.

> DIY data science is not scalable.

Data scientists usually have their own preferences on the language, set of libraries and execution frameworks they would like to leverage to build models. Hence, they may use a wide variety of ML libraries and at times pretrained models for building an ensemble of models for implementation.

Therefore, it is required that you build a scalable ML architecture leveraging a data science platform that brings structure to the model development process while providing the data scientists the autonomy they require. The ML and data science platforms discussed in Step 3 usually support an array of scripting and developing languages and provide features and functionalities to manage ML libraries, code versioning and dependences.

Inconsistent Provisioning of Models

It's hard to track model output and results coming out of a development cycle unless the deployment or provisioning of the models have a consistent framework. Failing to catalog models and interpret their outputs across multiple AI-based system implementations can lead to an

operational nightmare. Ignoring this pitfall results in loss of reproducibility and the ability to have sustainable business impact and improvements going forward.

Therefore, you need to build a deployment framework discussed in Step 4 to deploy common model types on diverse platforms as part of the model management system. Leverage functions within MLaaS and data science platforms to deploy either as Docker containers and/or API REST services. Otherwise, implement an API gateway to track access to the published models via API.

Failing to Track and Monitor Models

When models are not monitored or managed over time we usually see degradation and drift set in from a business KPI perspective. ML algorithms and models don't get better with age and need to be consistently monitored against the parameters and thresholds identified. Most ML algorithms are built using stationary data, and the reality is that streaming data is rarely stationary, which is what it deals with when integrated within an AI-based system. Models tend to suffer from loss of predictive quality and precision, which results in model drift and degradation.

Hence, ML models need to be periodically retrained with new data, and in order to do that they need to be consistently monitored. The changes in the input data and the quality of predictions need to be monitored, and the cost of inaccurate predictions is to be included within the variables as part of the KPI analysis. A better alternative is monitoring the model quality by testing the inputs and predictions for changes over time and using change points in retraining decisions. Implement model monitoring systems, and collect and monitor logs generated to ensure model and AI-based system performance within optimal limits.

Conclusion

Create a coherent MLDLC workflow that provides a structured approach to deal with the proliferation of new tools and exchange of data and model assets across multiple environments. To execute a successful AI strategy, you should consider implementing a consistent model development and operationalization framework, which is core to the entire initiative. Successful MLDLC requires the orchestration of multiple, disparate tools and the skills to integrate them while managing it across a complex environment. Therefore, it requires a systematic approach. The aim should be to minimize data movement and a robust scalable architecture with the right set of tools and skill sets toward fulfilling the AI requirements of your organization.

Gartner Recommended Reading

Some documents may not be available as part of your current Gartner subscription.

"A Framework for Applying AI in the Enterprise"

"Driving an Effective AI Strategy"

"How to Operationalize Machine Learning and Data Science Projects"

“Preparing and Architecting for Machine Learning: 2018 Update”

“Integrating Machine Learning Into Your Application Architecture”

“Solution Path for Planning and Implementing the Logical Data Warehouse”

“Identifying and Selecting the Optimal Persistent Data Store for Big Data Initiatives”

“Use Design Patterns to Increase the Value of Your Data Lake”

“An Introduction to and Evaluation of Apache Spark for Big Data Architectures”

“Enabling Streaming Architectures for Continuous Data and Events With Kafka”

“Enabling Data Quality for Machine Learning and Artificial Intelligence”

“Deploying Effective Metadata Management Solutions”

“Staffing Data Science Teams: Map Capabilities to Key Roles”

Note 1 Model Drift

Concept drift or model drift is identified as a change over time in the precision and quality of the predicting capabilities of the models. This causes problems because the predictions become less accurate as time passes and there’s more exposure to varying streaming data in a production environment.

Note 2 Feature Variables

Feature variables are a subset of relevant attributes (variables, predictors) used in statistical and machine learning model construction. Feature variables assist with simplifying the models for interpretation, facilitate shorter training times, avoid the dimensionality curse and enhance generalization by reducing overfitting and reducing the variance.

Note 3 PMML

PMML stands for “Predictive Model Markup Language.” It is the de facto standard to represent predictive models and solutions. A PMML file may contain a myriad of data transformations (pre- and post-processing) as well as one or more predictive models. PMML allows for different statistical and data mining tools to speak the same language. In this way, a predictive solution can be easily moved among different tools and applications without the need for custom coding. For example, it may be developed in one application and directly deployed on another.

Note 4 Hyper-Parameter

A model hyper-parameter is a configuration variable that is external to the model and whose value cannot be estimated from data and specified by the data scientist as part of the model training process. They are used in processes to help estimate model parameters and tuned for a given predictive modeling problem.

Note 5 RMSE

The root-mean-square deviation or root-mean-square error is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed. The RMSD represents the square root of the second sample moment of the differences between predicted values and observed values, or the quadratic mean of these differences.
These deviations are called residuals when the calculations are performed over the data sample that was used for estimation, and are called errors (or prediction errors) when computed out of sample.

Note 6 CRISP-DM

Cross-industry standard process for data mining, known as CRISP-DM, is an open standard process model that describes a structured approach to planning a data mining project. The model is an idealized sequence of events where many of the tasks may be performed in a different order, repeated and might often require a feedback loop but provides all possible routes to structure the data mining process.

GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp