How to Create a Data Strategy for Machine Learning-Powered Artificial Intelligence

Published 31 May 2017 - ID G00324342 - 51 min read

By Analysts Carlton E. Sapp,

Supporting Key Initiative is Analytics and BI Strategies

MLpAI can help deliver systems with more automation and less human intervention, but success requires a data strategy to deal with the complexity of real-world data. This research guides technical professionals involved in MLpAI on developing a data strategy to support successful deployments.

Overview

Key Findings

- In MLpAI initiatives, most of the time is spent on designing and managing the data, and not on the algorithms applied. Failing to focus on extracting relevant attributes or on cleansing the data will decrease the overall effectiveness and significantly increase the probability of error.
- MLpAI systems are complex systems that feature many interacting data components. The data strategy must employ reusable, general-purpose components with well-understood parts.
- Cloud providers (CPs) offering machine learning as a service (MLaaS) or artificial intelligence as API services complement data strategies, but they do not eliminate the need for a data strategy. Many CPs offer pretrained models and vended data to help accelerate MLpAI projects.
- Data strategy for MLpAI is an effort that can overlap with and be complementary to overall enterprise data strategy. Aligning overlapping areas significantly improves the rate of success.

Recommendations

Technical professionals tasked with delivering effective MLpAI capabilities must:

- Build your MLpAI data strategy by focusing on data processing requirements of the ML tasks. Gartner clients struggle when they focus only on the technology solutions.
- Classify the different data processing requirements within the strategy by the desired learning capability to aid in organizing and reusing critical components. Many of our clients deploying MLpAI waste a great deal of time or effort in creating workflows that already exists.
- Avoid designing individual training workflows on-premises when you don't have proper staff to support them. Take advantage of cloud environments for training. The most computationally and data-intensive part of the workflow is the training stage.
- Insert human-driven domain knowledge experts early in the process when developing the data strategy to aid with the testing and evaluation (T&E) of MLpAI. T&E is a growing concern with our clients deploying MLpAI.

Problem Statement

Advances in technology to capture and process large amounts of information mean we're still drowning in data and will continue to do so. Moreover, we still lack the ability to extract understanding and insight from data at the rate in which we're receiving new data. That

is where machine learning-powered artificial intelligence (MLpAI) comes to the rescue. But MLpAI can only rescue us if we build an effective strategy for using data to support it. Lack of planning or strategy for the data can lead to costly delays and overextension of resources.

A data strategy for MLpAI is a component of the enterprise data strategy. It is not a replacement for an enterprise data strategy; rather, it is an extension of the enterprise data strategy to support the citizen data scientists (CDSs) whose requirements may vary from the traditional data and analytics professional. MLpAI can be enveloped in an enterprise data strategy, but it should be a distinctive process during design in order to support the nontrivial tasks of preprocessing and postprocessing of data.

Introducing MLpAI and Its Limitations

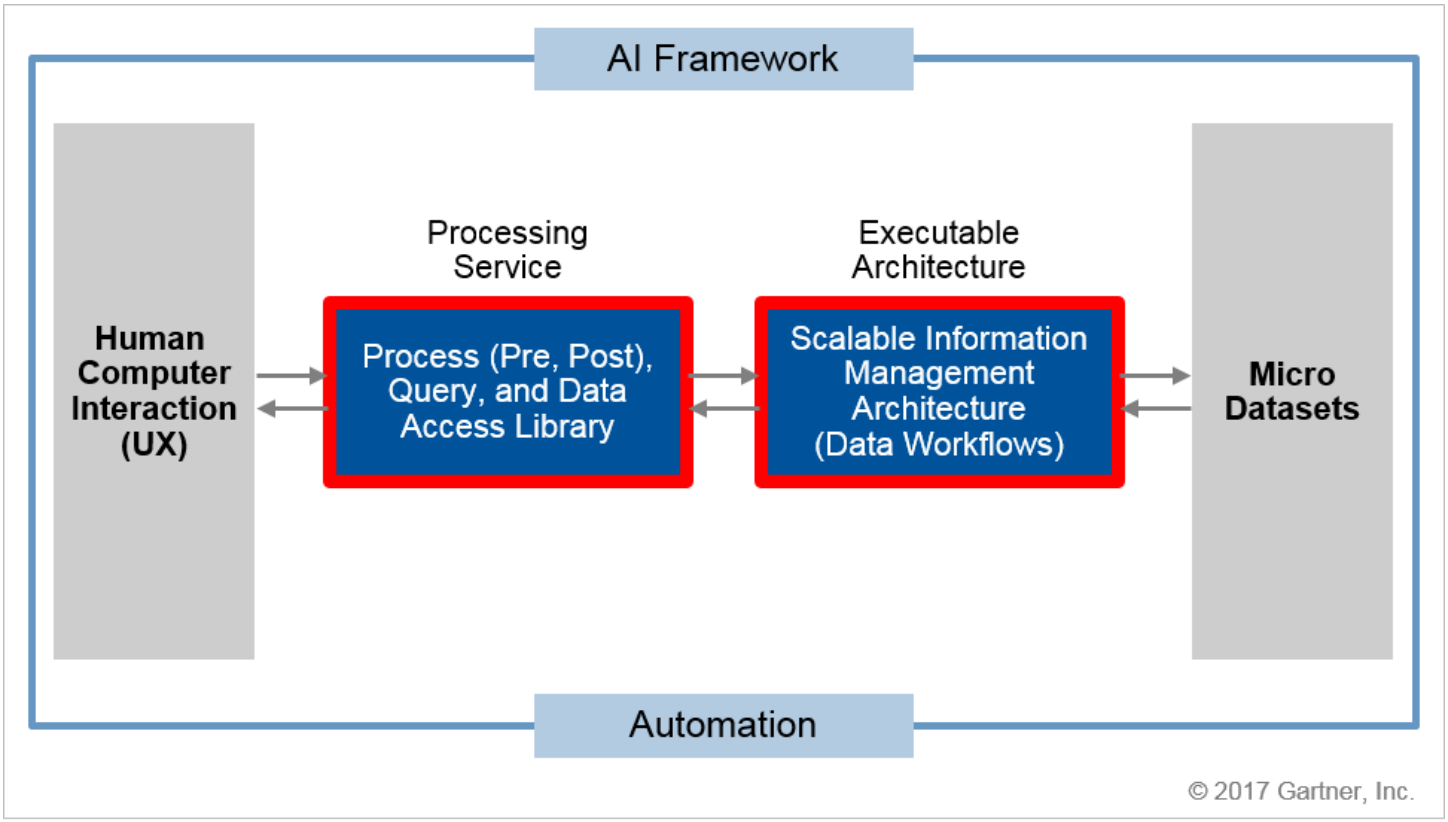
> **Machine learning powers artificial intelligence with advanced learning and cognitive algorithms, but better data continues to beat better algorithms.**

Gartner expects artificial intelligence (AI) to be the next disruptive game changer, not just in the technology changes, but also in the operational changes it enables. These systems employ machine-learning techniques to achieve new levels of performance and insight. [1] (#dv_1_artificial_intelligence) Machine Learning (ML) consists of advanced techniques that are founded on statistics, probability theory, optimization and decision analysis that can amalgamate to perform a specific task. The biggest impact of MLpAI is the ability to accelerate automation and decision making at scale for the business.

However, in order to accelerate automation and decision making at scale using MLpAI, data must be managed to support the complexity in processing for MLpAI and its executable architecture. ML powers AI by providing computationally and data-intensive processing and architecture, as shown in Figure 1. Moreover, the process of managing data through the stages of building MLpAI is often complex and underestimated due to the level of uncertainty in each stage and the nonlinear behavior of the data required to support the citizen data scientist. For example, when building MLpAI, much of the data being acquired must be explored for dependencies. In other words, we can't treat streams of data as if they are independent and have nothing to do with each other because there are "known unknowns" and "unknown unknowns" that we must explore.

Figure 1 describes an example AI framework that enables ML techniques and demonstrates how ML can power AI and be bound by the information architecture. There are limitations to applied AI based on the information architecture. Overlooking the information architecture (as outlined in red in Figure 14) may significantly impact the output of the intended solution. This research primarily focuses on the components outlined in Figure 1. The components outlined in red are the key drivers to developing a data strategy to support the AI capability.

Figure 1. Machine Learning-Powered Artificial Intelligence Framework



Source: Gartner (May 2017)

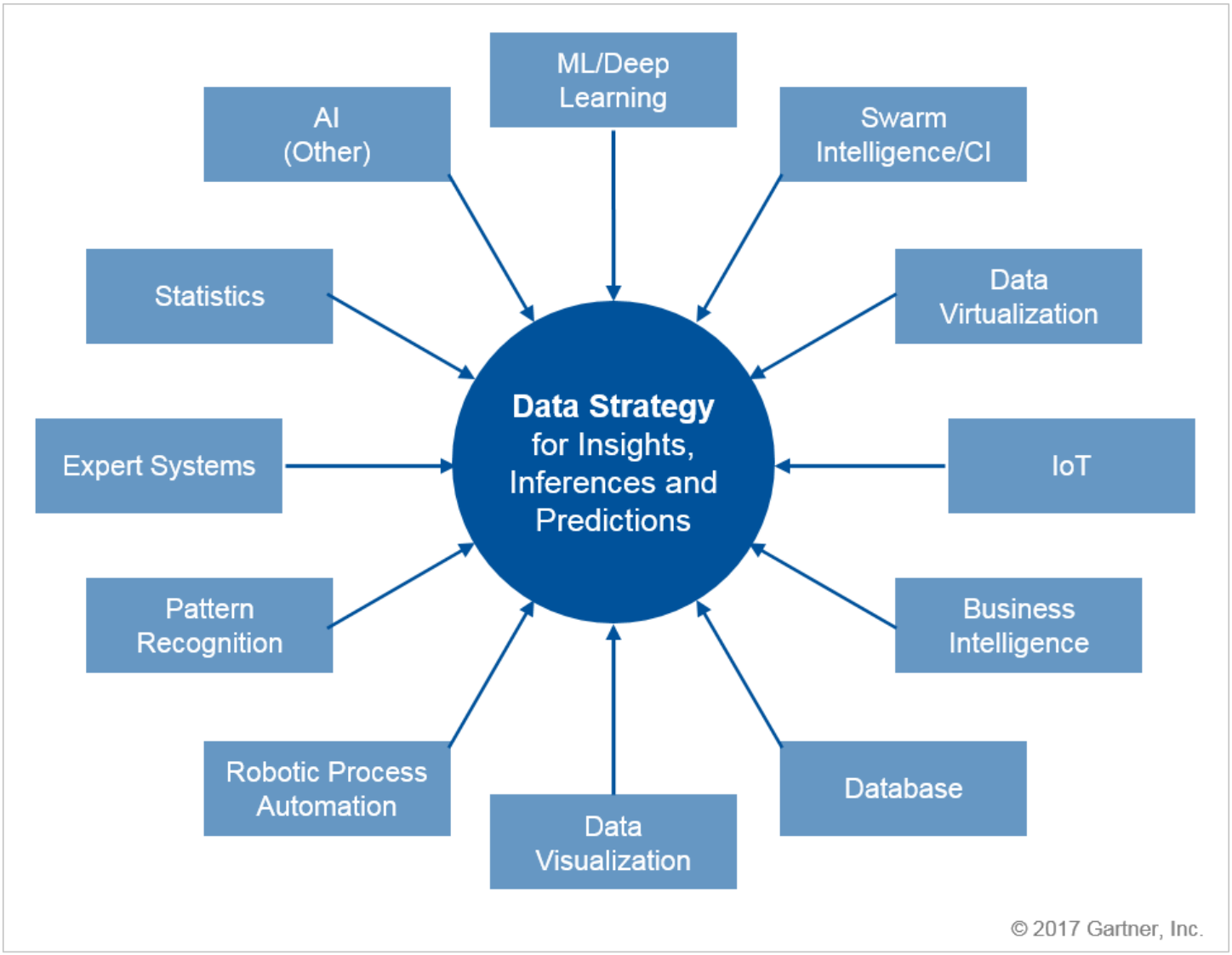
MLpAI systems are capable of analyzing large amounts of complex data, but how we acquire and process data used by ML techniques, along with the quality of that data, will vary and can be nontrivial. More importantly, data scientists are frustrated with a lack of a strategy for dealing with ML data, which is hindering MLpAI initiatives. As a result, organizations often realize too late that they don't have a strategy for dealing with data for ML — or that their existing enterprise data strategy is not fit for MLpAI initiatives. Enterprise data strategies can support MLpAI. However, they must understand new challenges that are presented as a result of MLpAI requirements.

First, technical professionals struggle with provisioning data to support AI and machine-learning initiatives due to the complexity of the data structures (e.g., scalar, nonscalar and multidimensional). Making use of complex data (e.g., for the purpose of ML) is much more difficult than making use of big data. Even smaller complex datasets can take contemporary processing approaches to deal with their unique structures, which differ from the more traditional approaches of dealing with data.

