MLOps - MindTickle

September 2021

Contents

The goal of this course is to give you an overview of what is offered in DataRobot's MLOps platform. We balance depth with breadth, in order to give you a better understanding of the power of MLOps. Hopefully, you will read about a feature that inspires you to dive deeply into that topic and learn more about it using other resources.

In this course, we will cover the following:

- The Problem
- Challenges Solved with MLOps: *get more models in production*
- Model Deployment: *test your ideas & unlock value more quickly*
- Performance Monitoring: *detect decay in model performance in real-time*
- Governance: *track model version history*

1 The Problem

Building a machine learning (ML) system that interminably operates in production is challenging. Requiring extensive time for development, stress-testing and collaboration among different teams within an organization, each with different skill sets and competing demands on their time. These differences in skills, motivations and competing constraints pose challenging obstacles to successful model deployment. Industry surveys show that many of these artisan, hand-crafted models are shelved, an unfortunately common tragedy and a lost opportunity.

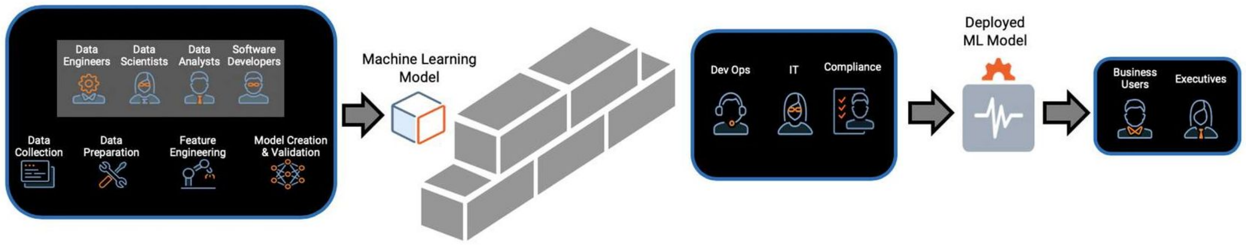


Figure 1: This infographic visualizes the stakeholders involved in moving an ML model from ideation, to creation, to deployment.

1.1 Challenges of Putting Models into Production

ML deployments can incur massive ongoing maintenance costs at the system level. Let's look at some of the challenges you may encounter:

A Data Science team has developed a model using Python but IT infrastructure requires the model to be written in Java, a language that the former is only superficially familiar with though one most Data Scientists have no exposure to. After some time struggling, they produce the necessary code for IT and it is put into real-world production. After several months, the model starts to perform poorly. No one knows why and especially not the end users. There is no accessible record of model updates or model-based performance at the Operations level.

This example highlights several ML deployment points of failure: * The data science and IT teams failed to communicate effectively * The variety of skills and ML tools across teams made collaboration between the data science and IT teams difficult * ML model behavior is difficult to track with traditional software tools * Models have a complex life cycle that is costly to ignore, but also costly to manage with manual updates * Governance is required to minimize risk and ensure compliance with regulations

2 DataRobot MLOps as a Solution

DataRobot MLOps provides a central hub to deploy, monitor, manage, and govern all your models in production, regardless of how they were created or when and where they were deployed. MLOps maintains the quality of your models using health monitoring that accommodates changing conditions via continuous, automated model “gladiator” competitions using **challenger models**. **MLOps** also ensures that all centralized production machine learning processes work under a robust governance framework across your organization, leveraging and sharing the burden of production model management.

With MLOps, you can deploy any model to your production environment of choice. By instrumenting the MLOps agent, you can monitor any existing production model already deployed for live updates on behavior and performance from a single and centralized machine learning operations system. MLOps makes it easy to deploy models written in any open-source language or library and expose a production-quality, REST API to support real-time or batch predictions. MLOps also offers built-in, write-back integrations to systems such as Snowflake and Tableau.

MLOps provides constant monitoring and production diagnostics to improve the performance of your existing models. Automated best practices enable you to track service health, accuracy and data drift to explain why your model is degrading. You can build your own challenger models or use DataRobot’s Automated Machine Learning to build them for you and test them against your current model champion. This process of continuous learning and evaluation enables you to avoid surprise changes in model performance.

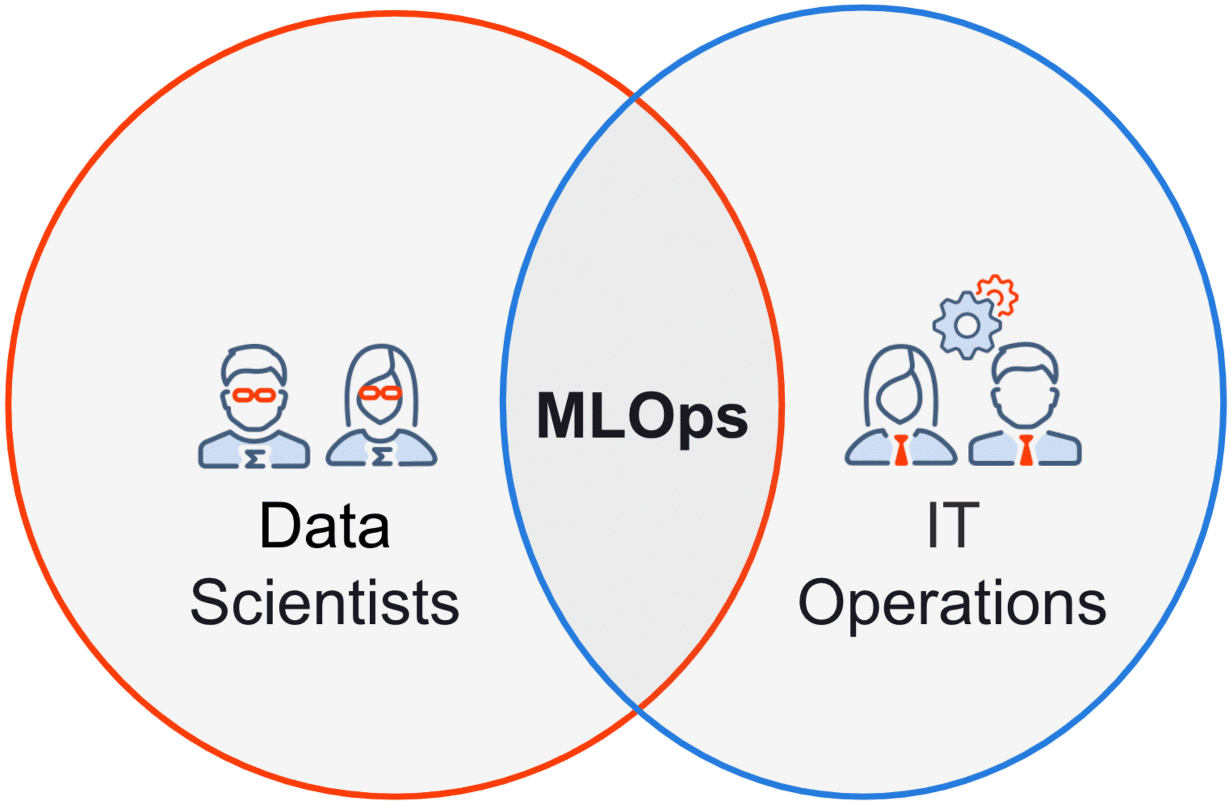


Figure 2: MLOps bridges the gap with Data Science and IT/Engineering skill sets, allowing infinitely easier productionization of results and freeing up IT/Engineering time for other pursuits. Note: "MLOps" is an industry term as well as the name of the DataRobot MLOps product. For the rest of this course, "MLOps" will refer to DataRobot MLOps.

2.1 Deployment

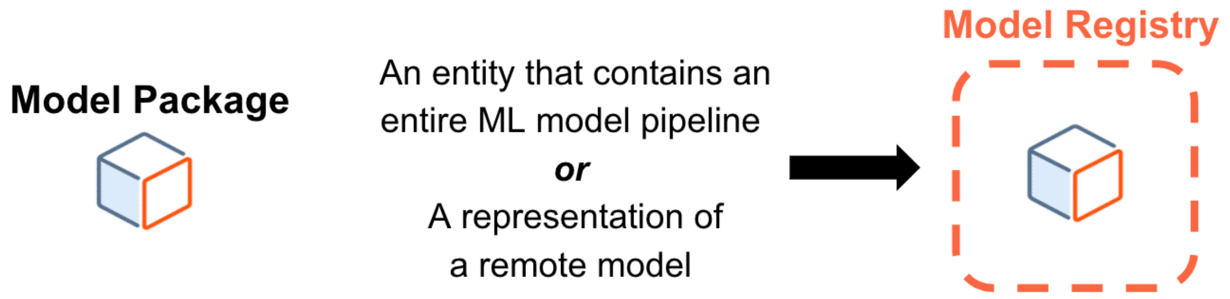
Although deploying ML models creates strong value, only a small percentage of ML projects make it to production because integration into production environments can be complex. Many challenging problems do not arise until the model is deployed. Often, the real modeling work begins in production.

Yet to trust a model to power mission-critical operations, users need to have confidence in all aspects of model deployment. Many models never generate value because they never reach production. In the end, data scientists have to help deploy and maintain their models, which is costly and takes away from doing new data science.

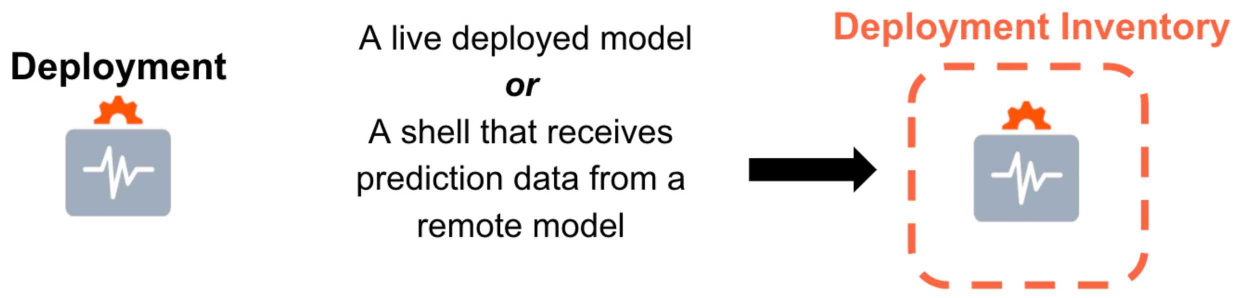
With MLOps, the goal is to make model deployment easy. Operations teams, not data scientists, can deploy models written in a variety of modern programming languages like Python and R onto modern runtime environments in the cloud or on-premise.

There are two components to successful model deployment: 1. creating the deployment and then 2. monitoring and managing it. With DataRobot MLOps, you can create deployments in a variety of ways depending on the artifact you want to deploy.

- Before we talk about deployment options, let's align on some definitions:
- **a DataRobot model / "MLDev model"**: one built using the DataRobot Platform. These are models built within the DataRobot AutoML platform, then exported as MLPKG files ('model package files') that your firm has full ownership of.
- **a model package**: A Model Package is an archived model artifact with associated metadata and can contain an entire model pipeline to be used in a new live deployment, or represent a remote model deployed on a platform outside of DataRobot. Regardless of their origin, MLOps manages all models as model packages with the same functionality, maintained in the Model Registry.


