# ModelOps RFP Requirements

**This document is an example RFP for addressing ModelOps (and MLOps) functional requirements. It is the result of interviews with several industry experts and analysts.**

**ModelOps (and its MLOps subset which focus on ML models only) is a key capability that is required for successful AI/ML model operations once models have been developed. It is a discipline that is separate and apart from model development. Industry experts and analysts are recognizing that model development and model operations are different disciplines, requiring different capabilities, tools and even teams. Gartner, in a recent article, states, "Platform independence: AI pipelines span multiple environments from developer notebooks to edge to data center to cloud deployments. A true ModelOps framework allows you to bring standardization and scalability across these disparate environments so that development, training and deployment processes can run consistently and in a platform-agnostic manner." Gartner report "Innovation Insight for ModelOps", Farhan Choudhary, Shubhangi Vashisth, Arun Chandrasekaran, Erick Brethenoux. 6 August 2020.**

| Category | Requirement | Description |
|---|---|---|
| **Model Management**<br>suggested weighting: 20% | | |
| | **Model Implementation Agnosticity** | Supports the operationalization of all models, inclusive of:<br>- All Model Types: AI, ML, mathematical optimization, rules-based, etc.<br>- All Model Languages (and associated Frameworks): Java, Python, R, C/C++, Scala, Matlab, etc.<br>- All Model Execution Environments: Kubernetes/Docker, Java, C/C++, Matlab, SageMaker, Azure ML, GCP, Spark, etc.<br>- All Model Execution Locations: On-Prem, Cloud, Hybrid |
| | **Standard Model Definition** | Provide a consistent definition (and underlying persistence mechanism) of all the core elements that compose a model, regardless of the language/framework, Data Science workbench, underlying infrastructure, or data platform used. |
| | **Custom Metadata** | Allow for extension of the core Standard Model definition with custom metadata, which may come from various integrating systems or via user input. Custom metadata may be supplied for a model as a whole or for specific snapshots of that model. The custom metadata must be available to be used in a Model Life Cycle to allow an enterprise to define and enforce governance, technical, and business requirements that are specific to their processes. |
| | **Production Model Inventory** | Provide a centralized store for viewing, managing, and maintaining all models (post-development) across the enterprise, regardless of the model type, framework, platform, or environment. |
| | **Model Registration** | Allow users to onboard their developed model into the ModelOps system by collecting the key elements that compose a model. Model registration should be enabled via CLI, import from a git repository, or via model factory plugins, such as a Jupyter or Rstudio plugin. |
| | **Model Artifact Management** | Collect and manage all model "artifacts" with the model metadata as part of the "Standard Model Definition" for the registered model. Depending on the type of model that the data scientist uses, the "model training" will result in a set of outputs, which can be coefficients, weights files, binary objects, etc. These are critical parts of the "model" and therefore must be managed in conjunction with the model code. Additionally, the artifact management capabiltiy must integrate with common file/binary artifact storage mechanisms such as an Object Store (S3) to provide for scalability and manageability, while also allowing an enterprise to leverage their existing IT investments. |
| | **Model Documentation Management Integration** | Ability to persist model-specific documdntation and/or link to existing content management systems that contain model documentation. Documentation should be snapshotted with all other model assets such that there is an audit trail to the specific version of the documentation for that version of the model. |
| | **Dependency Management Integration** | Manage the list of frameworks, libraries, and other dependencies that are required to execute a given model. While the actual libraries are typically managed in an existing repository (e.g. Artifactory), the ModelOps capability needs to manage--with each version of the model--the exact libraries/frameworks--and the version of the library--for the model. Ensure that the model is deployed to an available execution context that has the required libraries to run the model. |