Second, collecting the data is the easy part, but preparing the data for ML initiatives requires the integration of various factors (e.g., structure, type of data, algorithm and business goal). There is no one size that fits all. Thus, provisioning data for MLpAI must be a unilateral approach while remaining part of your enterprise data strategy.

There are many different complex relationships and disciplines with varying data structures that may be associated with a data strategy for the purpose of insights, inferences and predictions, as shown in Figure 2. The different disciplines may require independent data pipelines within the data strategy. Developing strategies for different disciplines does not mean we should abandon our enterprise data strategy nor develop a series of ministrategies. Understanding the different disciplines that are impacted (as outlined in Figure 2) and the complications that may be presented from each discipline enables a more integrated and comprehensive approach to dealing with different kinds of data. See Note 1 for additional information on the complications of ML data in the enterprise data strategy.

Figure 2. Relationship Diagram With Other Disciplines



Source: Gartner (May 2017)

Because of the complexity of the data and the various preparations needed, technical professionals and their data scientist counterparts ask:

> **How do we develop a data strategy to support MLpAI initiatives, and how do we integrate it with our enterprise data strategy?**

Whether building ML applications from scratch, purchasing proprietary platforms or using cloud service providers, this guidance framework will prove useful for data management teams evaluating:

- How to manage data when developing ML applications
- What roles support the execution of the data strategy
- How to build MLpAI applications

The Gartner Approach

A data strategy for MLpAI is a roadmap that includes building data science pipelines from raw data to models where characteristic features are evaluated.

One of the most important tasks that a technical professional must develop to enable MLpAI capabilities is the data strategy aligned with the enterprise data strategy. Whether you're developing a strategy for on-premises architectures or in-the-cloud solutions, the data strategy for MLpAI will improve productivity and reduce overhead. A data strategy for MLpAI is the means of accessing and leveraging dissimilar data structures for the purpose of extracting or discovering features that can be used to make predictions or inferences. This differs from the more traditional enterprise data strategies because of the need to:

- **Analyze time series data using different methods, including linear, nonlinear, univariate and multivariate techniques:** Much of the data used in MLpAI applications involves discrete-time data that data scientists must evaluate for relationships between factors affecting a process or its output. This is a process often known as design of experiments (DoE).
- **Ingest and process on multimedia data types:** MLpAI works with different types of digital data. Traditionally, what was primarily text and numbers is now different multimedia data types, such as audio, images, graphics and video.
- **Collect everything:** Data scientists believe that it's better to have all than not enough because you don't know what you don't know. Dark data may yield significant insights that contribute to a prediction or inference. They often require access to different data structures in their attempt to find the most relevant attributes for a prediction or inference. To avoid problems with having to make repeated attempts to collect from data pipelines, they look to ingest all data for preprocessing. However, this may also cause the citizen data scientists to drown in data. Therefore, careful consideration is needed during the distillery process.
- **Event stream some things:** In some instances, data science citizens may want to employ a learning technique that generates a prediction as data becomes available rather than waiting to learn from a large dataset at rest. For example, when developing MLpAI from telemetry data (i.e., Internet of Things [IoT] data), it may not be feasible to ingest and store the entire training set to determine an inference or prediction. Instead, it may be more feasible to train models consecutively.
- **Preprocess and clean data:** This is different from traditional transformation services because of the amount of preprocessing that includes techniques outside of the standard data integration functionality (e.g., statistical procedures used to convert correlated observations to uncorrelated variables and vice versa).
- **Establish independent workflows with multiple stages:** Data science teams often need to build multiple data pipelines to support ML/AI initiatives. For example, one pipeline may be used for exploratory analysis, and another platform may be used for classification and prediction. Building multiple data pipelines to support ML can be dissimilar, involve different workflow processes and be conditionally dependent.
- **Analyze cross-sectional and multivariate data that aggressively uses correlations (i.e., beyond relationships):** Unlike dealing with traditional relational data, dealing with multivariate data involves more focus on correlation. Data scientists want the best variables to build their models (often evaluated using the DoE methods noted above). They must understand the data and find the right input for models (not to be confused with connectedness). Data scientist typically spend more time on data curation and finding correlations between dependent and independent attributes. Therefore, it is difficult to consume the data in a structured storage repository because of the need to establish a relationship before consuming or using.

Technical professionals have tried various strategies for building data pipelines for MLpAI — all with little long-term impact. This research can act as a framework for building a data strategy to support MLpAI that will produce long-term value for the organization.

The Guidance Framework

A data strategy for MLpAI is not an isolated task of your organization that supports MLpAI initiatives. It is a system that facilitates extracting knowledge from data, and it defines the process for consuming new data sources and datasets as they become available.

The process of getting raw data from disparate systems, and to the CDS to be used in MLpAI, is a greater level of effort than traditional processes involved with data and analytics. A great deal of preprocessing and postprocessing activities goes into delivering or

providing access to data for the citizen data scientist, as shown in Table 1. In contrast to traditional data and analytics strategies, where the transformation process is typically linear, processing for MLpAI is much more cyclical in nature.

To aid with the complexity of the processing required, Table 1 identifies high-level components to support the development of a data strategy for MLpAI.

Table 1: Extending Gartner's Enterprise Data Strategy to Support MLpAI

↓	Acquire ↓	Organize ↓	Analyze ↓	Deliver ↓
Gartner's Enterprise Data Strategy for Data and Analytics	Acquire data regardless of where the information is generated	Organize the data using logical data warehouse (LDW) at the core to connect to data as needed	Analyze data when and where it makes sense	Deliver insights and actionable results to the enterprise
Traditional Enterprise Data and Analytics Approach	Discovery	Integration and repository strategy	Analytics	Embed in business process
(NEW) MLpAI Requirements	Exploration	Preprocessing and postprocessing Correlation and dependence	Test and evaluation (independent verification and validation [IV&V]; see Note 2)	Operationalization using DevOps principles
Alignment Level of Effort	Easy	Difficult	Medium	Medium

Source: Gartner (May 2017)

For more on Gartner's Enterprise Data Strategy for Data and Analytics, see "2017 Planning Guide for Data and Analytics." (https://www.gartner.com/document/code/311517?ref=grbody&refval=3734817)

The new components for MLpAI outlined in Table 1 set the foundation for developing a data strategy for MLpAI, and they should be used to describe how to acquire, organize, analyze and deliver MLpAI projects.

Data Strategy for ML Process Framework

> **The data strategy must support the machine learning (ML) tasks that will be implemented in the MLpAI system and enable supervised, unsupervised and reinforced learning.**

The data strategy for ML process framework in Figure 3 outlines the best practices and process in establishing a strategy for dealing with ML data. These techniques include:

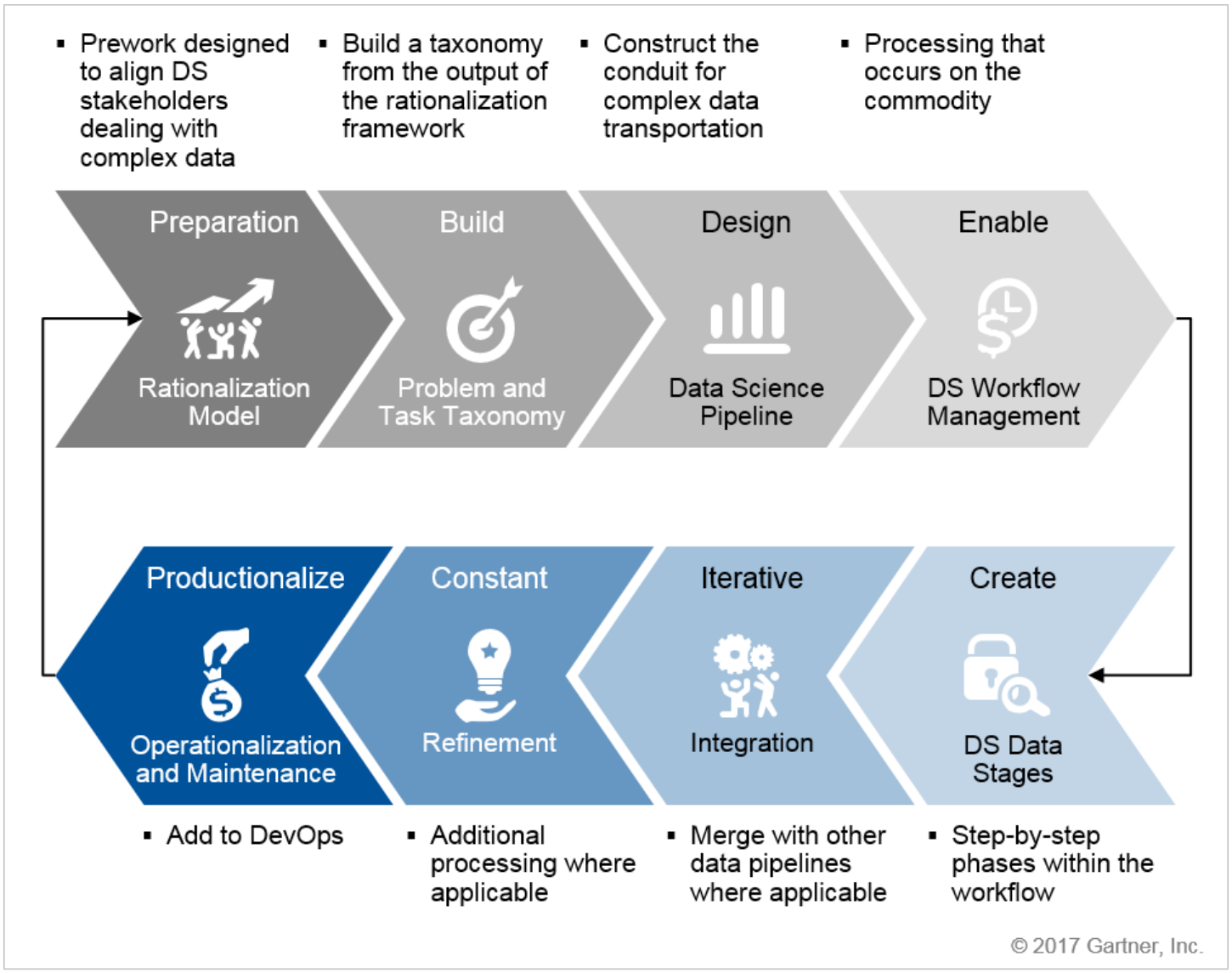
- Building a problem and task taxonomy
- Designing and constructing a data science (DS) pipeline
- Enabling DS workflows
- Creating DS stages

- Integrating and connecting iteratively with the enterprise
- Refining the data pipelines
- Operationalizing and developing maintenance procedures

Figure 3 depicts components of the approach to building the data strategy for MLpAI.

The stages in Figure 3 are sequential.

Figure 3. Data Strategy for ML Stages Framework



Source: Gartner (May 2017)

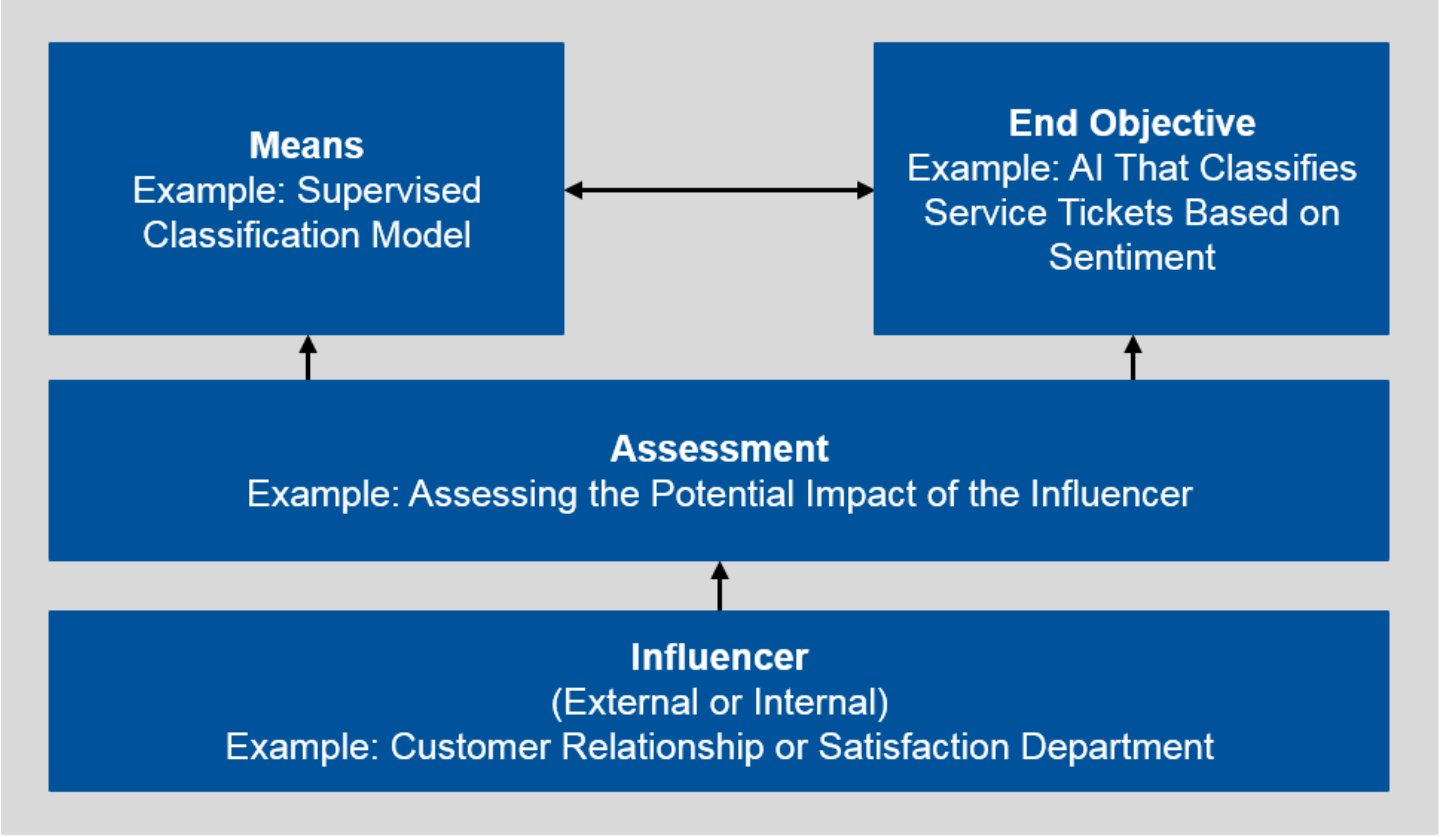
Prework: Building a Rationalization Framework for MLpAI