Model Package
An entity that contains an entire ML model pipeline
or
A representation of a remote model
Model Registry

- **Deployments**:Deployments are managed in the Deployment Inventory and refer to either: a live deployed model or a shell created in MLOps to receive prediction data from a remote model deployed elsewhere.



- **your own "custom model"**: Models built and trained by data scientists outside of DataRobot using programming languages such as Python R, Java, Julia using using packages like scikit-learn, xgboost, PyTorch, tensorflow/keras, caret, h2o, MLJ, etc.

Figure 3: Remote models

- **a "remote model"**: If you opt for this route, you give our "MLOps agent" permission to monitor your deployment and report back model deployment metrics relating to performance, accuracy and health to a dashboard for convenient deployment tracking. With MLOps, you can test, compare, and migrate models from one cloud to another or between cloud and on premises. Using MLOps agents you can also monitor models deployed anywhere.

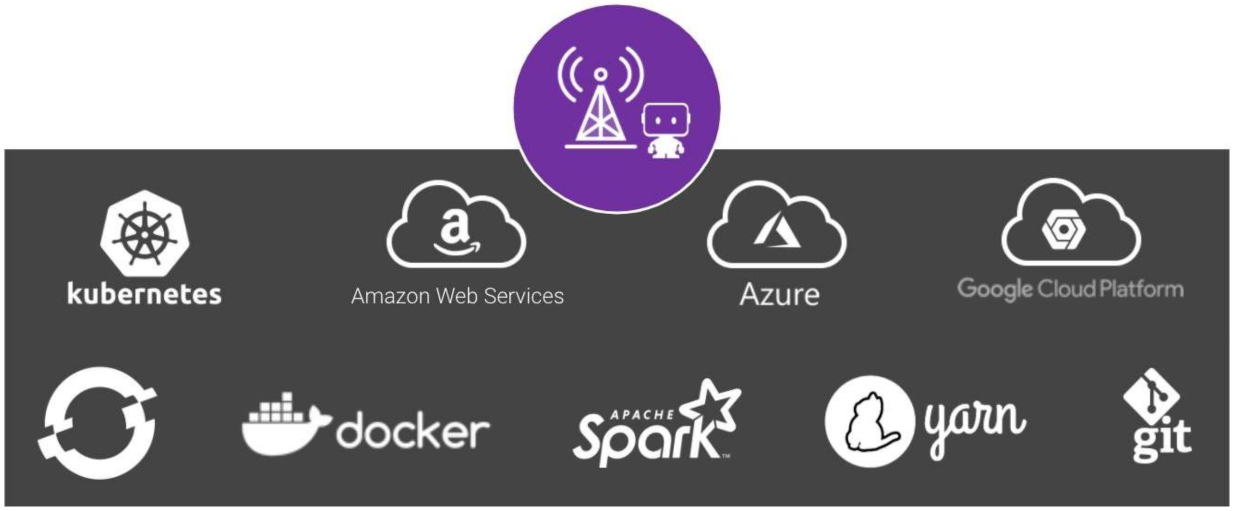


Figure 4: DataRobot's MLOps "agents" can monitor the health of your models, regardless of where they live. These are examples of enviroments our MLOps agent can monitor.

2.1.1 The Model Registry

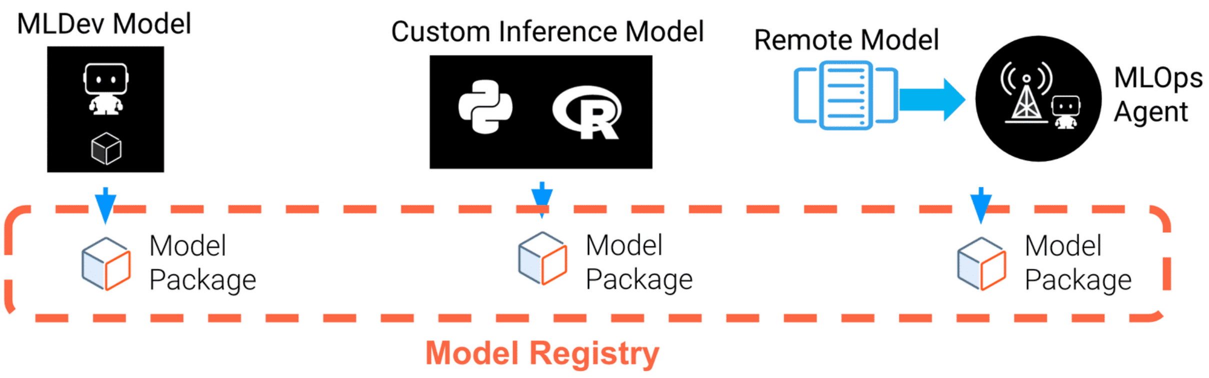
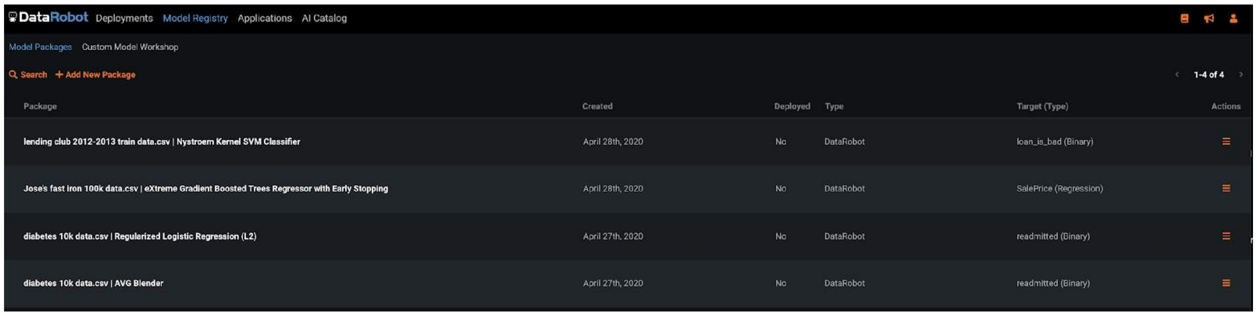


Figure 5: After your model is converted into a model package, it's managed in the Model Registry. The Model Registry is the organizational hub for all models managed in MLOps.

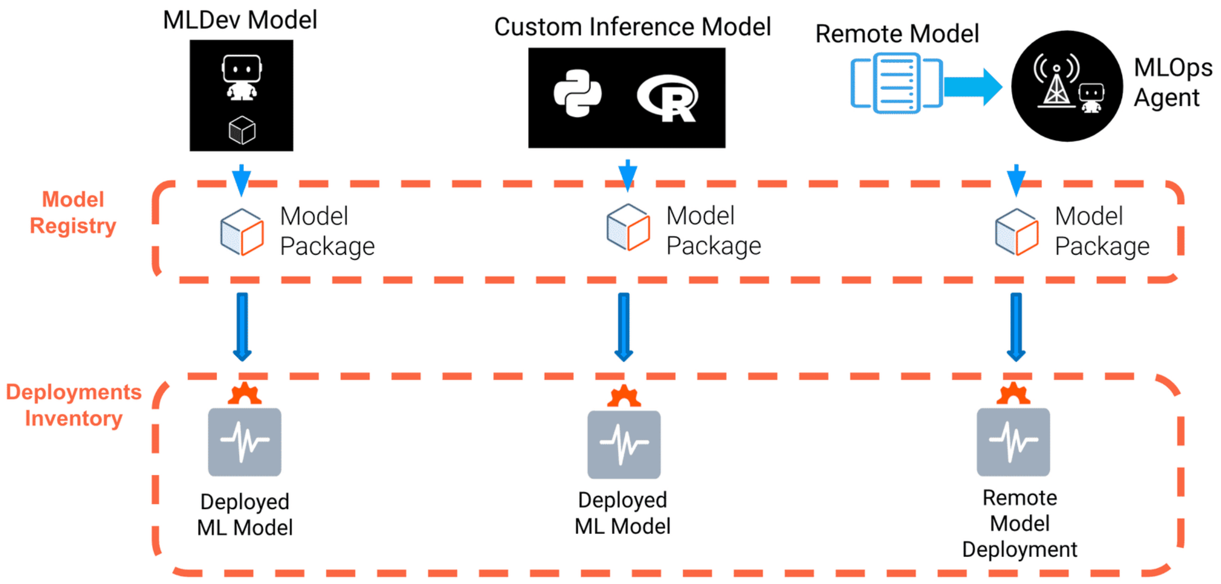


Package	Created	Deployed	Type	Target (Type)	Actions
lending club 2012-2013 train data.csv | Nystroem Kernel SVM Classifier	April 28th, 2020	No	DataRobot	loan_is_bad (Binary)	≡
Jose's fast iron 100k data.csv | eXtreme Gradient Boosted Trees Regressor with Early Stopping	April 28th, 2020	No	DataRobot	SalePrice (Regression)	≡
diabetes 10k data.csv | Regularized Logistic Regression (L2)	April 27th, 2020	No	DataRobot	readmitted (Binary)	≡
diabetes 10k data.csv | AVG Blender	April 27th, 2020	No	DataRobot	readmitted (Binary)	≡

Figure 6: Model packages can be deployed, shared, or archived. The Model Registry helps you manage them, showing you their creation date, deployment status, package type, and the target predicted by the model.

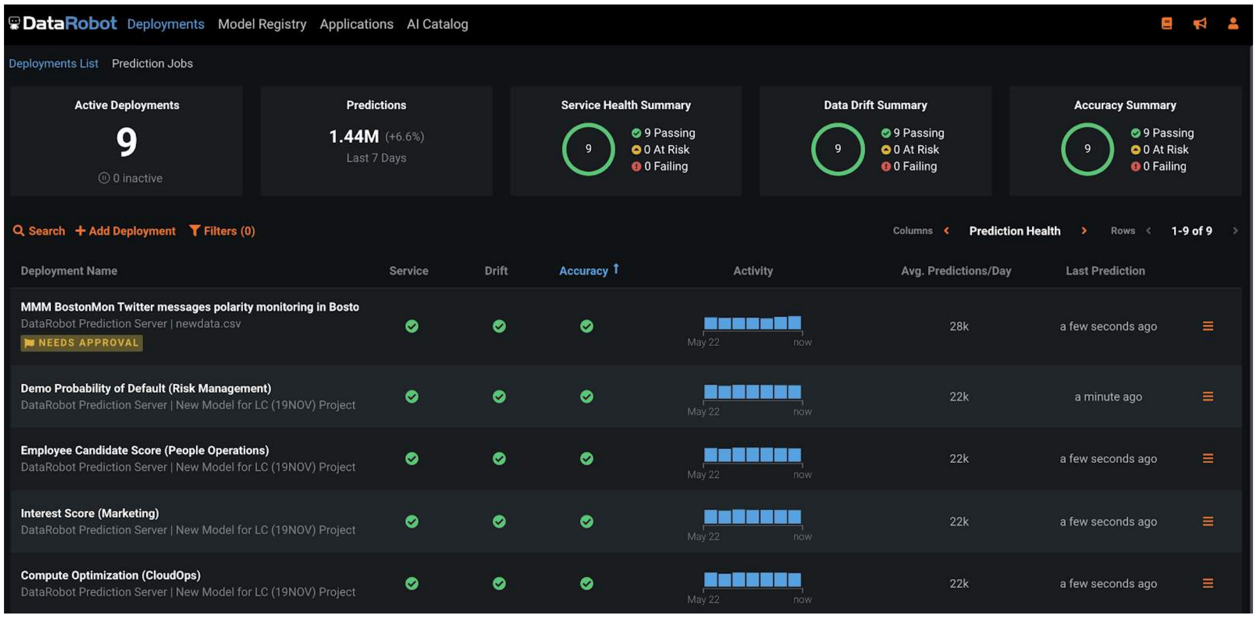
2.1.2 Deployments and Deployments Inventory

Using model packages, any model can be deployed as a DataRobot Deployment. Model packages representing MLDev and Custom Inference models can be deployed from the Model Registry in a few clicks.