| | | |
|---|---|---|
| | **Source Code Management Integration** | Integrate with the enterprise's standard for source code management (Bitbucket). Associate details of the model's git assets as metadata with the standard model definition. |
| | **Test Results Management** | Persist all instances of tests/metrics that have been executed on a per-model basis, regardless of the type of model or the platform upon which it executes. A "record" of a test/metrics job should be tied to an immutable snapshot of a model for history and auditability. |
| | **Associated Models** | Ability to define associations between models, which may be focused on reusable monitoring "models" (e.g. drift models) across multiple models; or could be associating multiple models in a parent child or linked ensemble approach. |
| **Model Life Cycle**<br>suggested weighting: 25% | | |
| | **Model Life Cycle (MLC) Design** | Ability to design and build holistic Model Life Cycles (MLC's) that incorporate business, technical, and compliance/governance requirements across the entire life of a model to allow orchestration and management of the entire AI Governance workflow, from registration to testing, promotion, business approvals, governance approvals, technical approvals, deployment, monitoring, improvement, and eventual retirement. Must provide the ability to create different Model Life Cycles (MLC's) for different groups, classes of models, or even on a per model basis. |
| | **Model Life Cycle (MLC) Integration** | Ability to integrate with existing operational systems (e.g. Production Support CR/IM ticketing, alerting systems, etc.) and with existing Governance systems (e.g. MRM/Compliance systems). The integration must be systematic (via API's/other) to allow for automation of the overall Model Life Cycle. Must provide the ability to systematically collect information/metadata from the integrating systems and persist as metadata with the specific version of the model. |
| | **Model Life Cycle (MLC) Automation** | Manage and automate the entire life cycle of a model--from model registration to testing, promotion, reviews, approvals, etc. Integrate with the underlying standard IT systems (ticketing, security scanning, etc.), MRM/compliance systems, and other AI Governance processes/systems to manage and orchestrate the overall AI governance workflow. Additionally, allow for the creation of custom MLCs to cater to the differing needs of each team, business unit, or even types of models. |
| | **Model Life Cycle (MLC) Lineage** | Collect and persist each step in the model's life cycle for full auditability for each and every version of a model. This includes all of the steps taken to deploy a model to production, including all approvals, promotions, and other operational controls; as well as all steps involved in running on-going production operational controls, monitors, and metrics. |
| | **Model Life Cycle (MLC) Reusable Operational Patterns** | Built-in customized Model Life Cycle definitions, which are prepackaged automated MLCs to allow for reusability across the enterprise for multiple use-cases. These include MLC's for model registration, model approvals and operationalization, on-going compliance enforcement, on-going comprehensive monitoring, production issue remediation, etc. These MLC's specifically focus on enterprise requirements for compliance enforcement as well as remediation paths for production operational or compliance issues. |
| | **Model Approvals (technical, business and compliance)** | Automate the specific approval process(es) required to get a model into business. Integrate with the enterprise's standard change and incident management systems (e.g. ServiceNow) as well as task ticketing systems (e.g. JIRA), and Model Risk Management (MRM)/Compliance systems, to automate and track approvals across all business, data science, operations and IT teams. These approvals must be added as metadata to the model snapshot for traceability. |
| | **Unencoded Thresholds Definitions** | Ability to define--via decision tables or other rules--specific thresholds within which a model should operate for various points in its Model Life Cycle. For example, thresholds may be set for matching compliance rules back to statistical performance to provide clear but comprehensive rules for operating the model within the bounds of the compliance and business requirements. Thresholds definitions do not require programming skillset to be decoded. |
| | **Model Compliance with Security Standards** | Integrate with enterprise security scanning software (e.g. SonarQube) and/or code scanning (e.g. Veracode) to streamline operationalization of a given model. Additionally, allow for automation of the security scanning process as part of a given Model Life Cycle. |