This section explains how to build a rationalization framework as prework to building a strategy for MLpAI to aid in launching the data strategy needed to support ML. The framework also supports aligning various stakeholders and projects to the strategy. Gartner recommends using the rationalization framework outlined in Figure 4 to help launch the data strategy and provide a platform for collaborating disparate ML projects and their motivations. The benefits of starting with a rationalization framework are:

- Aligning stakeholders to avoid duplication or excessive redundancy in provisioning data
- Defining a structure around the AI and ML problem space to be used as part of the data strategy

- Aiding in requirement elicitation while collaborating with domain knowledge experts and citizen data scientists
- Developing a concept of operations that aligns with the data strategy

Figure 4. Rationalization Motivation Framework: Prework



Adapted from the Object Management Group's Business Motivation Model (http://www.omg.org/spec/BMM/1.3/PDF)

Source: Gartner (May 2017)

Figure 4 describes the rationalization framework that can be used to align the AI/ML task with the data strategy. The framework consists of four components:

- End objective
- Means objective
- Assessment
- Influencers

Defining the End Objective

To help rationalize your data strategy, you must first work with the citizen data scientist and define the end objective. The end objective is the desired final output of the MLpAI application. For example, organizations looking to reduce the overhead of their service help desk may want to use AI to classify service tickets based on sentiment. This would help to prioritize tickets or determine which customers are most angry with your product or service. Explicitly defining the end objective as "using AI to classify service tickets based on sentiment" will help define the actions that must be carried out for the purpose of achieving that objective (end objective).

End objectives can have higher-order clauses depending on how you wish to frame the problem space. For the purpose of this research, we will use a lower-level order to define our end objective. The following is a sample template to approach defining the end

objective.

Sample higher order function: "This AI system is being developed to _____ (end objective/or means objective) by _____ (means objective)."

Sample answer: "This AI system is being developed to **classify service tickets based on sentiment (end objective)** by using a **supervised classification model (means objective)**."

Defining the Means Objectives

The means objective is the action that is needed to achieve the end objective. For example, when developing AI applications, data scientists may want to use a specific learning algorithm or, in some cases, multiple algorithms (e.g., ensemble modeling). The following is a sample template to approach defining means objectives.

Sample means objective: "The _____ learning algorithm is being used to support _____ (end objective)."

Sample answer: "The **supervised** learning algorithm is being used to support the **classification model**"

You can use multiple means objectives to support an end objective, and establishing the means objective helps to define high-level requirements for the data strategy.

Providing Assessment and Governance to Support the Data Strategy

The assessment component is used to evaluate the means and end objectives to ensure consistency and to provide governance. Governance is critical to ensuring that the right objectives are defined, and not necessarily dependent on execution.

For additional information on applying governance models, see "EIM 1.0: Setting Up Enterprise Information Management and Governance." (https://www.gartner.com/document/code/294451?ref=grbody&refval=3734817)

Defining Influencers Critical to the Success of the Data Strategy

The influencers are the stakeholders involved in the projects, and they are necessary to ensure proper buy-in. Influencers can be internal or external to the organization. The purpose for explicitly defining the influencers is to determine stakeholders that may influence data streams designed for MLpAI. For example, deploying the data strategy for users outside of the organization may require the support of third-party consumers that influence when and how users can use the data from the data stream. Defining those influencers early in the development life cycle will reduce the risk of having to rework data streams.

Critical Roles and Responsibilities

There are roles and responsibilities that are critical to the development and execution of the data strategy for MLpAI, as shown in Table 2. The roles or stakeholders are also driving influencers critical to the success of the data strategy.

Table 2: Critical Roles and Responsibilities for MLpAI

Role/Stakeholder ↓	Responsibilities ↓
Data Scientist	■ Establishes the workflow and stages of the strategy that prepare, process and deliver predictions and inferences made by MLpAI
Data Management Professional	■ Develops and designs the acquisition and delivery of data to the appropriate end user, system or application ■ Aligns data strategy for MLpAI with the enterprise data strategy
Domain Knowledge Expert (Business)	■ Provides oversight and verification regarding the expected outcome of MLpAI

Source: Gartner (May 2017)

These three roles should work together to develop a data strategy to support MLpAI. They will also be responsible for maintaining the strategy as the organization matures with the MLpAI efforts.

For more information on developing the roles and responsibilities required for MLpAI, see Gartner's "Staffing Data Science Teams." (https://www.gartner.com/document/code/270087?ref=grbody&refval=3734817)

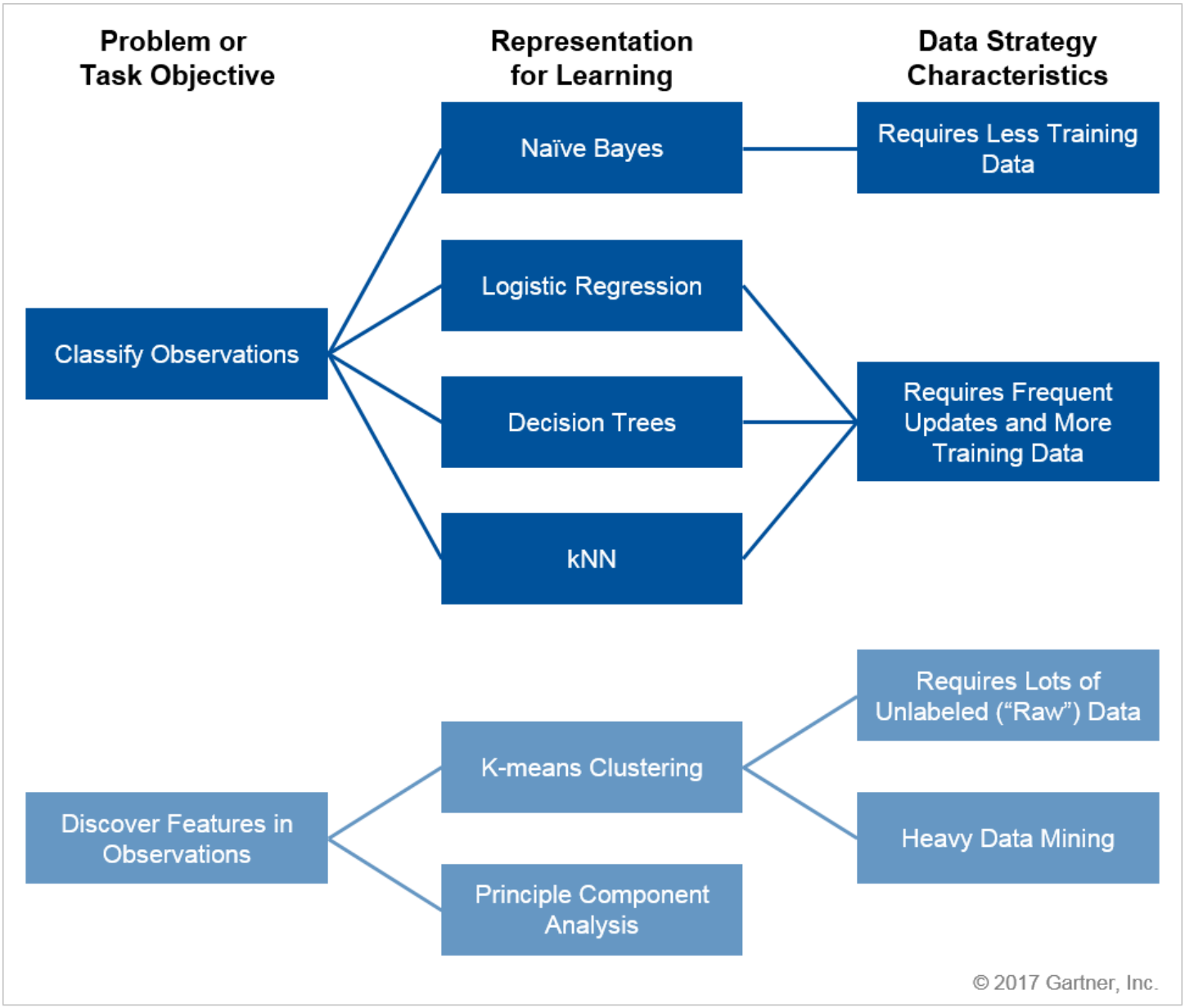
Step 1: Build Problem or Task Taxonomy

Building a problem or task taxonomy involves using the "means" tasks from the rationalization framework and grouping them into classifications. The purpose of building the task taxonomy is to provide a basic understanding about the components of ML algorithms necessary for effective decision making about the type of data required, about how much data is required and about suitable use. There are three components in every ML algorithm: [2] (#dv_2_a_few)

- **Representation:** The hypotheses space that represents what is to be learned (e.g., rules, decisions and models from variables)
- **Evaluation:** The ability to score the ML's ability to classify or categorize observations or features
- **Optimization:** The method for searching for the most optimal solution, based on internal or external evaluation

Representation is the most important component to factor into the data strategy for ML, and it is the foundation for the taxonomy. Building the taxonomy based on representation helps classify based on problem or task, as shown in Figure 5. It is important to remember that there are many different types of ML algorithms. However, for the purpose of this illustration, we focus on a small subset of algorithms.

Figure 5. Sample Problem or Task Taxonomy



kNN = k-nearest neighbors

Source: Gartner (May 2017)

The most important part of the taxonomy is properly identifying the problem or task. To help with this process, Gartner recommends focusing on two areas to help drive this analysis:

- **Classifying problem/observations:** This involves using datasets to train a learning algorithm to classify new observations. For example, in the classical example of classifying emails based on spam or nonspam, the classifier uses a training set of data to learn to classify all incoming emails.
- **Discovering features in observations:** This is often used when it is not feasible or efficient to use training data, but the desire is to discover features or new information from large datasets.

After the desired representation for learning is selected, Gartner recommends building features (i.e., characteristics) about the data. For example, as outlined in Figure 5, certain classification systems will require more training data than others. This is a good indication that the data strategy may need to involve additional preprocessing by the citizen data scientist or technical professional.

Step 2: Design Data Science Pipeline

Pipelines designed for MLpAI are based on the learning approach. Either they are constructed based on the entire training dataset, or they are based on incremental loads of sequential order data.