You can also create deployments for externally deployed models. The deployment is a shell that receives prediction data from the remote model through the MLOps Agent. In this way the remote model can be monitored just as MLDev and Custom Inference models.

You manage all deployments through the Deployments Inventory. By default, the inventory automatically refreshes every 30 seconds with updates of information gathered during that time.



DataRobot Deployments Model Registry Applications AI Catalog

Deployments List Prediction Jobs

Active Deployments	Predictions	Service Health Summary	Data Drift Summary	Accuracy Summary
9	1.44M (+6.6%)	9	9	9
0 inactive	Last 7 Days	9 Passing, 0 At Risk, 0 Failing	9 Passing, 0 At Risk, 0 Failing	9 Passing, 0 At Risk, 0 Failing

Search + Add Deployment Filters (0) Columns Prediction Health Rows 1-9 of 9

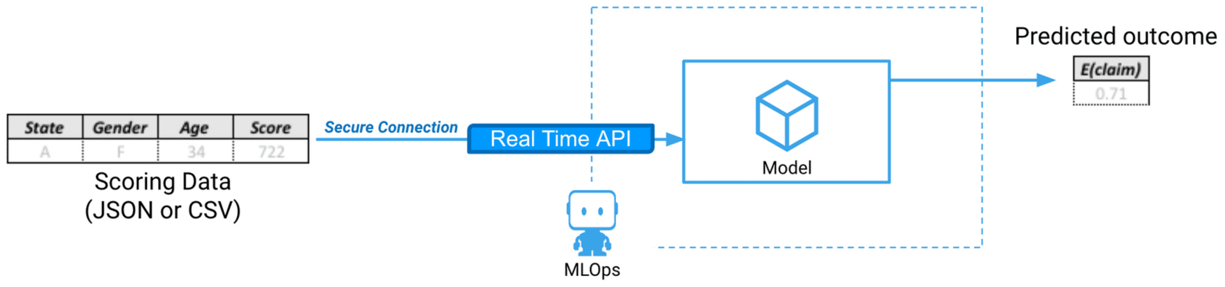
Deployment Name	Service	Drift	Accuracy	Activity	Avg. Predictions/Day	Last Prediction
MMM BostonMon Twitter messages polarity monitoring in Bosto DataRobot Prediction Server | newdata.csv NEEDS APPROVAL	✓	✓	✓	May 22 now	28k	a few seconds ago
Demo Probability of Default (Risk Management) DataRobot Prediction Server | New Model for LC (19NOV) Project	✓	✓	✓	May 22 now	22k	a minute ago
Employee Candidate Score (People Operations) DataRobot Prediction Server | New Model for LC (19NOV) Project	✓	✓	✓	May 22 now	22k	a few seconds ago
Interest Score (Marketing) DataRobot Prediction Server | New Model for LC (19NOV) Project	✓	✓	✓	May 22 now	22k	a few seconds ago
Compute Optimization (CloudOps) DataRobot Prediction Server | New Model for LC (19NOV) Project	✓	✓	✓	May 22 now	22k	a few seconds ago

Figure 7: The "Deployments List" that allows one to view the real-time status of all deployed models.

2.1.3 Making Predictions

You can make predictions from a deployed model using either real-time predictions on single rows of data or a batch file approach.

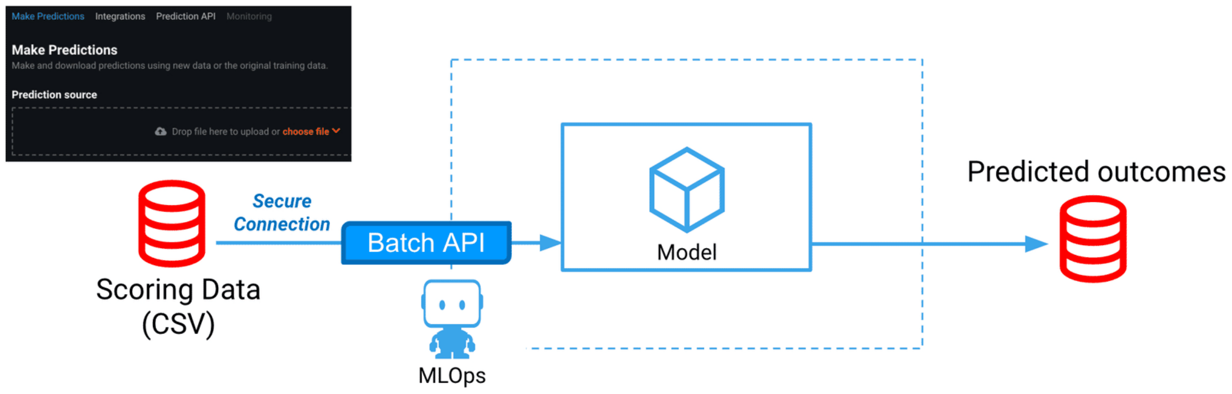
2.1.3.1 Real-time prediction API Prediction explanations give insight into which attributes of a particular input cause it to have exceptionally high or low scores. Prediction warnings alert you to predictions that are outliers. When using the Real Time API, model scoring takes place one data point at a time. You can supply the data to be scored in either JSON or CSV format.



2.1.3.2 Batch prediction API This approach generates unlimited batch predictions on large datasets, typically 100GB and higher. You can stream local files and start scoring while still uploading and simultaneously download the results.

The Batch API also supports prediction explanations and prediction warnings, which can be leveraged using scripts in several programming languages. You can also be generate batch predictions in the

GUI using the Make Predictions interface. When using the Batch API, the data to be scored must be in CSV format.



There are four possible deployment scenarios with starting artifacts, or starting points:

STARTING ARTIFACT	DEPLOYMENT METHOD
DataRobot model	Deploy a DataRobot model by selecting it from the Leaderboard and navigating to the Predict > Deploy tab. Alternatively, upload your model directly to the deployment inventory using the Add Deployment link.
Custom model	Deploy a custom model from the Custom Model Workshop. Alternatively, upload your model directly to the deployment inventory using the Add Deployment link.
Remote model	Upload the external model's training data directly to the deployment inventory to create a deployment. Alternatively, deploy an external model package. Once a deployment is created, you can instruct the MLOps agent to monitor the model and track prediction statistics. You can also upload historical prediction data after the deployment is created.
Historical prediction data	Upload the training data for a model that made predictions in the past directly to the deployment inventory to create an external deployment. Add historical prediction data post-deployment.

2.2 Monitoring

Because the world and the data describing it are constantly changing, ML models start to degrade the moment you put them into production. For this reason, the data that the model sees in production can differ significantly from the data used during training. This is especially true when there is a feedback loop of new decisions being made based on the deployed model's predictions that changes how a business operates.

You must carefully monitor deployed models to minimize any real-world consequences from incorrect model predictions.

MLOps helps you monitor the service health, data drift, and accuracy of your model.

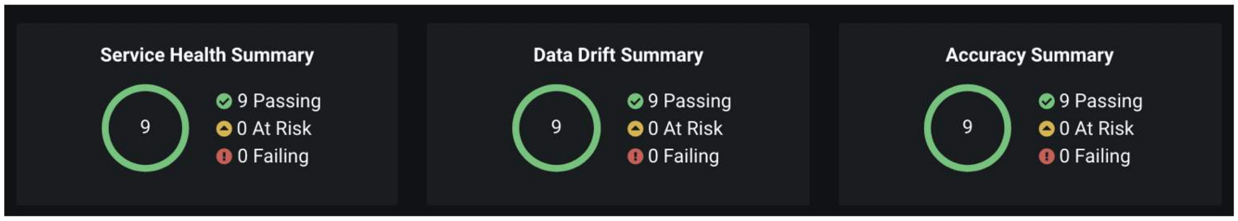
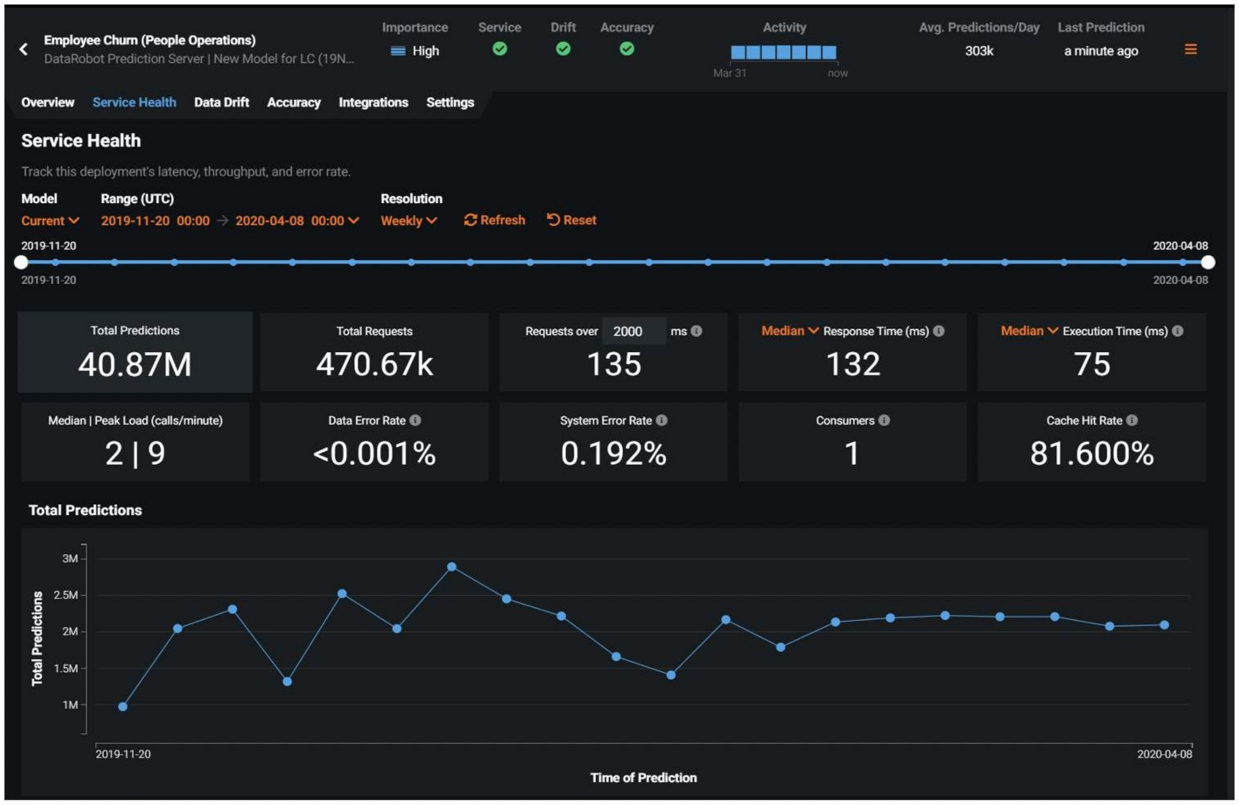


Figure 8: Summaries of Service Health, Data Drift, and Accuracy.