| | | |
|---|---|---|
| **Model Governance**<br>suggested weighting: 25% | | |
| | **Model "Snapshots" ("Versions")** | Provide the ability to systematically take a snapshot of a model in time, including all of the model's source code, artifacts, documentation, and other metadata. This snapshot must be immutable and maintained in perpituity for long-term auditability. Each Production model must have traceability to a specific immutable snapshot. The snapshot--and all of the associate metadata for that model snapshot--can be exported for reporting purposes. |
| | **Per Model Comprehensive Audit Trail** | Support full auditability for each and every version of a model. Capture each step of the entire process to deploy a model to production, including all approvals, promotions, and other operational controls; as well as all steps involved in running on-going production operational controls, monitors, and metrics. |
| | **Model Audit Report** | Ability to generate, as required, an Audit Report (document) that provides a detailed history of the model, including all steps in the model life cycle, as well as related metadata, artifacts, documentation, and test results. |
| | **Model Reproducibility** | Allow for rapid systematic reproduction of training, evaluation, and/or scoring of a particular model version, especially of models that are currently or previously deployed in Production. The specific version of the relevant code, artifacts, and other model assets must be used to conduct the reproducibility test. |
| | **Continuous Compliance Checking - Define Rules** | Ability to define--via decision tables--specific conditions in which a model must operate to be within compliance. Must be able to define the rules on a per-model basis which are persisted, maintained, and orchestrated with each and every version of a given model. These rules must be able to incorporate custom metadata for a given model, which may have been pulled in automatically via an MRM/compliance/other governance system. |
| | **Continuous Compliance Checking - Enforcing Rules** | Automated enforcement of all compliance/governance rules that have been defined on a per model basis. Breach of any particular rule can trigger alerts/notifications as dictated by the Model Life Cycle. |
| | **Interpretability Integration** | Integrate with leading frameworks (Shap, KLime, etc.) to provide a consistent and automated approach to identifying feature importance for models for a given version of a model. |
| | **Ethical Fairness/Bias Integration** | Integrate with leading frameworks (Aequitas) to measure and monitor a model's fairness in providing positive outcomes for all protected and/or sensitive classes. |
| **Model Execution & Monitoring**<br>suggested weighting: 15% | | |
| | **Support for multiple model execution platforms ("runtimes")** | Integrate with multiple model execution (DSML) platforms, providing automated orchestration of jobs, deployment, and monitoring. The integration should occur regardless of the underlying framework (scikitlearn, R, xgboost, TensorFlow, pytorch, etc.) and language (R, python, Scala, C, etc.). The capability should support all existing models--whether they are machine learning, statistical, or even rules-based and regardless of where the model execute: Cloud, on Prem, Hybrid. |
| | **Support for multiple modes of scoring: Batch, Request/Response, Streaming** | Supports batch, request/response (e.g. REST), and streaming model execution across platforms. Pending the runtime's capability, switching of execution modes should be possible without changing the underlying algorithm (the "math"). Support switching of scoring modes based on the current stage of model development without changing the model's math: e.g. during early development, a data scientist may want to quickly test using a REST deployment; however, during testing, UAT, and production, they actually need to execute the model in batch due to data volumes. |
| | **Job Orchestration & Monitoring** | For all models in the ModelOps production model inventory, provide overarching job orchestratration and monitoring, regardless of the model execution platform. In particular, a "record" of a job--and any available status, metrics, and results of that job--should be tied to an immutable snapshot of a model for auditability. |