A data science pipeline is a system that captures data attributes and automates the movement and transportation of data for analysis and building data science applications. It is also categorized as a learning approach based on the consumption, or lack thereof, of training data. A data science pipeline has the following parts that will be discussed in subsequent sections of this document:

- The pipeline
- The workflow
- The stages

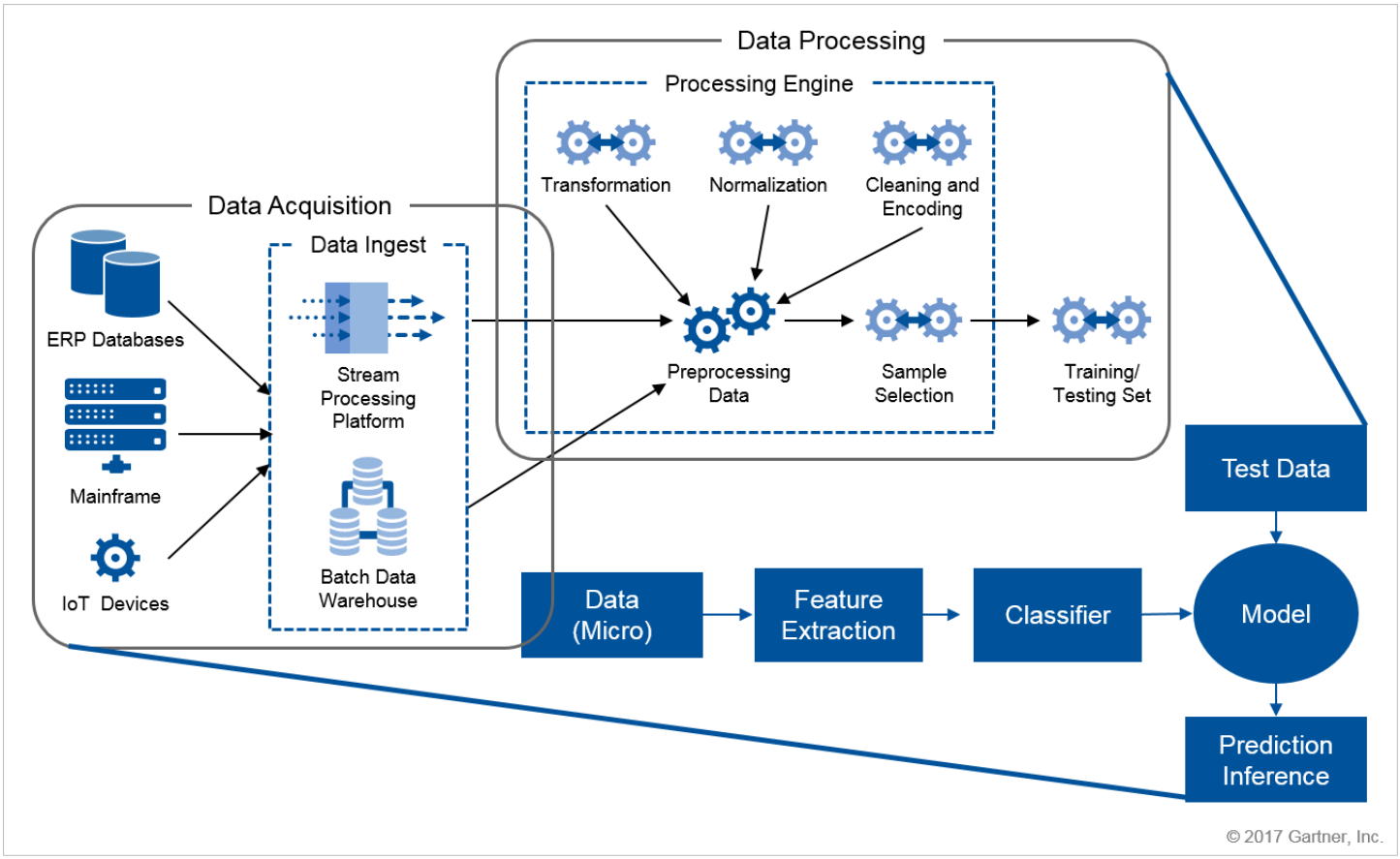
Within a data science pipeline, you can define data-driven workflows (outlined in Step 3) and tasks (outlined in Step 4) that can achieve operational objectives within the workflow. Think of constructing the data science pipeline as building an underground pipeline system. The objective is to enable access to a commodity (i.e., data) by building a conduit that delivers the commodity to the algorithms and ultimately the relevant user and/or system. The DS pipeline acts as the infrastructure to support workflows and the stages that follow.

ML needs two types of pipelines:

- **Batch pipelines:** These are used to feed data into an algorithm in batches so that the learner or classifier can develop hypotheses off of the entire training dataset. Batch pipelines are typically used when large compute resources are available and larger predictions are required (e.g., robotic process automation, conversational AI and recognition). Models are trained in batches.
- **Online pipelines:** These are used to feed data into a learner as the data becomes available (i.e., streaming data). Data is transported sequentially as the learner is constantly updated, and the model evolves with new information. Online pipelines are typically used when you need quick predictions and the training sets are too large to process (e.g., fraud detection, web advertisements and spam detection). Models are trained consecutively.

Figure 6 outlines data pipelines involved in data acquisition. For more information regarding the architecture to support MLpAI applications, refer to "Preparing and Architecting for Machine Learning." (https://www.gartner.com/document/code/317328?ref=grbody&refval=3734817)

Figure 6. Standard Data Science Pipeline



Source: Gartner (May 2017)

Gartner recommends providing multiple ways to create the data science pipelines. For example, you can offer the ability to create or modify a data pipeline by the user using a web-based service when building for cloud environments (e.g., AWS data pipeline). Or you can establish a method for constructing the pipeline using a proprietary data integration tool, such as RapidMiner, SAS or Informatica. Or you can use high-level programming languages, such as Python, when heavy customizations are necessary.

2.1 Constructing Batch Data Science Pipelines

Constructing a batch data science pipeline is a process leveraging traditional data integration strategies that involves ingesting and integrating data from batch data systems to allow for training over an entire training dataset. The data can be traditional text or multimedia data. For example, batch data science pipelines typically involve extract, transform and load (ETL) jobs or extract, load, and transform (ELT) data.

There are common application patterns used to develop batch data science pipelines, and they are closely aligned with traditional data and analytics patterns for data integration, as shown in Table 3.

Table 3: Common Architectural Patterns for Constructing Batch DS Pipelines

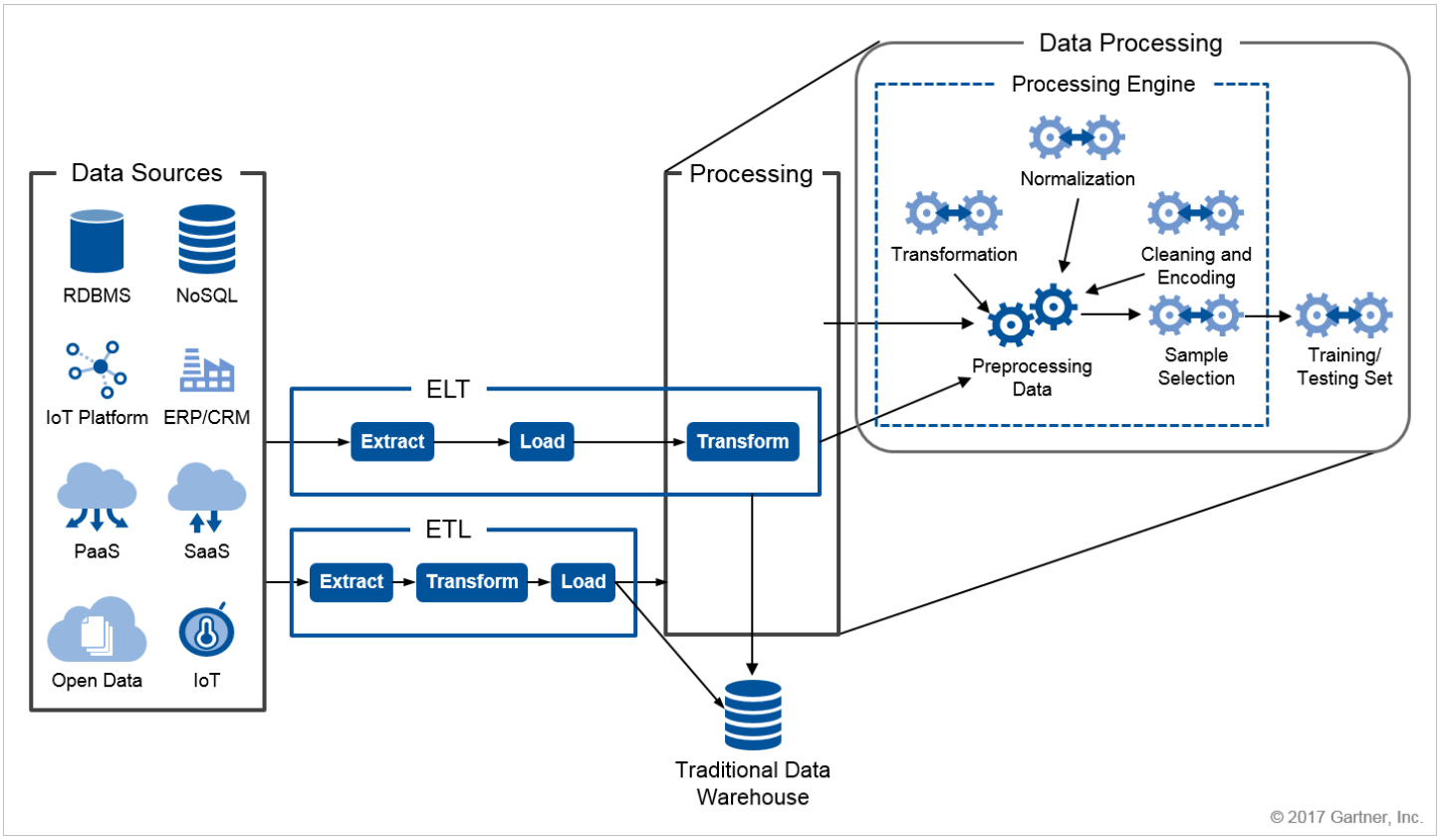
DS Pipeline	Architectural Patterns	Design Patterns	Solution Patterns	Related patterns
Batch	Extract, transform, and load (ETL) Extract, load, and transform (ELT)	Data integration	Bulk ML processing	Data virtualization Analytical data lakes Data warehouses

Source: Gartner (May 2017)

The batch DS pipeline should focus on moving data from the source system to the ML model, but it is not necessarily involved with how the model executes on the data. Figure 7 outlines a simple use case involving developing a data pipeline for an MLpAI application. Gartner recommends developing pipelines with workflows that include the ability to process or transform data in transit (ETL) and when data is at rest (ELT). In many ML applications, preprocessing of data will occur while the data is at rest. For example, in feature

engineering, which is a preprocessing activity, transformations will occur in a Hadoop environment independent of transformations that occur to clean data in transit.

Figure 7. Batch Data Movement Architecture (ETL vs. ELT)



Source: Gartner (May 2017)

2.2 Constructing Online Data Science Pipelines

Constructing online data science pipelines is a process for establishing methods for capturing sequential data continuously from data sources to update a prediction. The online data science pipeline is most commonly used in situations where it is difficult to train over a large dataset, so it allows for a more incremental learning approach. Table 4 shows some common architectural patterns for developing online data science pipelines.

Table 4: Architecture Patterns for Constructing Online Data Science Pipelines

DS Pipeline	Architectural Pattern	Design Pattern	Solution Pattern	Related Pattern
Online	Hybrid transactional/analytical processing (HTAP) Event-driven architecture Message-oriented architecture Change data capture (CDC)	Streaming ingestion	Online ML processing	Streaming analytics

Source: Gartner (May 2017)

With online DS pipelines, data is continuously updated to develop new hypotheses. In contrast, batch data pipelines develop hypotheses on the data after it has been ingested, trained and stored at rest. An advantage of offering online DS pipelines is to enable updates to predictions or inferences before needing to retrain the data. In other words, instead of waiting to train the ML models until new features arrive, online pipelines incrementally update the hypotheses when new features or data becomes available. Online DS pipelines are common when it is not feasible to evaluate the entire training set.