2.2.1 Prediction Service Health

Each deployment has a Service Health tab that enables you to understand model usage and health over time. The Service Health tab includes: total predictions, total requests, response times, and error rates. These metrics keep track of the quality and reliability of the prediction service.



Service Health

Track this deployment's latency, throughput, and error rate.

Model: Current
Range (UTC): 2019-11-20 00:00 → 2020-04-08 00:00
Resolution: Weekly
Refresh
Reset

2019-11-20 — 2020-04-08

Total Predictions	Total Requests	Requests over 2000 ms	Median Response Time (ms)	Median Execution Time (ms)
40.87M	470.67k	135	132	75

Median | Peak Load (calls/minute)	Data Error Rate	System Error Rate	Consumers	Cache Hit Rate
2 | 9	<0.001%	0.192%	1	81.600%

Total Predictions

Figure 9: Data and system error rates can impact the accuracy of predictions, even though they are not directly related to the deployed model itself.

2.2.2 Segment Analysis

Sometimes operational issues with your model come from specific user segments. You can use segment analysis to drill down to the root cause of the issues. Segment analysis identifies operational issues with prediction request data segment-by-segment.

Prediction service requests can be separated into segments by:

- Consumers — users of a deployment that have made predictions requests (segments are tracked by email address).
- Host IPs — the IP address of the prediction server hosting a model.
- Remote IPs — the IP address of a prediction request caller.

In this example you can see that the model consumer segment was selected from the dropdown list. It shows the Data Error Rate for a specific time period made by *another-user@datarobot.com*.

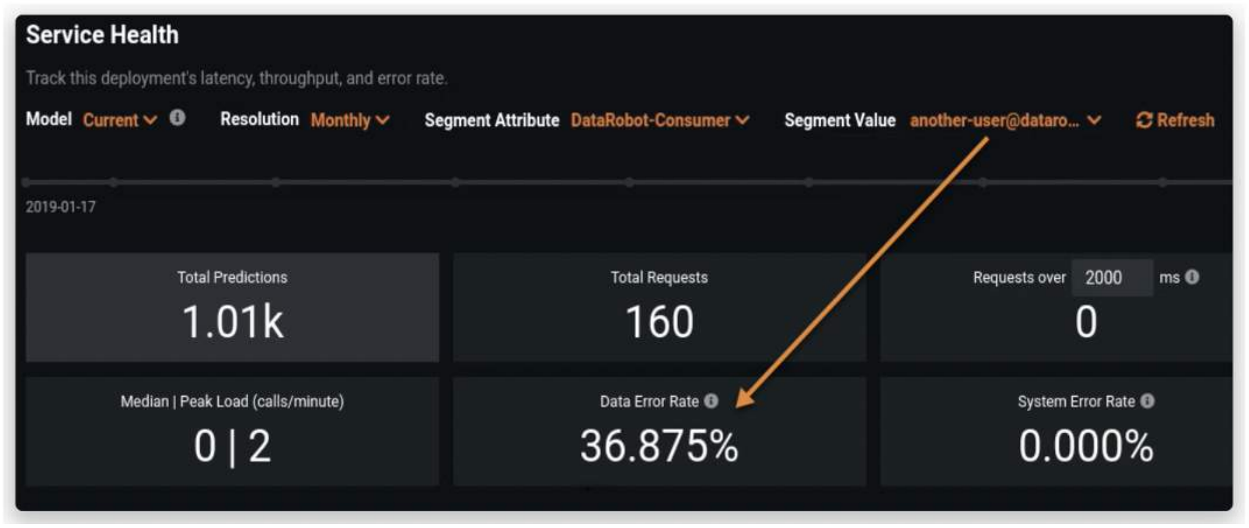


Figure 10: Use Segment Analysis to explore the source of data errors.

DataRobot recomputes all metrics to show you service health statistics based on prediction requests made by a user within the specified time period.

2.2.3 Model Decay

Model Decay refers to a degradation in model performance due to changes over time. When monitoring deployments for model decay, you should track two types of changes in the data being scored by the model: data drift and concept drift.

2.2.3.1 Data drift

Data drift occurs when there are changes in the distribution of values for a given feature. It happens when the statistical distribution of the data that are seen in production is different from the distribution of data used to train the model. For example, an ML model is created to identify loans that are likely to default. As time passes, the income distribution of individual loan applications changes, as do applicant purposes for applying for these loans. This results in changes in the distributions of the income and loan-purpose features during prediction relative to the distribution during training. Consequently, the model's accuracy on current loan applications will also deteriorate.

2.2.3.2 Concept drift

Concept drift occurs when there are changes in the way the target is defined. It happens when the statistical properties of the target change over time in unforeseen ways. Perhaps a model was trained using one definition of the boundary between two classes, but later the definition changed in the data seen in production. As a result, the predictions generated by the model are no longer accurate.

For example, you have a model that was trained on historical data to detect fraud. Then the fraudsters change their strategies, which changed the definition of what is fraud.

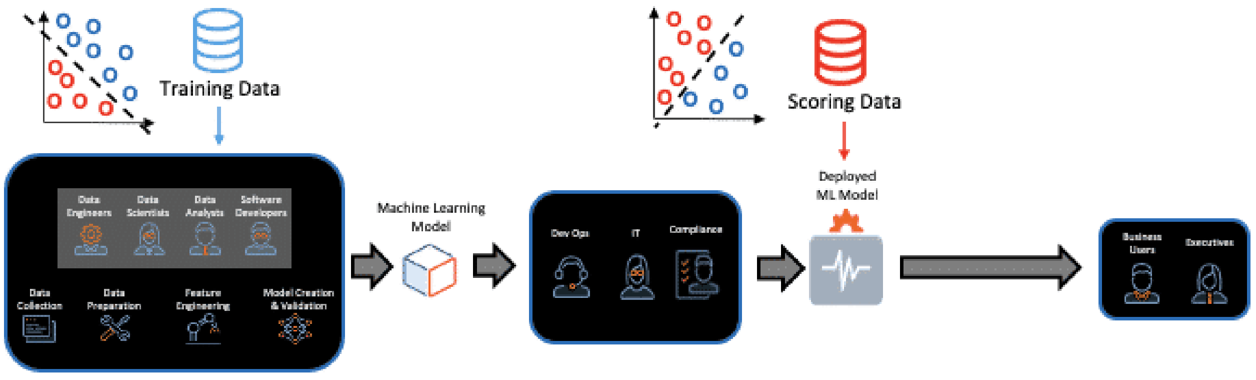
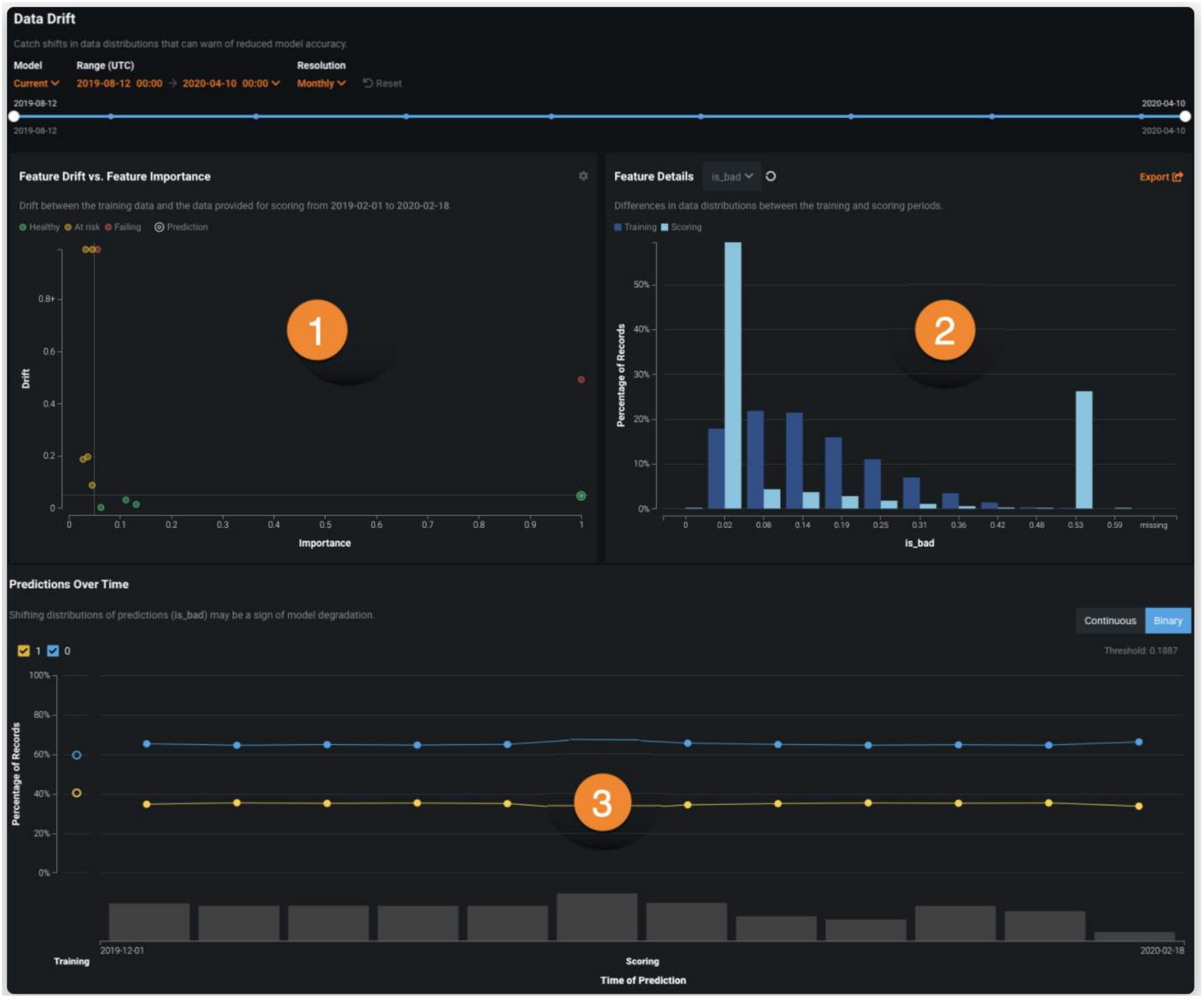


Figure 11: Concept drift: the definition of fraud has changed, as evidenced by the different slope of the line.