| | | |
|---|---|---|
| | **Alerting and Automated Notifications** | Provide ability to automatically raise alerts and notifications based on model-execution errors, issues from model compliance/other monitoring, or from specific actions/steps in the model life cycle. Typically these alerts are handled by orchestrated remediation paths that are defined in a model life cycle (MLC), and would be tied to thresholds (see below) to allow for clear, but comprehensive, orchestration of when alerts are triggered. The ModelOps capability should be flexible to integrate with existing operational alerting and/or production ticketing systems to enable L1 support to provide 24x7 production support for all models across the enterprise. |
| | **ModelOps Life Cycle KPI / Process Monitoring** | Collect and aggregate detailed metrics about Model Life Cycle execution across the enterprise, including specific times for each step in a Model Life Cycle, allowing for understanding of where there are bottlenecks in the process so that they can address accordingly. |
| | **Standard Monitors** | Enable the registration, orchestration, and tracking of comprehensive model monitors for all Production Models and/or Production candidates, including:<br>- Data Drift Monitoring<br>- Volumetric Monitoring<br>- Model Concept Drift Monitoring<br>- Statistical Performance Monitoring |
| | **Compliance Monitors** | Enable the registration, orchestration, and tracking of comprehensive compliance-focused monitors and controls for all Production Models and/or Production candidates, including:<br>- Characteristic Stability<br>- Population Stability Index<br>- Rank Order Break<br>- Ethical Fairness Drift |
| | **Custom Metrics Monitoring** | Allow a data scientist or ModelOps engineer to define custom statistics to be captured about the model. |
| | **Model-Specific Thresholds** | The user must be able to set model-specific thresholds to evaluate if the monitoring metrics are within the expected range. Typically, this includes comparisons to baseline data (e.g. compare a current window of production inference requests to the training data input feature distribution). Per above, the system must be capable of triggering automated alerts if thresholds are exceeded. |
| | **Trigger model refresh or retraining based on monitoring alerts** | Allow users to monitor specific statistics about a model, and based on pre-defined thresholds, trigger an automated re-training process if those thresholds are exceeded. |
| | **Champion/Challenger** | Allow for the comparison of the current "champion" production model against a proposed “better” model, using a variety of statistical and technical performance metrics (e.g. AUC, RMSE, F1/F2, ROC Curve, Confusion Matrix). |
| **Visualization**<br>suggested weighting: 5% | | |
| | **ModelOps Console** | A robust ModelOps Console that provides the ability to manage all production models, Model Life Cycles, and model runtimes across the enterprise, as well as detailed model metrics such as statistical metrics, drift metrics, and bias metrics. |
| | **Visualization: BI tools integrations** | Ability to integrate with BI tools to create custom reports to gain visibility of aggregated and detailed views on model performance against statistical, business, and risk thresholds. |
| | **ModelOps Executive BI Dashboard Templates** | Out-of-the-box ModelOps BI dashboard templates that can be used and customized for executives that are accustomed to using BI visualizations for KPIs and Scorecards |
| **Security**<br>suggested weighting: 5% | | |
| | **Authentication/Authorization** | Integration with the enterprise's oauth2 and LDAP/AD systems for authentication, leveraging existing oauth/ldap systems, processes, and AD/LDAP group structure to for authentication |
| | **Role-Based Access Control** | Role-based access control to expose functionality according to the user's role (e.g. Admin vs. Non-Admin), which should be obtained based on group membership within the organization's AD/LDAP |

| | | |
|---|---|---|
| | **Multi-Tenancy (Group-based isoloatio** | Group-based access control to isolate which groups/teams can see specific models, model assets, runtimes with specific deployed models, test results, integrating with the enterprise's oauth2 and LDAP/AD systems to obtain the associated group entity information. |
| | **Model Privacy** | All privacy aspects for the Model including "policy that the model was built, retrained, validated with data that adhered to the corp data privacy requirements" (this can be call out to Data Governance); assurance that the production scores adhere to data privacy policy (e.g. opaque IDs in the scores); when there is a one to one relationship of model to person (e.g. individual customer behavior model) that the appropriate policies are enforced. |
| **Non-Functional Requirements** suggested weighting: 5% | | |
| | **Product Architecture** | ModelOps capability should be designed to be flexible and extensible, using a modern micro services architecture, to allow enterprises to customize and integrate the ModelOps services into existing systems, platforms, and processes. |
| | **Infrastructure** | Ability to run Software anywhere: on Cloud, on Prem, Hybrid. |
| | **Systematic Access (API's)** | Provide a comprehensive API to allow enterprises to create custom Model Life Cycles or custom integrations into consuming business applications. Expose core ModelOps services as RESTful API's allowing enterprises to easily customize the use of the ModelOps solution to their specific business requirements. |
| | **Ownership** | Allow for definition of ownership for model operationalization, by providing a console for 24x7 model management in production, and with visibility into the model path to/in production with pre-defined ownership of responsibilities, triage and remediation processes. |
| | **Localization** | Support for the linguistic, technical, and overall cultural requirements for all regions of a global enterprise |