Step 3: Enable Data Science Workflows

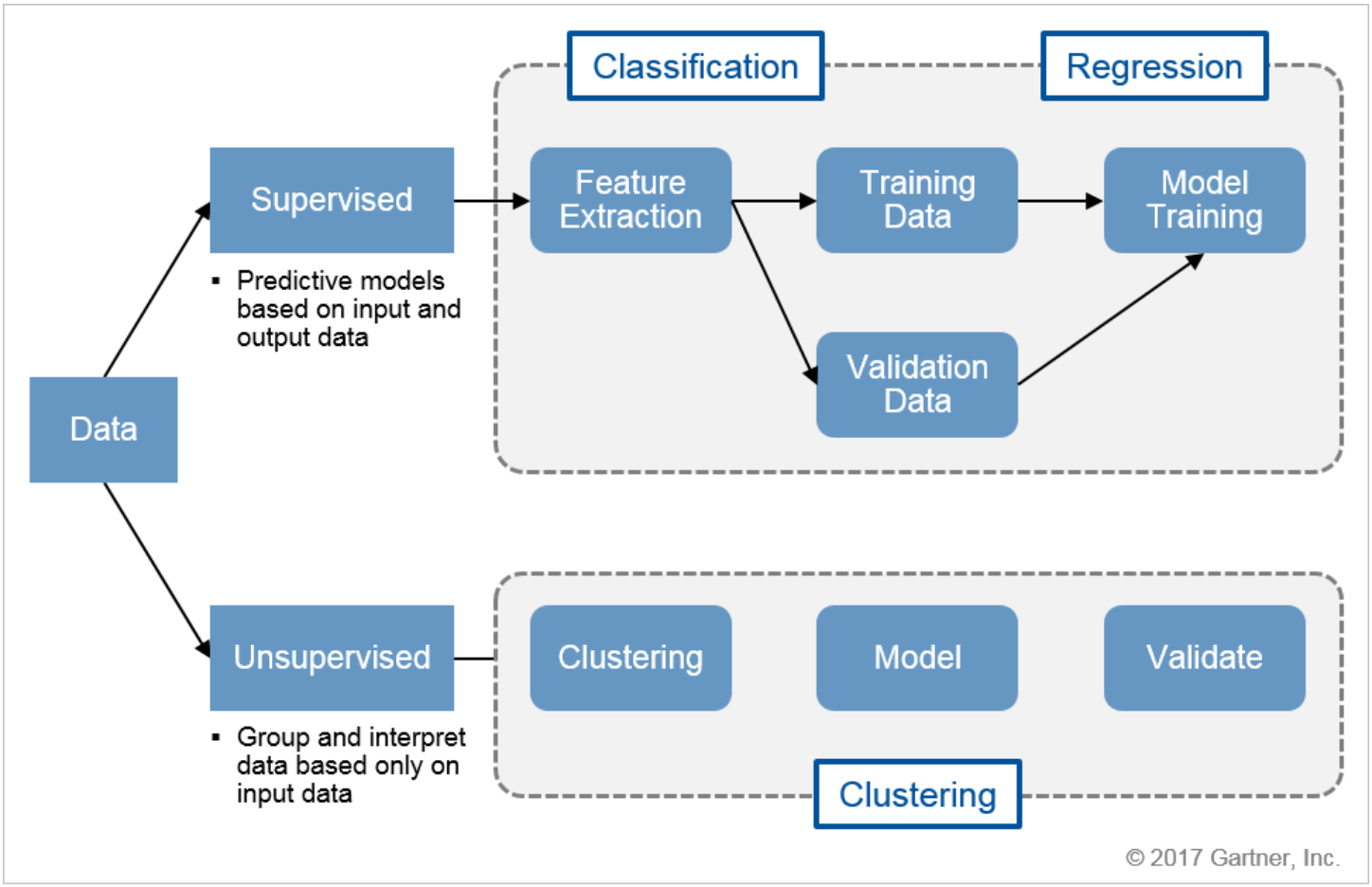
Workflows for MLpAI applications can be independent or dependent processes.

The workflow management system is the most complex portion of the data strategy. It involves developing streams of data to process based on the desired learning algorithm. This step describes two of the most common algorithms:

- **Supervised learning workflow:** Takes raw data and trains a model based on labeled features (input and output required)
- **Unsupervised learning workflow:** Takes raw data and applies unlabeled features to a model for the purpose of exploratory analysis

Figure 8 provides a brief overview of the different workflows produced for the DS citizen. Notice in Figure 8 that the workflow for unsupervised learning is significantly shorter than the workflow for supervised learning. Because of this, Gartner recommends that your data strategy distinguish between the two paths within your data pipeline.

Figure 8. Machine-Learning Workflows Overview



Source: Gartner (May 2017)

Each learning algorithm requires an independent workflow because of the use of data required during the processing stages. For example, in some cases, the DS citizen may need to prepare the data for the purpose of classifying or categorizing it and then evaluate the results for accuracy (e.g., supervised learning and reinforcement learning). In other cases, the DS citizen may want to explore and find latent structures from unlabeled data without the need to evaluate for accuracy because the objective is to explore (e.g., unsupervised learning). The workflows used to support these requirements will contain different stages, which are discussed in the next section.

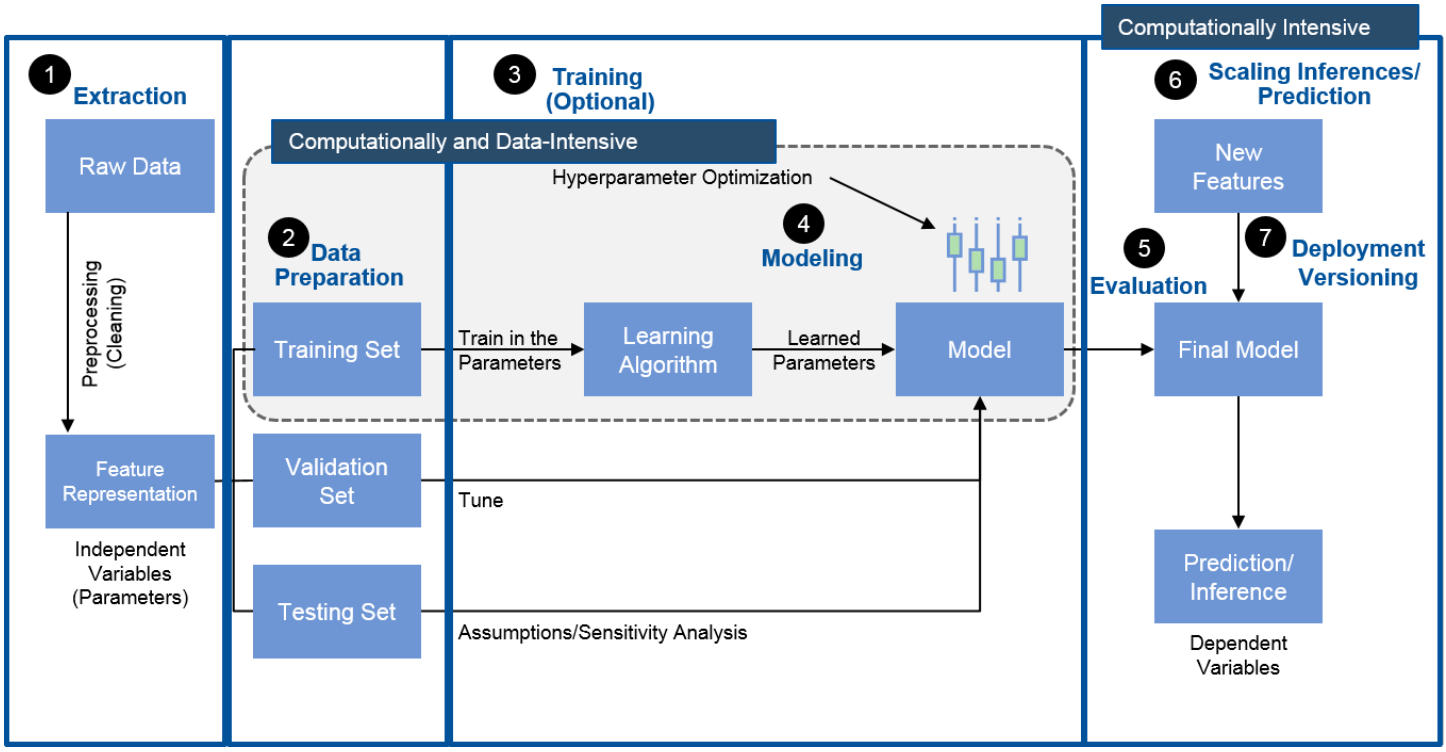
3.1 Enabling Supervised Learning Workflows

Figure 9 provides a sample supervised workflow with seven steps:

- **Extraction** is the process of obtaining observations or parameters from raw data. In many cases, this is simply raw data pulled from a data lake.
- **Data preparation** is the process of transforming the data (e.g., cleaning) and extracting features based on domain knowledge. Gartner recommends heavy collaboration with business experts with domain knowledge during this process.
- **Training** is the process of developing datasets for the purpose of training, validating, and testing against an ML model.
- **Modeling** is the process of fitting data to the model based on training of raw data and the process of formulating abstract parameters (i.e., hyperparameters) that is learned, but not directly from raw data.
- **Evaluation** is the process of verifying the output of an ML model.
- **Prediction** is the inferences made by the model.
- **Deployment** is the process of distributing the results to an application, system or human-to-computer interaction process.

Supervised learning workflows should contain these steps, and each step can be extremely difficult without the right tools. Gartner recommends evaluating tools based on their ability to provide the steps outlined in Figure 9 for supervised workflows.

Figure 9. Sample Supervised Learning Workflow



Source: Gartner (May 2017)

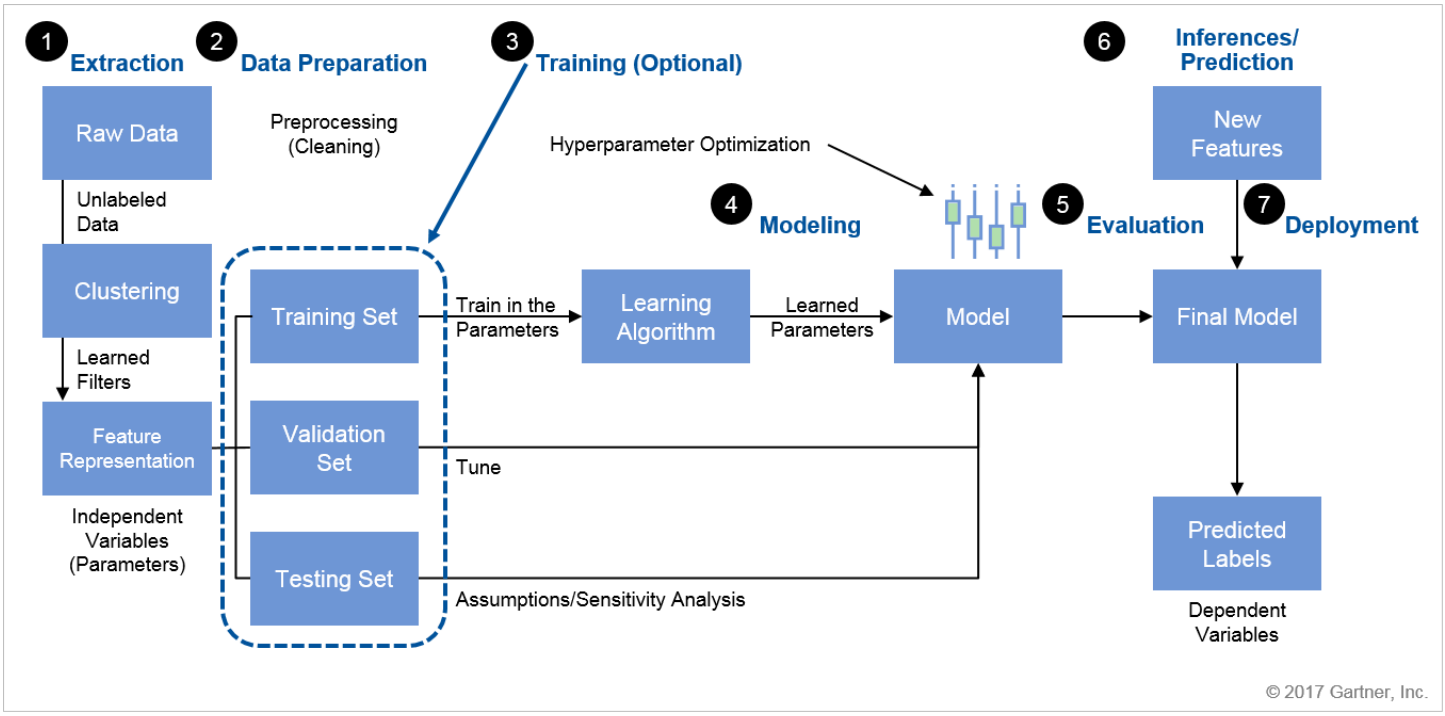
3.2 Enabling Unsupervised Learning Workflows

For unsupervised learning workflows, the workflow changes due to the need to explore the similarity in raw data or observations. Typically, the objective is to address the heterogeneity of the data, and instead of dealing with broadly disparate observations as raw data, the DS citizen will often look to divide the data into more homogeneous subsets. This is typically why clustering is a common tool for unsupervised learning requirements (as outlined in Figure 8). In addition, data mining is a common application for unsupervised learning requirements where you have a lot of data available but no teaching data available. In many applications, this can be an approach used to get training data for supervised learning.

The difference in this workflow for unsupervised learning is that the training steps are optional, as illustrated in Figure 10. CDSs may decide to use raw data that is not labeled to feed into a model for the purpose of learning to cluster input data into a set of

classifications that are not previously defined.

Figure 10. Sample Unsupervised Learning Workflow



Source: Gartner (May 2017)

A rising area in unsupervised workflow requirements is the use of deep learning. Deep learning is an attempt to simplify the process of feature learning on labeled or unlabeled data at massive scale and using massive amounts of input/output (I/O) data. In other words, as our dataset grows, the performance of our algorithms improves thanks to deep learning. This is a characteristic that was unavailable to algorithms of the past. However, we've traditionally used supervised learning to manually design our features. [3] (#dv_3_unsupervised_feature) The use of unsupervised feature learning will allow us to learn without having to explicitly define features, which reduces the need for human intervention. Features are attributes from raw data that apply domain knowledge to abstract the data. For example, image recognition is a common use of deep learning that recognizes features in images (e.g., tires, seat, handlebars).