2.2.3.3 Monitoring Data Drift

MLOps tracks data drift directly by comparing training data with the scoring data. Drift metrics account for the importance of the features used by the model.

The screenshot shows the visualizations that help you to determine if a model is experiencing data drift and the amount by which production data is drifting.



Data Drift

Catch shifts in data distributions that can warn of reduced model accuracy.

Model: Current
Range (UTC): 2019-08-12 00:00 → 2020-04-10 00:00
Resolution: Monthly

Feature Drift vs. Feature Importance

Drift between the training data and the data provided for scoring from 2019-02-01 to 2020-02-18

Feature Details: is_bad

Differences in data distributions between the training and scoring periods.

Predictions Over Time

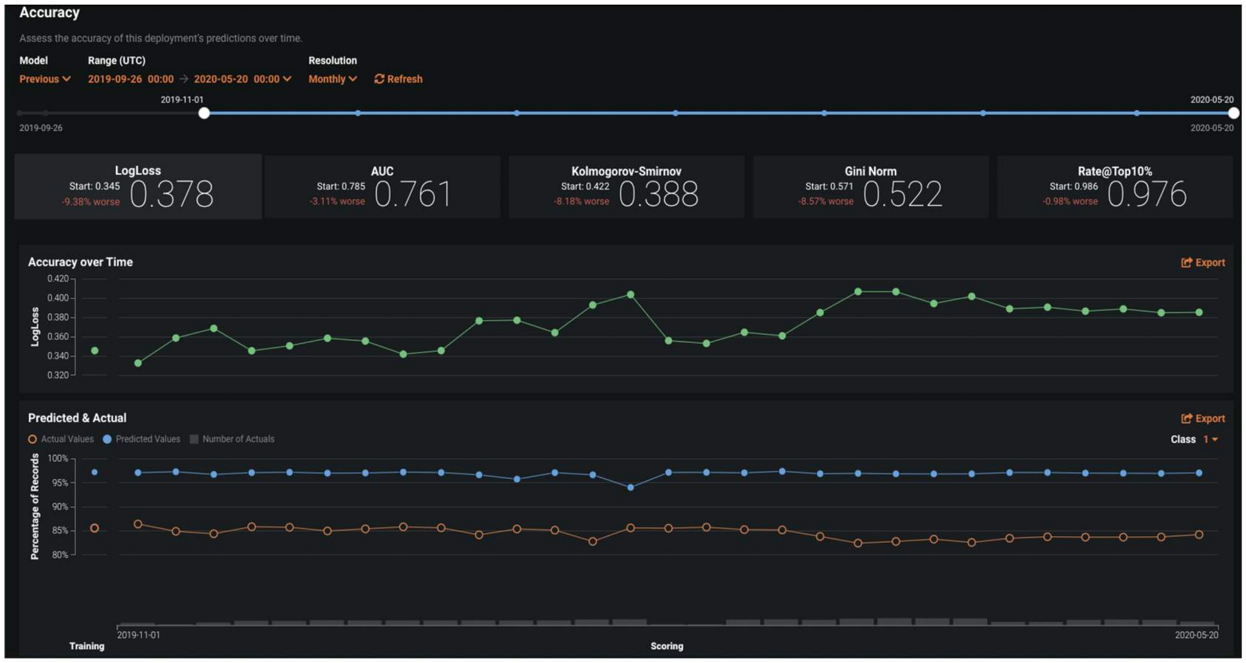
Shifting distributions of predictions (is_bad) may be a sign of model degradation.

Threshold: 0.1887

Figure 12: In the top left, we have the “Feature Drift vs. Feature Importance Plot showing the importance ofa feature in a model versus how much the distribution of actual feature values has drifted. In the top right, the”Feature Details" plot shows the distribution of the selected feature in the training data compared to the scoring data. This will give you an idea of how live data compares to training data. The bottom plot is the "Prediction Over Time Plot and shows how predictions have changed over time (model decay). Regression and binary classification plots display differently.

2.2.3.4 Monitoring Accuracy MLOps helps discover concept drift by monitoring model accuracy, comparing predictions against actual outcomes in the real world. This assumes the existence of an external feedback loop, which depends on the use case. In some cases, the actual outcomes will be available in a reasonably short time, while in other cases it may take months or years.

The resulting accuracy metrics display in the Accuracy tab, which enable you to analyze the performance of model deployments over time using standard statistical measures and visualizations. If a model’s quality is decaying, you can check if the cause is data drift or possibly concept drift.


Accuracy

Assess the accuracy of this deployment's predictions over time.

Model: Previous

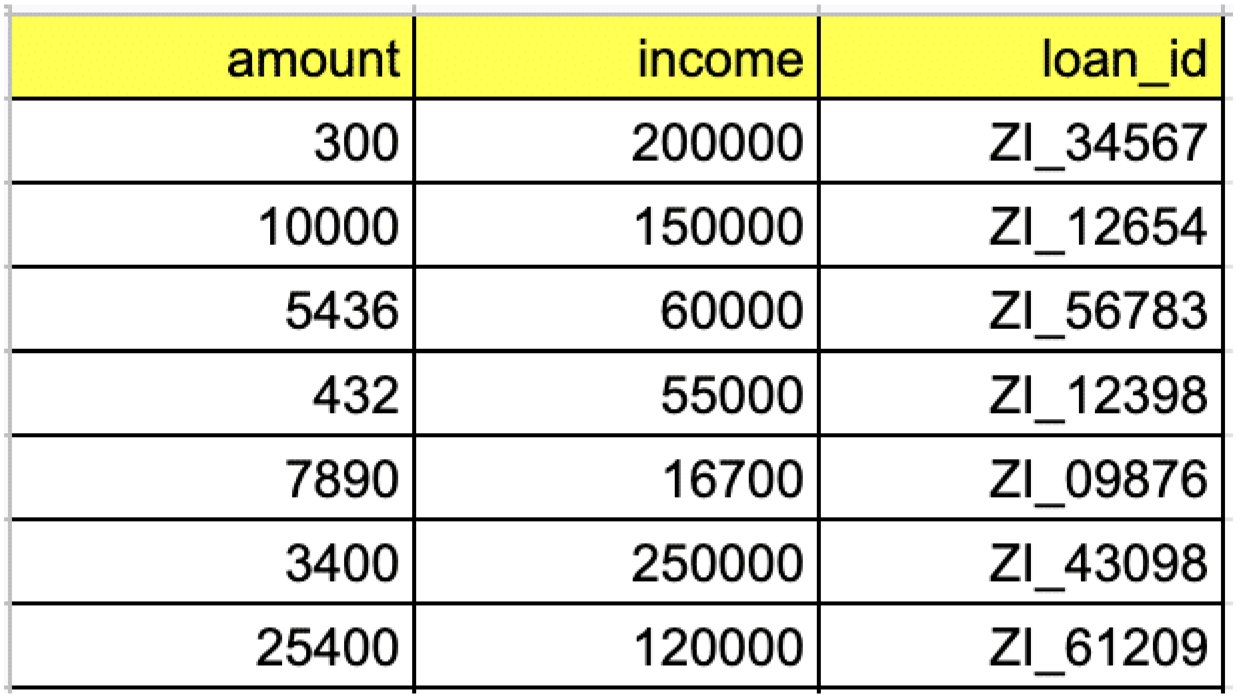
Range (UTC): 2019-09-26 00:00 → 2020-05-20 00:00

Resolution: Monthly

LogLoss	AUC	Kolmogorov-Smirnov	Gini Norm	Rate@Top10%
Start: 0.345	Start: 0.785	Start: 0.422	Start: 0.571	Start: 0.986
-9.38% worse	-3.11% worse	-8.18% worse	-8.57% worse	-0.98% worse
0.378	0.761	0.388	0.522	0.976

Figure 13: Track accuracy over time and predicted vs actual outcomes.

The Accuracy tab renders insights based on the problem type (regression or classification) and its associated optimization metrics. These metrics vary depending on regression or binary classification projects.

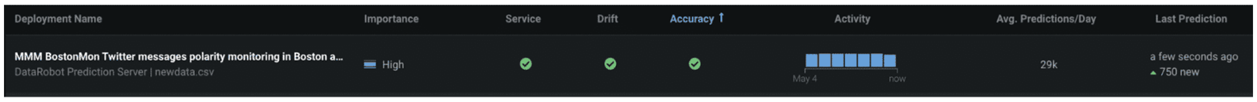


amount	income	loan_id
300	200000	ZI_34567
10000	150000	ZI_12654
5436	60000	ZI_56783
432	55000	ZI_12398
7890	16700	ZI_09876
3400	250000	ZI_43098
25400	120000	ZI_61209

Figure 14: For model accuracy to work, you need to include an association ID variable in your data. The association ID serves as a foreign key for your model predictions so you can later match the actual outcome with a prediction. For example, in order to track loan outcomes to see if any of them later defaulted, the loan_id column could be used here as an association ID.

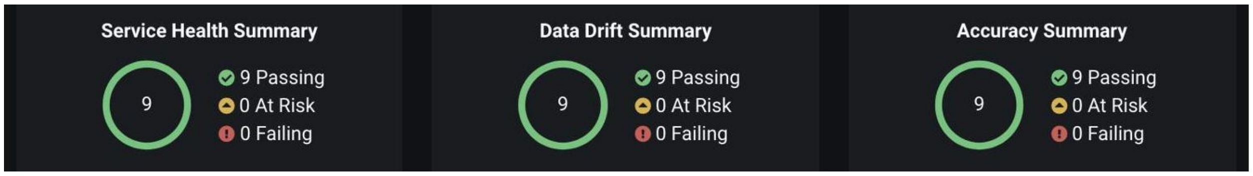
2.2.3.5 Alerts and Notifications MLOps generates alerts if predefined thresholds are exceeded for service health, data drift, and accuracy. You can customize the data drift and accuracy alert thresholds for each deployment.

With MLOps you can view the status of deployment-specific alerts and also color-coded status summaries of all active deployments.



Deployment Name	Importance	Service	Drift	Accuracy ↑	Activity	Avg. Predictions/Day	Last Prediction
MMM BostonMon Twitter messages polarity monitoring in Boston a... DataRobot Prediction Server | newdata.csv	High	✓	✓	✓	May 4 now	29k	a few seconds ago ▲ 750 new

Figure 15: Look at the alert status of specific deployments and set alert thresholds for data drift and model accuracy.

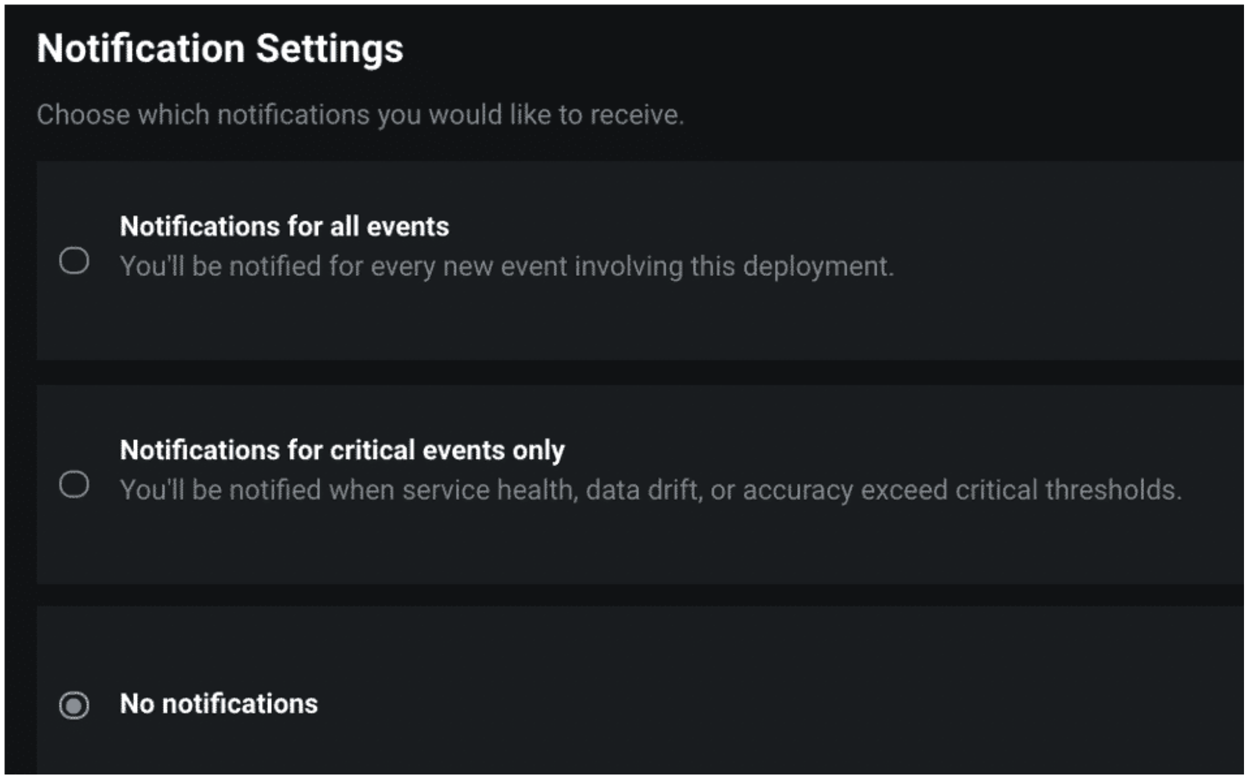


Service Health Summary
9
✓ 9 Passing
0 At Risk
0 Failing

Data Drift Summary
9
✓ 9 Passing
0 At Risk
0 Failing

Accuracy Summary
9
✓ 9 Passing
0 At Risk
0 Failing

You can also schedule to receive email notifications for service health, data drift, and accuracy whenever thresholds are breached. This helps ensure timely corrective action.



Notification Settings

Choose which notifications you would like to receive.

○ **Notifications for all events**
You'll be notified for every new event involving this deployment.

○ **Notifications for critical events only**
You'll be notified when service health, data drift, or accuracy exceed critical thresholds.

[✓ **No notifications**]

Figure 16: The frequency and types of notifications can be configured.

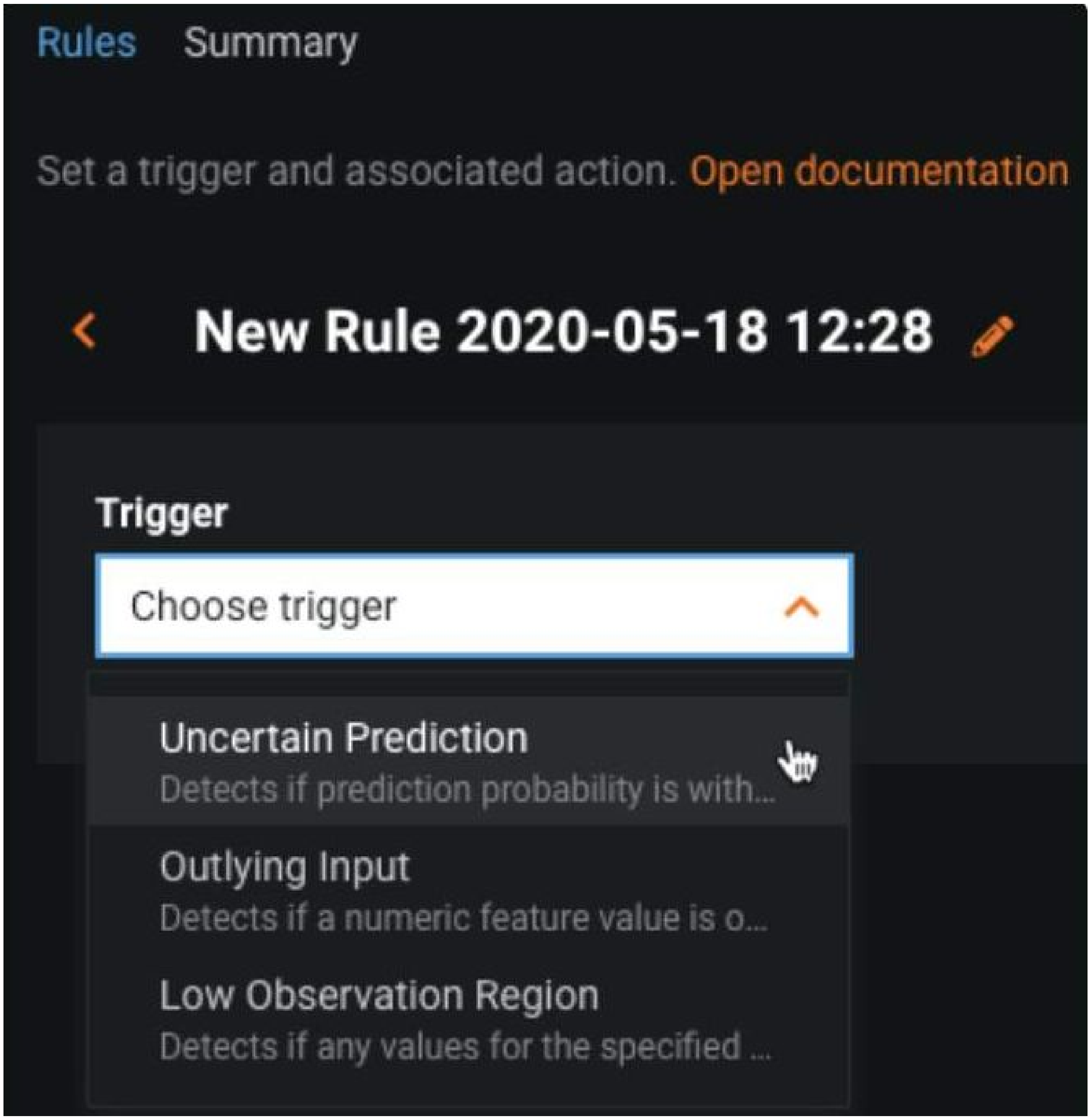
2.2.3.6 Humble AI There is inherent uncertainty in machine learning due to the statistical nature of most algorithms. Moreover, production data can be very different from the data used to train the model. There may be situations in which the model generates uncertain predictions.

You can manage this risk in production on the Humility tab, where you can include a set of rules that allows models to recognize when they make uncertain predictions in real-time or when they receive data they have not seen before.

A trigger can detect a rule violation and apply a pre-selected action.

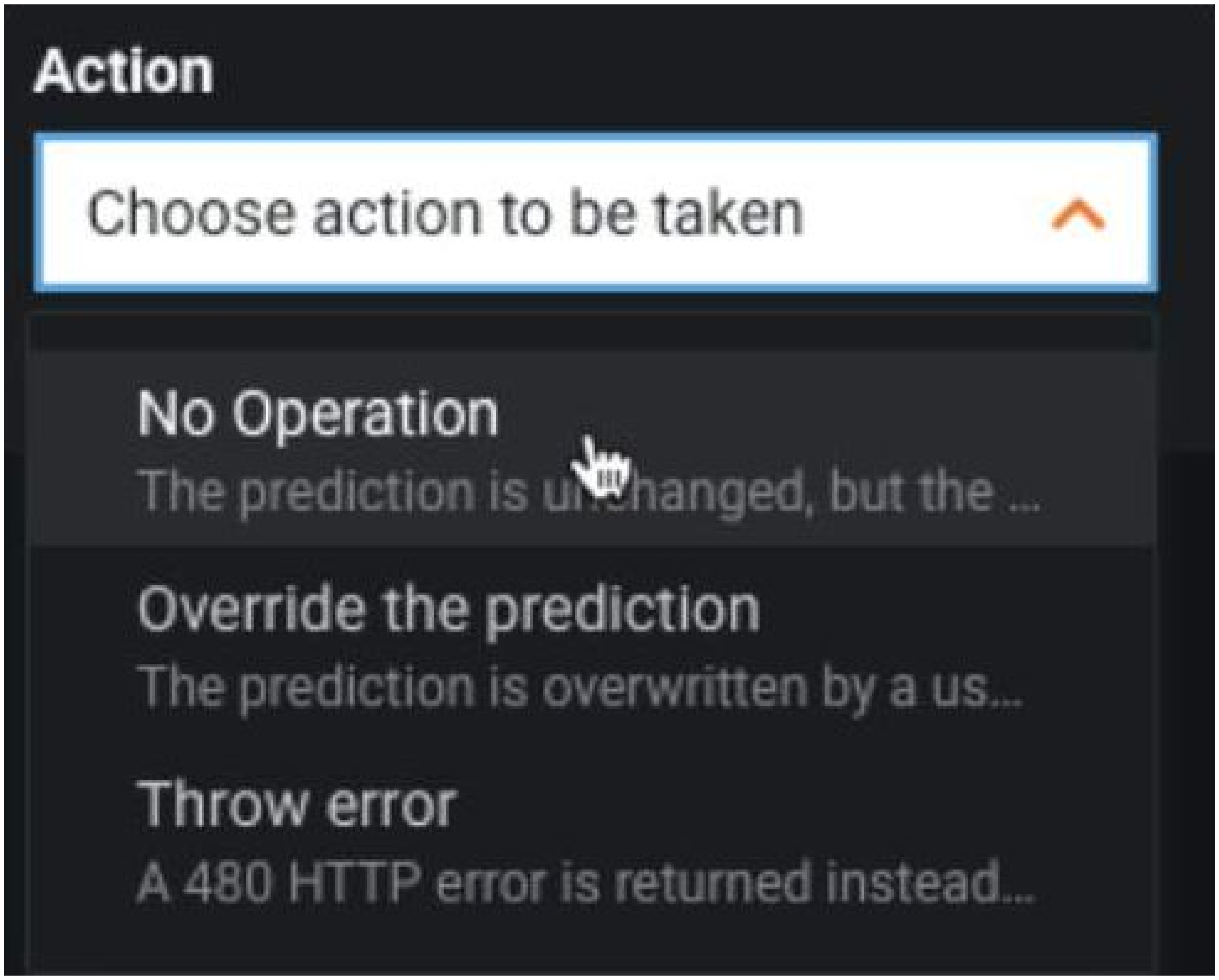
There are three types of triggers:

- Uncertain Prediction — detects whether a prediction's value violates the configured thresholds.
- Outlying Input — detects if the input value of a numeric feature is outside of the configured thresholds.
- Low Observation Region — detects if the input value of a categorical feature value is not included in the list of specified values.