Step 4: Create Data Science Stages

In MLpAI applications, we make predictions on features, not raw data.

Stages are produced within the workflow, and they may be different depending on the learning algorithm. Gartner recommends defining the DS stages based on the workflow identified. There are typically nine stages within a supervised DS workflow, as outlined in Table 5. In unsupervised workflows, the stages will be similar, but with less focus on the upfront training stages. Figures 11 and 12 illustrate the stages within a workflow according to our sample workflow management system.

Table 5: Stages Involved in Supervised Workflow

Workflow Stages ↓	Description ↓	Example ↓

1. Feature engineering	This is the process of extracting features from raw data. Features are individually measurable and observable data points.	**Example:** A restaurant reservation system with a data attribute for table reservations may be "null." The feature extracted would be "reserved." Therefore, "null" means "reserved."
2. Defining a training set	The training set is a dataset with the correct answer or desired/target outcome (i.e., the labeled data that you'd like to predict).	**Example:** To predict table reservations in a restaurant system, training data includes database records that properly identify when a table has been "reserved" or "not reserved."
3. Defining a validation set	The validation set is a subset of data (independent of the training set) that is used to validate the model's output against the training set. It is often used for parameter selection to avoid overfitting or underfitting models. Validation datasets are also used to tune the parameters of a model. Validation sets are also referred to as hypotheses.	**Example:** A percentage of data is used to tune the parameters of the model that predicts when tables will be reserved.
4. Defining a testing set	The testing dataset is a subset of the data that is independent of training data, used for performance evaluation.	**Example:** A portion of the dataset with unknown target values is used for testing the model.
5. Defining the learning algorithm	This is the learning algorithms used to produce a model.	**Example:** In classification requirements, Support Vector Machines (SVM), Naïve Bayes, Nearest Neighbor, etc. In regression requirements, Linear Regression GLM, Ensemble methods, decision trees, neural networks, etc. In clustering requirements, kMeans, hierarchical, neural networks
6. Modeling	Modeling is the process of building models based on ML algorithms. Models may contain multiple learning algorithms and are a higher order function of the learning process (i.e., models have inputs converted to outputs).	**Example:** You use ML models to get predictions on new data (input) that you don't have the answer to (output). The model uses learning algorithms (e.g., decision tree) to produce the prediction (output).
7. Addition of new features	This is the process of adding new features to the model once it has been trained, tested and validated. The new dataset of features should be independent of training, validation and testing dataset.	**Example:** New data that is not part of training, testing or validating datasets can be added to the model.
8. Final model after evaluation	This is the finalized model after verifying the prediction using domain knowledge.	**Example:** Create a final anomaly detection model after satisfied with the confidence level of predicting anomalies.
9. Prediction	This is the final output of the model. It is often referred to as the inference or prediction.	**Example:** The model predicts that certain incoming mail is "junk mail" based on learned data.

Source: Gartner (May 2017)

Table 6 outlines the different stages involved in unsupervised learning workflows. As previously discussed, many of the stages will be similar to supervised learning workflows.

Table 6: Stages Different in Unsupervised Workflows

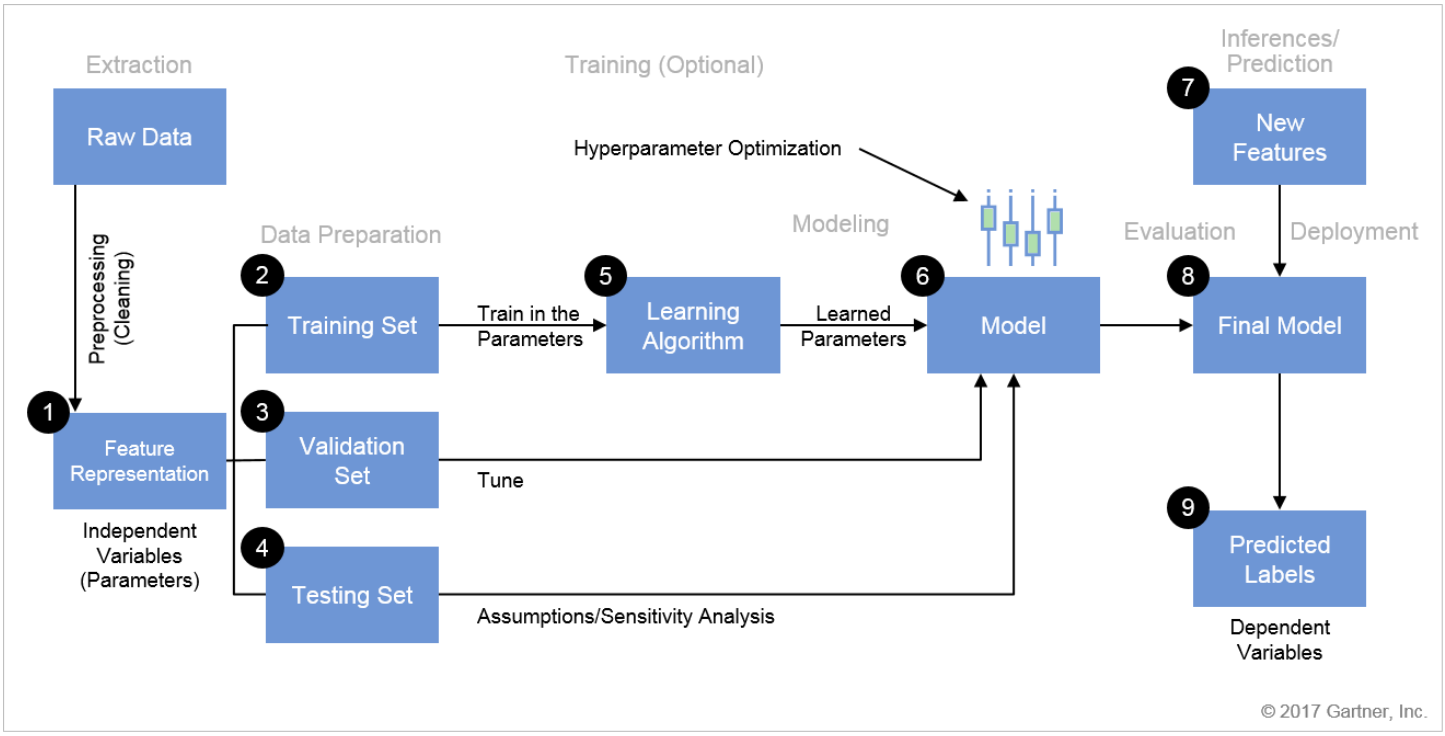
Workflow Stages ↓	Description ↓	Example ↓
Finding patterns in data (clustering/categorization)	The process of dividing big data into smaller groups so that the data within each group is relatively similar [4]	**Example:** Finding patterns in the restaurant data to determine similar variables associated with table reservations

Unsupervised feature learning	The process of mining big data to discover or learn new features from the unlabeled raw data	**Example:** Taking unlabeled data from many restaurants to learn new patterns that may help us predict when tables will be reserved in restaurants

Source: Gartner (May 2017)

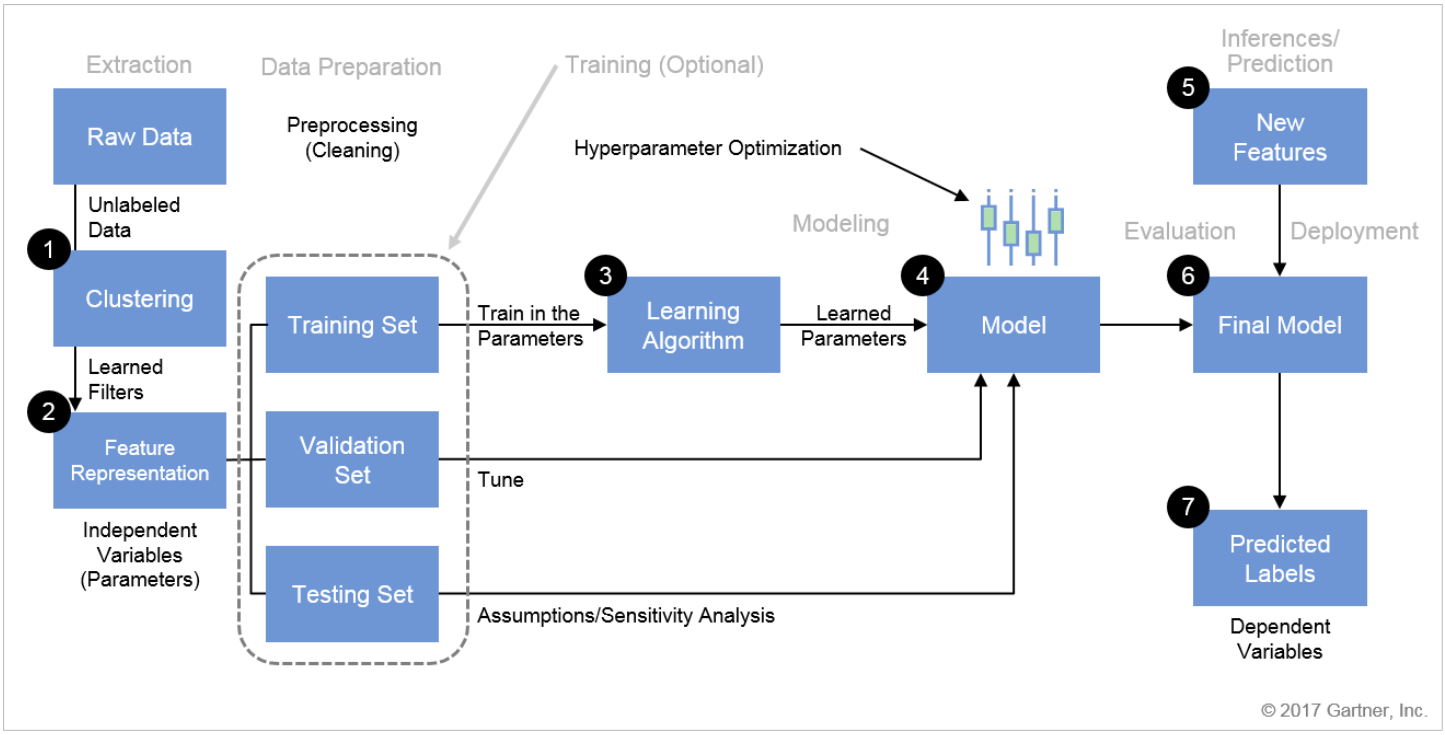
Stages vary between supervised and unsupervised workflows, with some stages that are optional, as shown in Figure 11 and Figure 12.

Figure 11. Supervised Workflow Stages



Source: Gartner (May 2017)

Figure 12. Unsupervised Workflow Stages



Source: Gartner (May 2017)

4.1 Critical Stages of Preprocessing

> In the ML world, relevancy of data is more important than the quantity of data.

Preprocessing stages are the heart of the data strategy, and often times of the ML/AI architecture. The stages will receive data from multiple kinds of sources and perform processing before delivering data to the appropriate model. The goal of the preprocessing stage is to supply the ML models with the most relevant data. Data scientists use a number of preprocessing techniques. However, for the purpose of this research, we will only focus on the critical processes more common today.

4.1.1 Supporting Transformation

Transformation of data for ML typically involves manipulating raw data for the purpose of enhancing the learning process. For example, ingesting data that is in a long string format may limit that amount of power from the learning algorithm because a long stream of data lacks expression. Manipulating the data to break apart the string may provide greater power by creating additional features and broadening the learning experience.

Supporting transformation involves leveraging tools and languages that allow for the manipulation of data to an optimal format for ML. Common languages for transforming data include Python, Java, R and Scala. Common ETL tools may also be used to transform data for ML.

4.1.2 Supporting Feature Engineering

Feature engineering is often touted as the most difficult, yet most critical form of preprocessing for ML. It involves extracting features from data with the aid of domain experts knowledgeable of the data. It is typically a manual effort. However, there is a growing interest in automating the extraction of features.

Two commonly used techniques to reduce the complexity and level of effort of manual feature engineering include:

- **Supervised feature learning:** The process of learning features from labeled data
- **Unsupervised feature learning:** The process of learning features from unlabeled data

The data strategy must support the ability to perform feature extraction from raw data. Gartner recommends looking at technologies that can be used to support this level of processing. For more information, see "Preparing and Architecting for Machine Learning." (https://www.gartner.com/document/code/317328?ref=grbody&refval=3734817)

4.2 Supporting Computationally Intensive Training Stages

> Training environments are compute-intensive and data-intensive stages within the workflow.

Training environments for MLpAI is one of the most computationally intensive stages in the workflow. Much of the time spent by data scientist on MLpAI is in the training stages. Training is one of the ways models learn to produce predictions or inferences when new data is presented; therefore, it can be a very data-intensive process as well.

The data strategy for MLpAI will likely involve a training stage that involves defining a training set — typically a subset of raw data. Training sets can be of different data types (e.g., images, text, audio and video) and must contain the correct answer or target attribute or feature. The learning algorithm will perform processing on the input data to find correlations with the target attribute and produce a model (also known as "fitting a model to a set of training data"). The model's ability to accurately and efficiently find the correlations and repeat the finding with new data is how we determine the confidence in the model.

There are two common challenges with supporting training stages that the data strategy should be cautious of:

- **Overfitting:** When you have so many parameters relative to the attributes in the raw data. For example, in sentiment analysis, let's say your model is accurately determining that customer dissatisfaction is related to the size of your service window when there's a problem. The model uses a survey with a Likert scale of 1 being unhappy and 5 being extremely happy. The service window number of hours decreases as customer satisfaction increases. However, in some cases, the data shows that the number of hours in a service windows increases as the satisfaction increases because the user sample included users that worked for the company. That data is considered noise b/c working for the company should have nothing to do with the impact of service hours on customer satisfaction.
- **Underfitting:** When an underlining trend or correlation cannot be determined based on the data, the model is said to be underfitting, and the model may yield poor predictions. For example, in the same sentiment analysis model as above, let's assume that the model did not find the correlation between customer satisfaction and the size of the service window. The model would yield poor inferences because it failed to correlate between the different variables or parameters provided.

The training environment is where much of the self-service processing will occur. Therefore, when developing a data strategy for MLpAI, Gartner recommends defining how to provision data and enabling postprocessing (e.g., sensitivity analysis and cross-validation) of data to avoid overfitting or underfitting ML models.

4.2.1 Complement Your Data Strategy With Pretrained Networks, MLaaS and AI APIs

> AI will be the main battleground for cloud providers through 2020

Training models often require very large datasets and can be very time-consuming. As a result, pretrained networks (e.g., Google Inception-v3) with packaged datasets (e.g., ImageNet) are often used to reduce the overhead associated with training models. Gartner recommends complementing your data strategy with pretrained networks and packaged datasets when you do not have enough data to train a complex or strong model.

MLaaS and AI APIs are emerging as cloud services that can reduce the overhead associated with training data and developing models that support MLpAI. The quality and depth of functionality makes ML and AI services in the cloud an attractive solution for clients looking to rapidly advance their MLpAI initiatives. Clients often choose AI and ML services in the cloud when they don't have staff to support large-scale MLpAI implementations or when they are looking to diversify MLpAI technologies by developing highly flexible microservice architectures. MLpAI services, such as Amazon Rekognition, Cognitive Services from Microsoft Azure or Google Cloud Machine Learning, provide ML services with pretrained models to accelerate developing independent workflow stages.

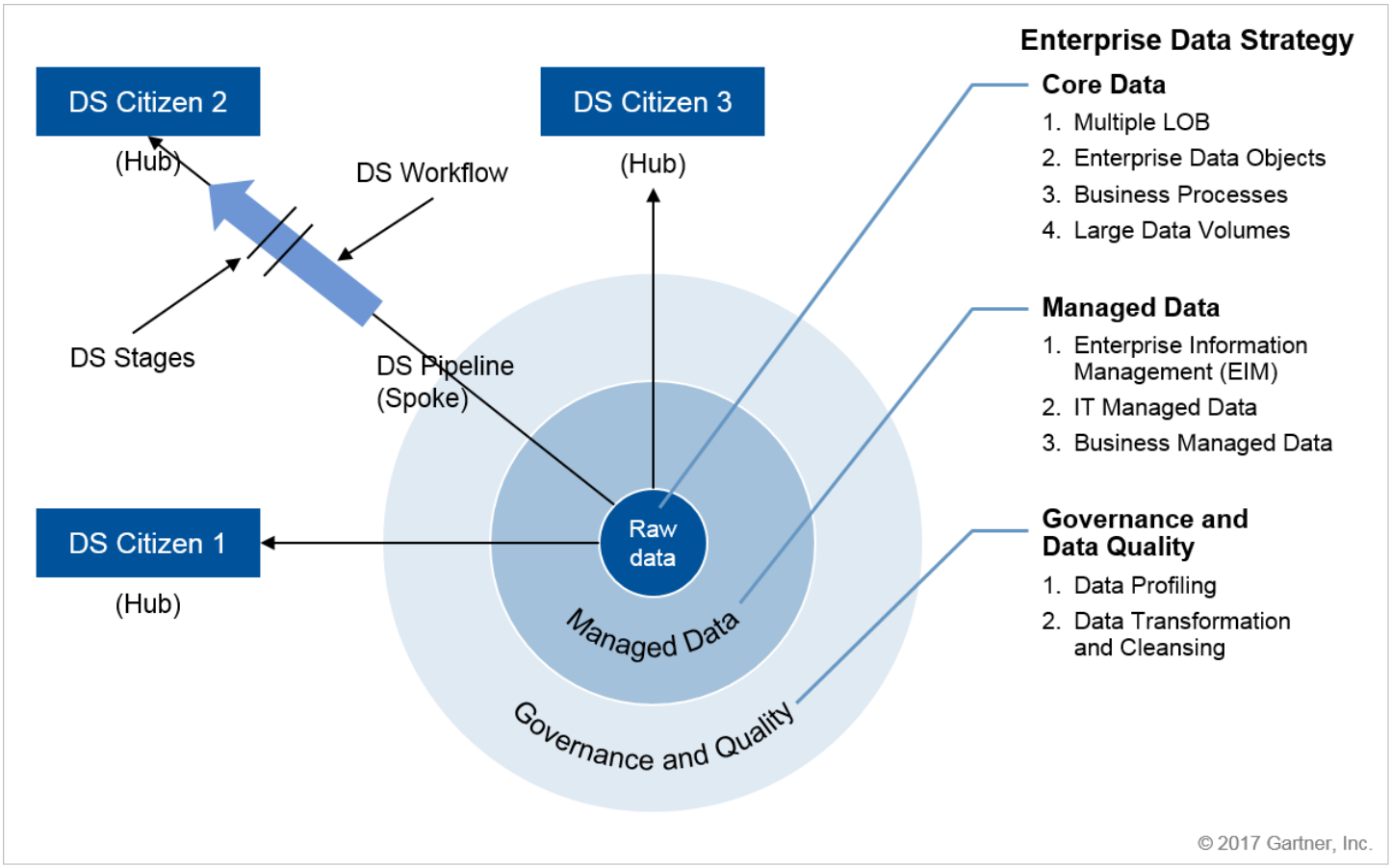
Lastly, deep learning frameworks offer architectures support for model creation and optimization. There are numerous deep learning frameworks (e.g., TensorFlow, Caffe, MXNet, CNTK) offering different advantages to their executable architecture. However, the data strategy to support them will be the same, even though the execution of the MLpAI program will vary.

Step 5: Integration

MLpAI requires integration solutions that are capable of processing complex data from a variety of data sources and data types. In many cases, this involves integrating with existing data strategies and connecting to new data sources. To achieve this, Gartner recommends giving the CDSs the capability to deploy, manage, govern and integrate data pipelines using integration platforms.

An integration platform solution aligns the DS pipelines with existing enterprise data strategies. It is a shift to a more data-centric framework that enables multiple pipelines where data moves along stages as workflows connected to the DS citizen, as shown in Figure 13.

Figure 13. Integration and Alignment Platform



Source: Gartner (May 2017)

Numerous technology patterns and vendors support integration for MLpAI. For more information on developing integration platforms to support MLpAI, see "Deploying Effective iPaaS Solutions for Data Integration." (https://www.gartner.com/document/code/324279?ref=grbody&refval=3734817)

Step 6: Refine With Storage

For a data scientist, feedback is a vital part of the data strategy. It helps to evolve DS workflows so that they can be refined into a better, more accurate process. Refining MLpAI solutions requires building in critical evaluation, workflow evolution, and encouraging a feedback loop into the DS workflow process.

One way to assist in the refinement process is with storage that offers reduced response time. For the purpose of this section, storage refers to the storing of the output of ML models and not the storing of ML model metadata. For more information on storing ML metadata, see "Preparing and Architecting for Machine Learning." (https://www.gartner.com/document/code/317328?ref=grbody&refval=3734817)

There are four common approaches to storing the output of ML models:

- Memory
- Distributed file systems
- Distributed file stores
- Databases

Table 7 provides some use cases for leveraging the aforementioned approaches to evolve and refine the DS pipeline. This is not an exhaustive list of technology frameworks. Instead, it provides an overview of options available based on different storage types.

Table 7: Evolve and Refine Use Cases With Different Storage Types

Storage Type	ML Solution Pattern	Use Case	Benefit	Sample Supported Technology
In-memory	Feature extraction	■ Using memory to cache subprocesses designed to extract features from raw data	■ Speed	■ Cache CPU memory ■ Arrays
Distributed file systems	Hyperparameter optimization	■ Finding and storing parameters most relevant to an ML model ■ Storing images, videos and sound	■ Scale ■ Availability ■ Handles unstructured data well	■ Java-based file systems (e.g., HDFS) ■ Amazon S3
Distributed data store	Hyperparameter optimization	■ Finding and storing parameters most relevant to an ML model ■ Storing images, videos and sound	■ Scale ■ Availability ■ Unstructured	■ NoSQL storage types ■ Cassandra
Relational databases	Reporting output	■ Storing the output of the model and/or integrating with more traditional reporting frameworks	■ Structured ■ Availability ■ Integration	■ Relational database management systems (RDBMS) ■ Oracle, Microsoft SQL Server

Source: Gartner (May 2017)

6.1 Using Memory

Memory can play a critical role in helping data scientists refine data workflows. It offers the ability to cache subprocesses in memory before applying data to the algorithm or model. For example, in use cases involving the need for fast feature extraction as a part of preprocessing data, in-memory systems offer the benefit of low-latency, higher-throughput processing of stages within the DS workflow. This provides the capability to refine and evolve data within the DS pipelines.

Memory is also used to store, for example, classifiers in the memory for faster processing and retrieval. However, there is a trade-off with using memory because the memory is often flushed, removing old data and old training sets. Although in-memory databases offer substantial processing power, they may not be ideal for long-term persistent storage.

Gartner recommends using in-memory databases for processing tasks, such as sophisticated data manipulation or storing of transient runtime state (e.g., storing subprocesses during executions). For more information on using in-memory databases, refer to "The Future of High-Performance Storage." (https://www.gartner.com/document/code/325069?ref=grbody&refval=3734817)

6.2 Using Distributed File Systems

Distributed file systems work well for storing unstructured information needed for long-term persistence at scale. The more common example of using distributed file systems is the Hadoop Distributed File System (HDFS) to support analytical data lakes. HDFS data lakes offer a scalable data repository solution and can also be used to refine and evolve data prior to fitting the data to an ML model.

Gartner recommends considering distributed file systems when looking for a scalable way to enable a distributed architecture to support MLpAI applications.

6.3 Using Distributed Data Stores (Persistent Data Store)

Similar to distributed file systems, distributed data stores are great for scaling large sets of data over a network, dealing with unstructured data and offering a high-availability solution.

In many hyperparameter optimization techniques, read latency, write performance and data processing speed is key because of the exhaustive searching and evaluation required. For example, using a grid search method for hyperparameter optimization, the method must sweep through all parameters available in the parameter space. Moreover, the method breaks the workload into a number of parallel tasks that are evaluated independently. For this reason, Gartner recommends identifying and selecting an optimal data store to support the refinement process of the strategy. For more information, review "Identifying and Selecting the Optimal Persistent Data Store for Big Data Initiatives." (https://www.gartner.com/document/code/322578?ref=grbody&refval=3734817)

6.4 Using Relational Databases

Relational databases remain the leader in storing structured information for ML/AI. In the case of MLpAI applications, relational database systems perform well in persisting the output of ML models and integrating that output with traditional reporting frameworks (e.g., business intelligence [BI] reporting and dashboards). As a result, Gartner recommends using a robust analytical and structured database to support analytical functions with MLpAI output, such as aggregations and summaries.