Rules
Summary
Set a trigger and associated action. Open documentation
New Rule 2020-05-18 12:28
Trigger
Choose trigger
Uncertain Prediction
Detects if prediction probability is with...
Outlying Input
Detects if a numeric feature value is o...
Low Observation Region
Detects if any values for the specified ...

For each trigger, you can set an action:

- No Operation — no changes are made to the detected prediction value.
- Override the prediction — modify predicted values for rows violating the trigger with the value configured by the action.
- Throw error — rows in violation of the trigger return a 480 HTTP error with the predictions, which also contributes to the data error rate on the Service Health tab.


Action
Choose action to be taken
No Operation
The prediction is unchanged, but the ...
Override the prediction
The prediction is overwritten by a us...
Throw error
A 480 HTTP error is returned instead...

2.2.3.7 Challenger Models The ML process includes building many candidate models to compete with each other, but there is only one "champion" — this is the model that is selected for a deployment.

The model competition process does not continue into production, but you can still have "challenger" models shadowing the champion and compare the accuracy of the challengers against the champion. This helps identify if you could potentially do better with a different model, as well as prepare for data errors or model decay.

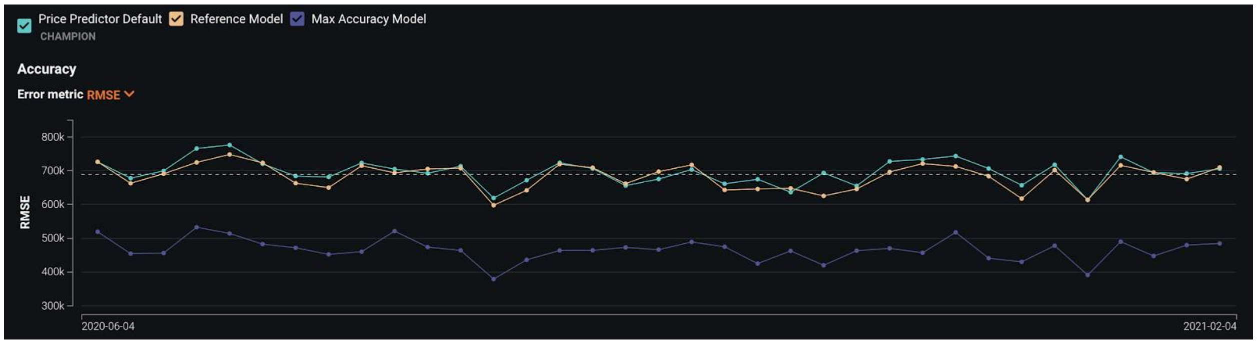


Figure 17: Compare the the scores of challengers against the champion. May the best model win!

2.3 Governance

A deployment's life cycle involves continual learning. After a model is deployed, DataRobot enables you to monitor the model for drift, i.e. when a model . If it has drifted, you can replace the model with a new, retrained model.

When replacing a drifted model, you typically gather new data, build a new model, and replace the stale model without any service disruption. At the same time, you want to track your model inventory. MLOps provides a central hub for this deployment and model management.

The Deployments Inventory shows real-time status of all deployed models.

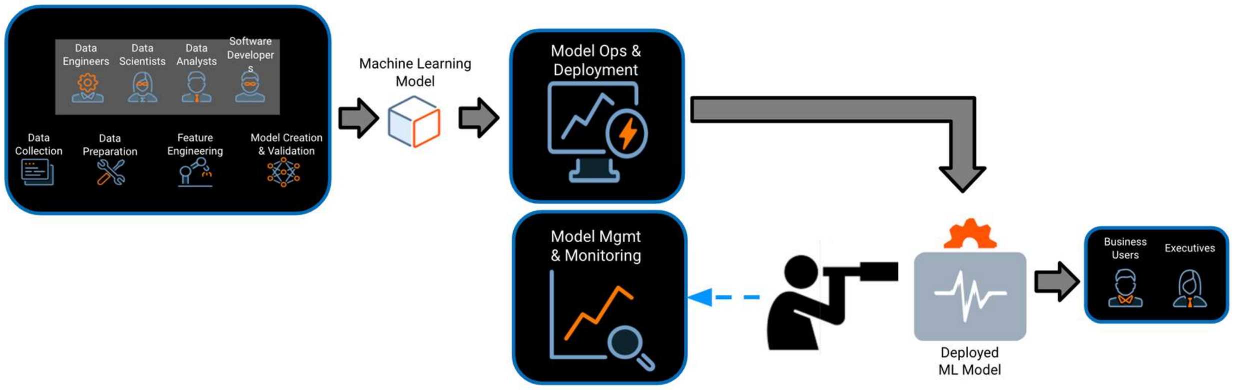


Figure 18: This infographic adds the Model Management and Monitoring step to the previously shown MLOps flowchart to aid intuition.

2.3.1 Model Replacement

Drift and accuracy monitoring capabilities help you determine whether a model's quality is decaying and if you should consider replacing it.

You can set notifications to send you early warnings when a problem is detected.

Then, you can use MLOps best practices to address model decay and model governance.

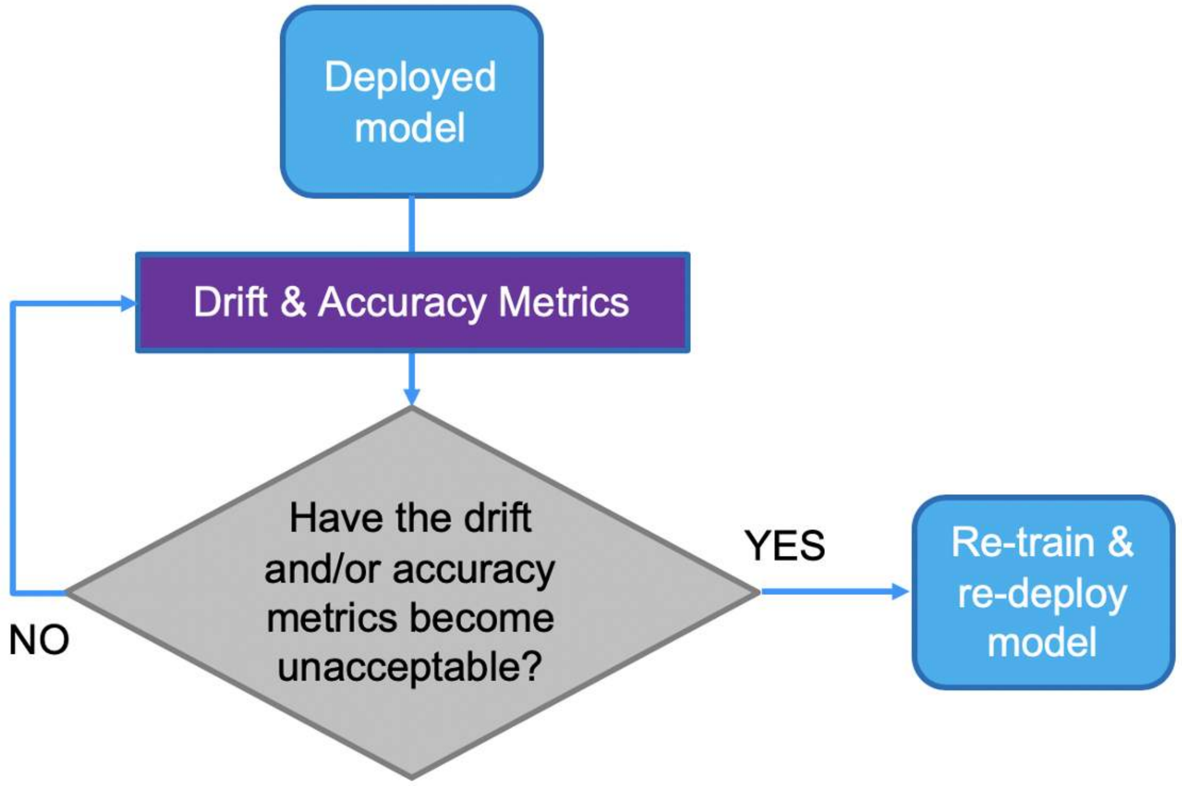
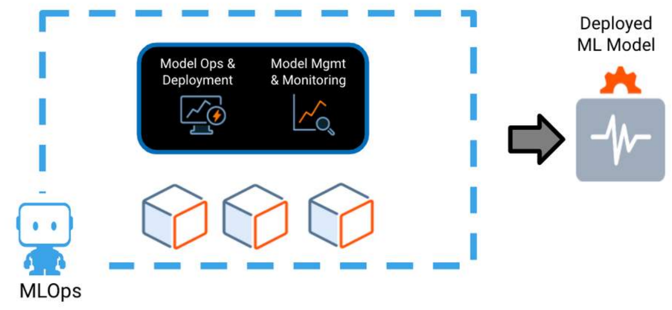

Deployed model
Drift & Accuracy Metrics
Have the drift and/or accuracy metrics become unacceptable?
YES
NO
Re-train & re-deploy model

Figure 19: Decision flowchart for when to replace a model. The steps in this flowchart are easy to do with MLOps, given our dashboards and automatic tracking of drift & accuracy metrics.

Remember, each deployment's model is provided as a model package that you can replace with another model package. During this process, MLOps tracks model history and checks for consistency in model inputs and prediction target.



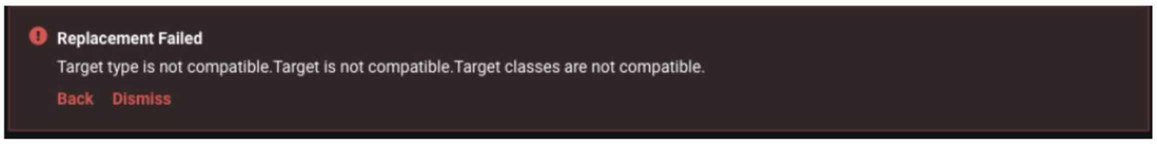
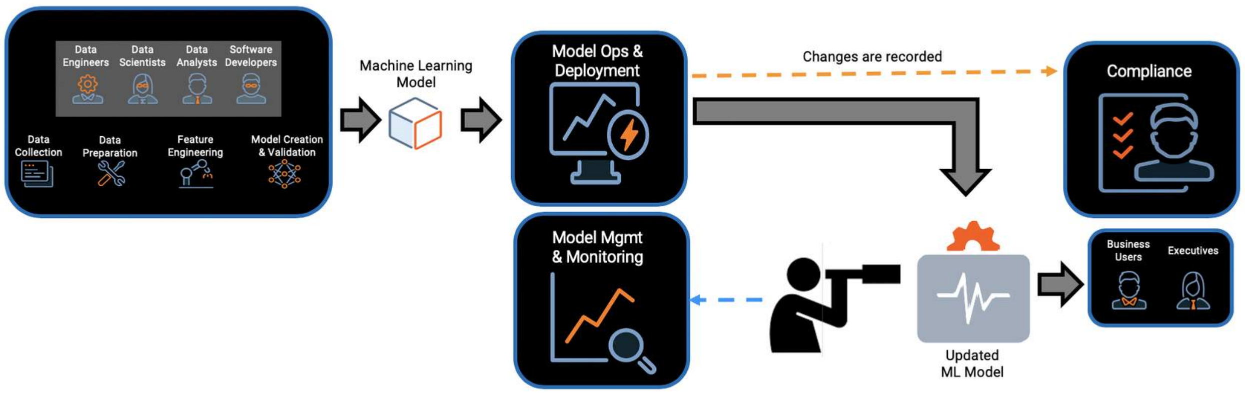

Replacement Failed
Target type is not compatible.Target is not compatible.Target classes are not compatible.