Selecting the right database to support your MLpAI application will depend on the use case. However, when starting an MLpAI initiative and to support rapid development of using MLpAI output, persisting to a database in the cloud should be evaluated. For more information on selecting the proper storage, refer to "Evaluating the Cloud Databases From Amazon Web Services" (https://www.gartner.com/document/code/303836?ref=grbody&refval=3734817)

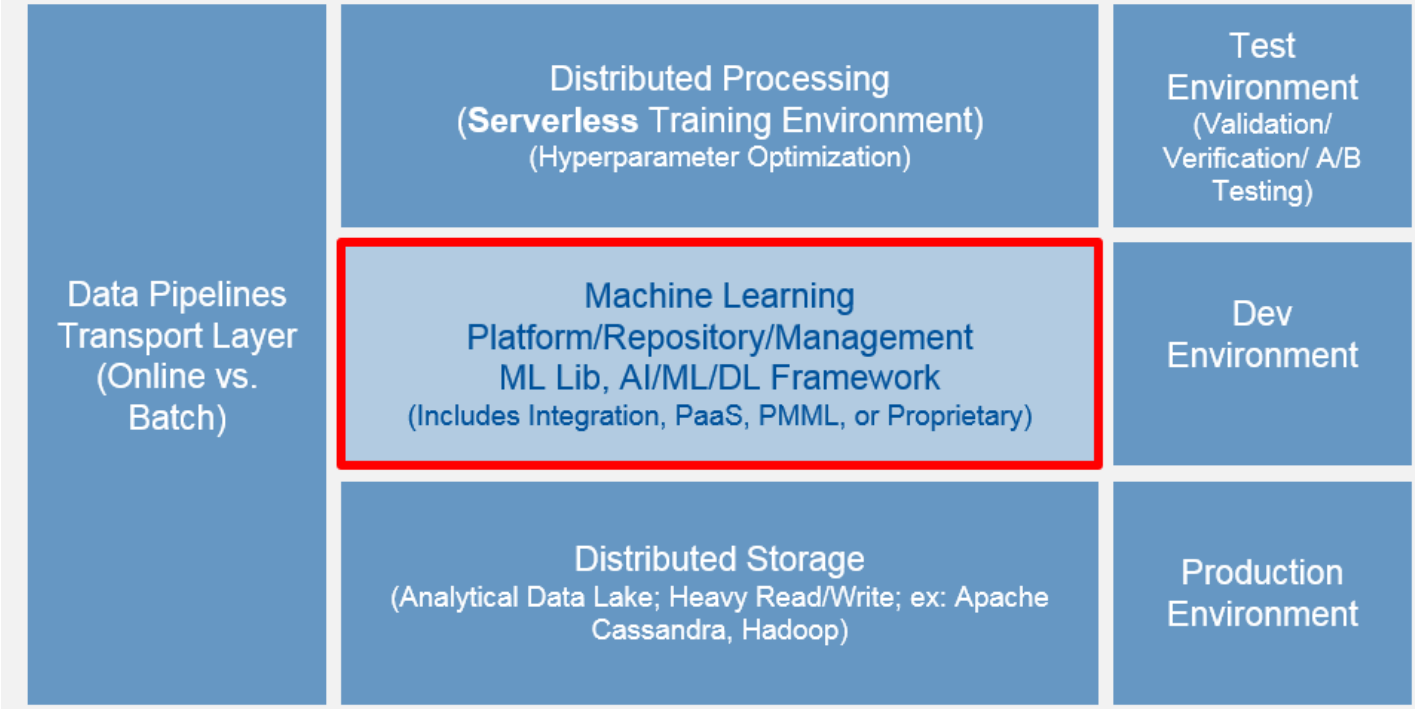
Step 7: Operationalization and Maintenance

The most computationally intensive and data-intensive parts of MLpAI are processing (for training) and execution (for model inference and prediction).

The final step in developing a data strategy for MLpAI applications is to establish a framework for operationalization and maintenance. The data strategy requires an operational construct that separates the computationally intense processing from the computationally intense execution (inference and prediction), as shown in Figure 14. This separation will enable greater use of data for testing and evaluation of MLpAI projects.

Operationalization is an approach to how you deploy and manage DS pipelines, workflows and stages. Once pipelines are created and are processing workflows and stages, a framework is needed to ensure repeatability. Figure 14 outlines a framework that can be used to support operationalizing MLpAI applications.

Figure 14. Sample Model Framework for Operationalization and Maintenance



Source: Gartner (May 2017)

The framework outlined in Figure 14 provides key components that will enable the operationalization of data used in MLpAI:

- **Data pipelines:** The data pipelines can be batch or online pipelines. Refer to Step 2 for additional information.
- **Distributed processing:** Due to the computationally intensive training stage, a distributed processing system is recommended in order to take full advantage of multiple processors, which are needed for training models necessary to support MLpAI.
- **Model repository:** The model repository is responsible for maintaining the metadata needed to support and execute models and to store the models themselves. It also provides a method for sharing and collaborating on models prior to elevating them into a higher environment.
- **Distributed storage:** The output of models is often used as input for other models. As a result, the output of MLpAI should also take advantage of multiple processors by using distributed storage technologies to store the MLpAI output.
- **Development environment:** A development environment is needed to support the CDS in developing models and applying data to support algorithms. Development environments typically contain the training environments, but they can be separated to support large-scale implementations.
- **Testing environment:** Testing MLpAI involves validation of results and verification of prediction or inference. Validation typically involves comparing the variations of the results to determine the efficacy of the model prior to going into production (e.g., hyperparameter optimization). Verification involves leveraging domain knowledge experts to ensure that the inference or prediction

supports the business requirement. The environment for testing typically includes randomly generated data used to test the model for accuracy and efficiency.

- **Production environment:** In contrast to the testing and development environments, production environments include live data producing executable models in a controlled setting.

Areas that require significant attention are distributed processing and distributed storage. Distributed processing is used to facilitate the training environments, and distributed storage can be used to maintain the output of ML models and facilitate the feedback loop process to aid in the refinement of DS workflows and stages. This separation is essential due to the behavior of the training environments and the resources needed to support them.

7.1 Compute-Intensive vs. Data-Intensive Components in Workflows

Certain stages within the DS pipeline are more computationally intensive and data-intensive than others. Operationalizing and maintaining MLpAI requires an understanding of those components and how to mitigate the risks they present. The framework outlined in Figure 14 provides a method for mitigating the risk of compute-intensive and data-intensive stages, and it enables the creation of an architecture to support the data strategy. In fact, many of the vendors offering platforms to support MLpAI focus on two areas:

- Computationally intense training
- Computationally intense inference

Gartner recommends deploying serverless training environments (i.e., distributed processing) for ML training requirements due to the need for elasticity and their compute-intensive requirements. For more information, see "Preparing and Architecting for Machine Learning." (https://www.gartner.com/document/code/317328?ref=grbody&refval=3734817)

Gartner recommends referencing the framework outlined in Figure 14 when developing the data strategy for MLpAI applications.

7.2 Securing Data Science Pipelines

Securing data at rest and in motion remains a critical component to any data strategy. This is especially the case with strategies to support MLpAI because of the amount of processing time devoted to I/O and manipulation of data. Therefore, Gartner recommends establishing a path to securing data pipelines as an addition to your data strategy.

For more information on securing the data pipeline for MLpAI, refer to "Securing the Big Data and Advanced Analytics Pipeline." (https://www.gartner.com/document/code/313004?ref=grbody&refval=3734817)

Follow-Up

When you start the journey of developing a data strategy for MLpAI applications, do not attempt to complete the process in one step. The process for developing the strategy is iterative, and you may end up updating DS workflows and stages as you discover new information. As a result, in getting started, be sure to repeat the process enough to gain economies of scale, and make the strategy flexible enough to support diverse use cases.

Introducing DevOps to MLpAI and Vice Versa

> MLpAI needs DevOps, and DevOps may want to leverage MLpAI.

The most critical part of following up on this guidance is to adopt a methodology of continuous improvement on your data strategy within your teams. Encourage the regular review of the strategy and its evolution. Developing a data strategy for MLpAI applications is a marathon. Therefore, you should treat the strategy as an organic document that learns and matures over time.

DevOps offers a methodology that allows for automation and collaboration between data science citizens in development and the technical professionals assigned with maintaining the systems used by the data scientists. DevOps has long been an established process for managing software applications in production. However, it is not typically inclusive of the rising citizen data scientists. DevOps can help the citizen data scientist deal with inefficiencies as a result of tuning and updating algorithms and the bottlenecks resulting from preprocessing and postprocessing.

You should also consider DevOps as a rising citizen to include in your data strategy. MLpAI is often used extensively in testing the resiliency of an application prior to moving it into a production environment. Much of the data used in DevOps initiatives includes machine data (e.g., log files, sensor data and authorization tokens). As a result, Gartner recommends implementing a bidirectional approach to working with DevOps where the data strategy is included in the DevOps plan.

Risks and Pitfalls

Developing a data strategy for MLpAI applications is a dynamic approach that is not without risk and pitfalls. The following is a set of risks and pitfalls to be aware of:

Risk No. 1: Building DS Pipelines Can Be Especially Challenging When Dealing With Big Data Without the Right Tools

Additional challenges will inevitably emerge when dealing with big data, and traditional ETL tools will not satisfy the DS requirements. Common scenarios include providing a learning algorithm with training data to learn from, as in the case of supervised learning, or when the data that needs to be subject to the clustering algorithm, in the case of a segmentation problem, is large. This makes processing stages, such as profiling, parsing, cleansing, normalizing, standardizing and extracting features from large datasets, extremely time-consuming without the right tools.

Don't consider the availability of tools before developing the data strategy. Moreover, when you've developed the data strategy and you are in the process of searching for tools to support the strategy, avoid looking for one tool to do everything. Many existing AI and ML projects involve using a series of tools when building a data strategy. Your data strategy should focus on the vision for tackling big data, not the procurement of tools. Proper tools will emerge once the data strategy is solidified.

Risk No. 2: Poor Data Quality Will Significantly Impact Performance and Accuracy

The performance and accuracy of MLpAI applications is significantly based on the quality of the data. Poor data quality or bad/incomplete or missing data impacts the accuracy of the predictions and inference made by the model. For example, in image classification programs, poor image quality may result in the extraction of improper features, leading to incorrect predictions or recognitions.

Implement the hub-and-spoke approach outlined in Figure 13 that centers on data quality and governance to ensure that data is being governed for quality as DS pipelines are being developed. This also aids in the alignment with the enterprise data strategy and avoids developing the strategy in isolations.

Risk No. 3: Techniques for Securing DS pipelines Are Still in Their Infancy

Securing the pipelines is not introduced in this research due to significant strategic and tactical approaches required to secure the ML and AI pipelines. The need for securing data pipelines is significant; however, tools and techniques for securing pipeline are still in their infancy. As noted in "Securing the Big Data and Advanced Analytics Pipeline," (https://www.gartner.com/document/code/313004?ref=grbody&refval=3734817) "the ability to provide segregation of duties remains a critical control to prevent adversaries from gaining perfect knowledge of the AI/ML workflow and gaining nearly unlimited opportunities for direct attacks."

Gartner recommends working with administers of your security policy to ensure pipelines will be properly secured prior to completing the data strategy. For more information on securing the pipelines, please refer to Gartner reading "Securing the Big Data and Advanced Analytics Pipeline." (https://www.gartner.com/document/code/313004?ref=grbody&refval=3734817)

Pitfall: Bounded Rationality Exists Even Within MLpAI Applications

Similar to the cognitive limitations of our minds, MLpAI applications have limitations based on their information architecture. Therefore, the ability to make accurate predictions and inferences (i.e., decisions) is based on the scalability of the information architecture (as outlined in red in Figure 1). This places significant pressure on technical professionals to deliver a data strategy that enables the most optimal output.

To avoid being bounded or constrained by your information architecture, technical professionals should continuously refine and update their data strategy to grow as the data grows. As more data is introduced into the environments, the data strategy should be re-examined and refined to include new techniques for enhancing the pipelines and delivering data to citizen data scientists or human-to-computer interaction components.

Evidence

[1]"Artificial Intelligence Primer for 2017" (https://www.gartner.com/document/code/318582?ref=grbody&refval=3734817)

[2]"A Few Useful Things to Know About Machine Learning," (https://cacm.acm.org/magazines/2012/10/155531-a-few-useful-things-to-know-about-machine-learning/abstract) Communications of the ACM.

[3]"Unsupervised Feature Learning and Deep Learning," (https://www.csee.umbc.edu/courses/graduate/678/spring15/visionaudio.pdf) Andrew Ng.

[4] "Analyzing Multivariate Data," J.M. Lattin, J.D. Carroll and P. Green.

Note 1
Complications With ML Data in the Enterprise Data Strategy

MLpAI can be used to analyze large sets of data much more efficiently than humans. But developing MLpAI to perform such tasks requires untraditional processing on different types of data. Traditional enterprise data strategies look to acquire, organize, analyze and delivery. However, MLpAI requires additional steps within the strategy due to the nonlinearity of the data and the multimedia data types that must be integrated with traditional number and text data types for processing.

Note 2
Test Data Management for MLpAI

Test data management (TDM) is an emerging technique often used to aid in the IV&V process. The purpose of the TDM for MLpAI is to validate the training datasets being used by the models, and it offers the ability to generate evaluation data to aid in the verification process. There are a growing number of tools offering TDM that are used to support MLpAI (e.g., Informatica Test Data Management Tool).

For more information on using TDM to support MLpAI, see "Magic Quadrant for Software Test Automation." (https://www.gartner.com/document/code/297494?ref=grbody&refval=3734817)

Recommended by the Author

Preparing and Architecting for Machine Learning (https://www.gartner.com/document/code/317328?ref=ggrec&refval=3734817)

Securing the Big Data and Advanced Analytics Pipeline (https://www.gartner.com/document/code/313004?ref=ggrec&refval=3734817)

Recommended For You

Making Machine Learning a Scalable Enterprise Reality — From Development to Production (https://www.gartner.com/document/3830149?ref=ddrec&refval=3734817)