Figure 20: Guardrails! MLOps helps prevent many silly mistakes. In this example, MLOps sees that the target/target type/target classes are not compatible with the new model the user is trying to deploy and gives a warning. MLOps can also check for model input consistency and prevent other mistakes.

2.3.2 MLOps Governance

When a deployed model supports a key business use case, you need to manage potential risk by tracking when the model was deployed, changed, or replaced, and who oversaw the events.



The rationale behind MLOps governance is to enforce an approval workflow for governance events such as creating or deleting a deployment, replacing a model, and indicating the importance of a deployment. You can fully audit and account for every deployment action — which users requested a deployment's creation, change, or deletion and which users reviewed and approved it.

2.3.2.1 Access Control In DataRobot you can grant access to a deployment by assigning a specific role to each person. There are three main roles:

ROLE	ACCESS
Consumer/Observer	Read-only

ROLE	ACCESS
Editor/User	Read/Write
Owner	Read/Write/Administer

There is also the Deployment Administrator role.

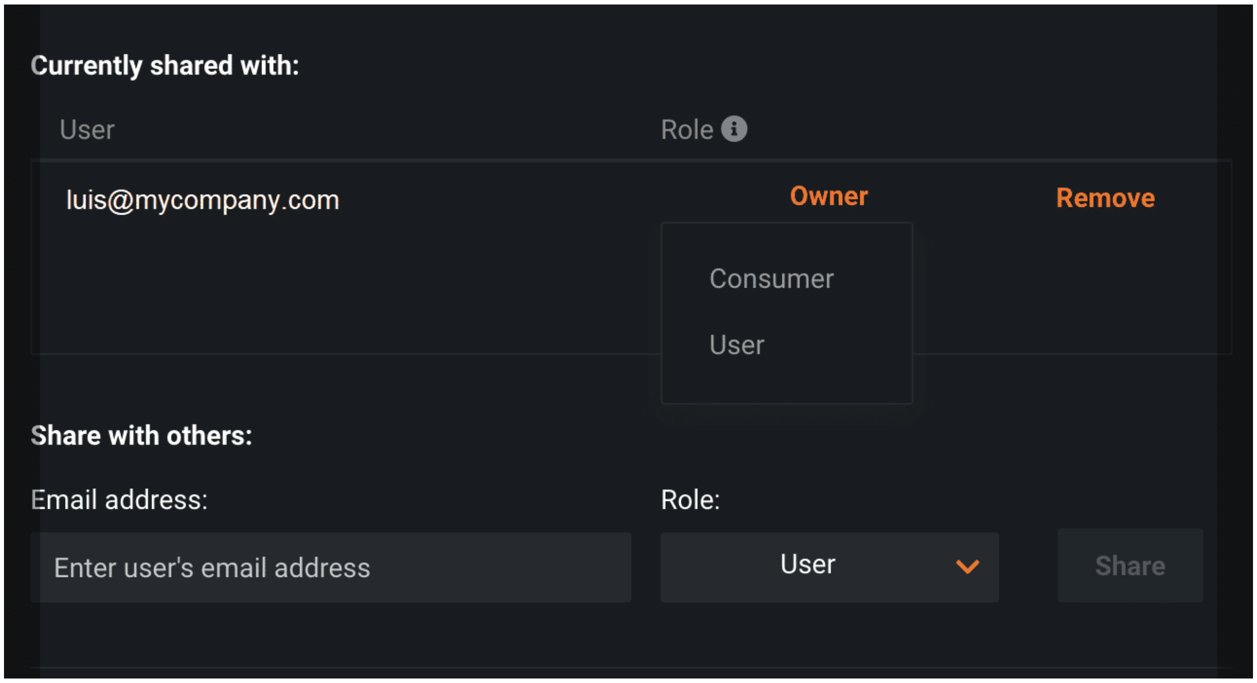


Figure 21: Here we see a screenshot of the 3 different roles that can be granted to each individual involved with a deployment. This makes it easy to manage permissions for security and risk management

Role-based access control (RBAC) controls access to the DataRobot application and is managed by organization administrators. The RBAC roles are named differently but covey the same read/write/admin permissions. The assigned role controls both what you can see when using the application and which objects you have access to.

A user can have multiple roles assigned for a single entity—the most permissive role takes precedence and is then updated according to RBAC. Consider:

- a dataset is shared to an organization, with members assigned the consumer role. The same dataset is shared to a user in that organization with the editor role. That particular user will have editor capabilities, other organization members will be consumers.
- a dataset is shared to a group, with members given owner permissions. You want one user in the group to have only consumer access. You must remove that user from the group and reassign them individually to lessen their permissions.

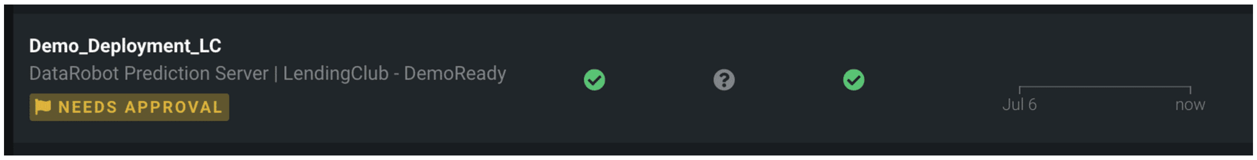
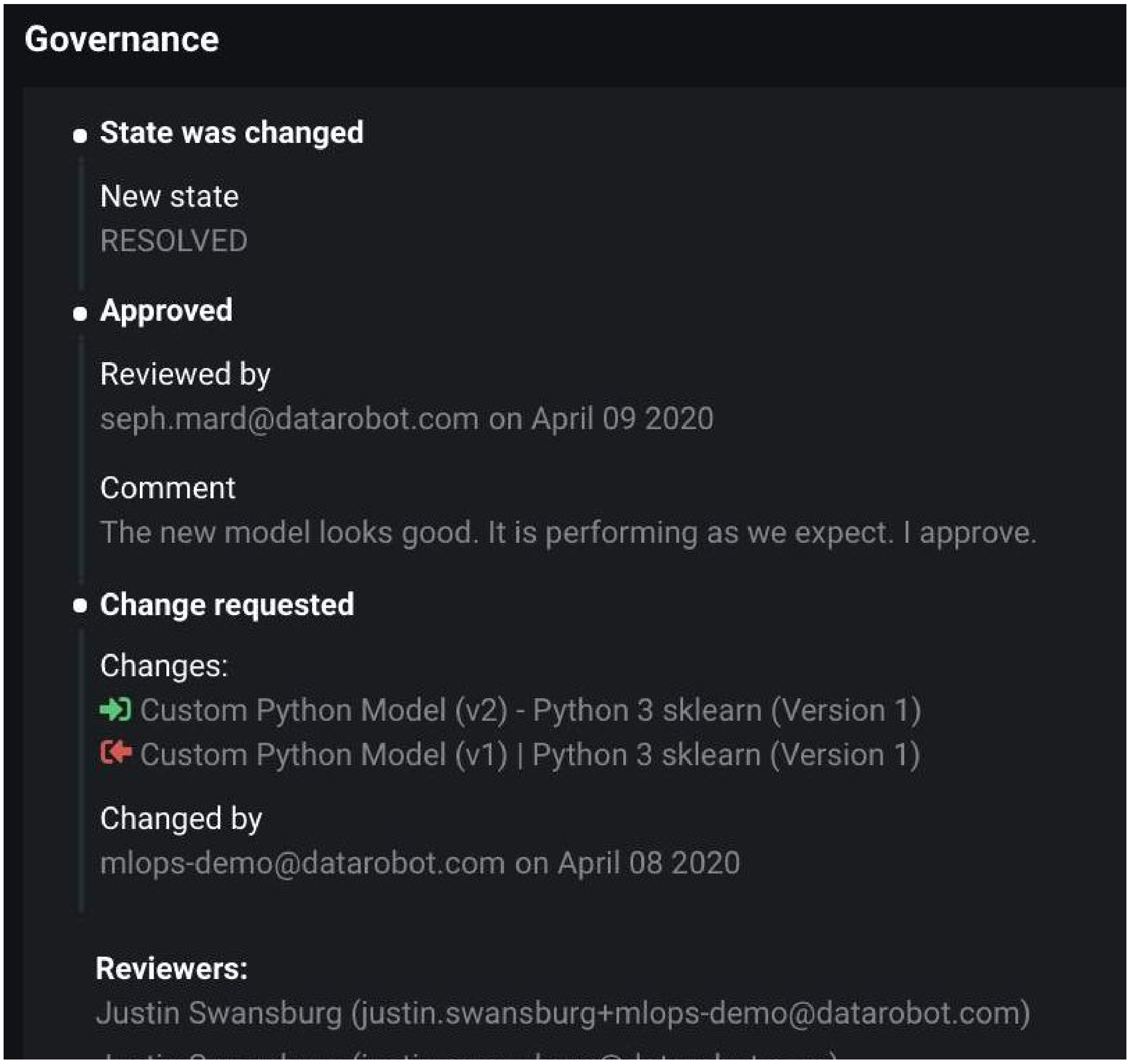


Figure 22: This is an example notification for deployment approval – this helps enforce a healthy process of development, test and review before unleashing new models into the wild


Governance

- State was changed

New state
RESOLVED

- Approved

Reviewed by
seph.mard@datarobot.com on April 09 2020

Comment
The new model looks good. It is performing as we expect. I approve.

- Change requested

Changes:
Custom Python Model (v2) - Python 3 sklearn (Version 1)
Custom Python Model (v1) | Python 3 sklearn (Version 1)

Changed by
mlops-demo@datarobot.com on April 08 2020

Reviewers:
Justin Swansburg (justin.swansburg+mlops-demo@datarobot.com)

Figure 23: This screenshot shows governance audit details in a model deployment's Overview tab. We see a complete audit trail for all applied changes and all reviewers involved. The history shows who created a deployment, who reviewed it, and whether the deployment was approved. You can also see details about a model being replaced, or if the importance of the deployment changed.

2.3.2.2 Deployment Auditing

3 References

- [1] DataRobot Platform Documentation (https://app.datarobot.com/docs/mlops/index.html)
- [2] DRU: MLOps Starter (Self-Paced) (https://university.datarobot.com/mlops-starter/568250/scorm/3vck60rhxfsdh)