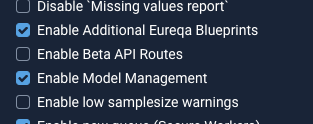
**Model Management Demo Track**

*September 27, 2018*

Setup:

* Make sure you have Model Management on.
  + 
* Make sure you are in an organization, i.e. that when you go to Deploy Model API, you have a server to deploy to.
* Check your Deployments tab to make sure you can see some of the demo deployments (Lending Club, Airlines, Baseball, Predictive Maintenance, etc.)
* Prior to starting the demo, run a version of the same demo dataset through DataRobot and deploy one of those models. We will use this deployment to show how to update with a new model.
* Consider doing the normal talk track through drag and drop, prediction API, Hadoop scoring, and scoring code, then coming back to prediction API for the deeper dive, highlighting the additional features you get with that
* Feature impact and reason codes should already have been calculated for the model you deploy.

Path:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Action** | **Talk** | **Notes** |
| 0 | Introduction | * DataRobot is bringing transparent automation to a new phase of the data science process. * DataRobot Model Management can bring together disparate personas to keep models performing well in production. | If your audience is new to ML, you may have to explain why MM is important. |
| 1 | Click on *Predict -> Deploy Model API* for the model.  Deploy the model. | * Once we’ve decided on what model you want, we want to get it into a controlled environment where we can monitor its performance and ensure no one accidentally changes it. * We’re creating a deployment--an interaction point between the data science team and the rest of the organization. * As we’ll see later, over time, we’ll be able to update the model in this deployment if we think its predictions are becoming unreliable | Ensure you are aware of the options - threshold for predictions, toggle to capture data drift, questions about data drift should be postponed for when you are showing them later |
| 2 | Go to the deployment. | * Here’s our new deployment. We’ve grabbed some metadata about the deployment, and we’ll continue to log any changes. * Since we want to see how to work with deployed models over time, let’s jump to some deployments we’ve had running for a while. |  |
| 3 | Go to the *Integrations* tab of the deployment.  Press the < arrow or click on Deployments. | * To make a prediction, we can just copy and paste this code, which sends data from a CSV against our prediction server’s REST API. Whenever we refresh the model, we’ll be able to use this same code. |  |
| 4 | Focus on different areas of the Deployments tab. | * These are all of our deployed models across all projects and use cases. * Across the top, we see the aggregate activity and the health broken out by three key dimensions:   + *Service health* looks at the core metrics from an ops or engineering perspective: latency, throughput, errors, usage.   + *Data drift* proactively looks at model quality to let us know if there are reasons to think the model is unreliable. We’ll dive into that in a second.   + *Accuracy* is available when we’ve gotten the actual values (or ground truth) corresponding to our predictions, so that we can assess model performance using standard ML metrics. * For each deployment, we’re reporting alarm status * Let’s jump into a deployment to see how it’s doing. | Deployment page takes a few secs to load - use that time to explain the concept |
| 5 | Click on *Service Health* for one of the running models.  Click on the dropdown to show you can look at different periods of time. | * This shows some standard metrics for tracking prediction performance, and catches any errors, either in the incoming data or the prediction server. * By the way, everything we’re going to show in model management is available via the API, so you could plug these statistics into a dashboard if you already use one. |  |
| 6 | Click on the *Data Drift* tab.  Focus on the *Predictions over Time* plot.  Then focus on the *Feature Drift vs. Feature Importance* plot. Hover over some of the points. | * Here’s where we assess if the model is reliable--even before we get the actual values back. Essentially, we’re analyzing how different the data we’ve scored this model on is from the data the model was trained on. * On the left, we’re looking at how the predictions the model made changed over time. If we have some dramatic changes here, that can indicate the model has gone off track. * On the right, we’re looking at the most important features in the model (each dot is a feature; the circled dot represents the predictions). On the y-axis, we look at how different each feature’s distribution is from the training data, the higher up, the more different it is. On the x-axis, we rank features by their feature impact--more important features to the right. This lets us set up the four quadrants: green along the bottom for features that haven’t changed much, yellow for features that have changed but aren’t very important (these are alarming but don’t necessarily mandate action, especially if you have lots of models); and red for features that are important and have drifted. You can adjust the thresholds for both importance and drift based on your needs for a deployment. | We use a tuned version of PSI (Population Stability Index) for measuring data drift. |
| 7 | (Optional) Adjust the time slider. | * We can also change the window of time we want to analyze, to help us identify when features started to drift. |  |
| 8 | Go back to the Deployments of the demo model from setup.  On the hamburger  menu for a model you have Owner permissions to, select Replace Model.  Paste in the replacement URL (from the current project) and hit *Select*.  Click on the dropdown for replacement reason.  Click *Accept and Replace*. | * Let’s assume we’ve identified a model drifting, we want to replace it to keep our predictions accurate. We have a deployment that solved the same problem we just saw in the project we just built. * We can easily update that deployment to use our new model. * We just need a URL for a DataRobot model that has compatible features. This could be from the same project or, more commonly, a model trained on more recent data from the same data source. In our case, this comes from the project we just completed. * After DataRobot validates the model matches, we select a reason why we’re making the replacement. This will be kept in the model log so we know when and why we made a replacement. * The model gets updated which you can see in the History section. New prediction requests will go against the new model. If we want to restore the previous model, we can easily do that through the same process. | The new model needs to have at least the same raw features as the original model in the deployment. This guarantees the prediction requests will succeed. |
| 9 | (Optional) Go to the *Data Drift* tab for the deployment you just made the replacement in.  Click on the dropdown for Model (this will show the previous models deployed). | * DataRobot preserves the full history for all the previous models in this deployment. We can always go back and did into how a model was performing during a certain period of time, and understand why we replaced it. |  |
| 10 | (Optional) Go to the *Integrations* tab and show that the snippet is still the same. | * Whenever we replace the model, the deployment ID stays the same, so we don’t need to update any code that calls this deployment. | DataRobot returns the model ID in the prediction response, so you can always know what model was used for a given prediction. |
| 11 | Go back to the Deployments inventory.  Click on a Accuracy status.  Go back to the Deployments inventory. | * When we have actual values corresponding to the predictions, we can assess how the accuracy changes over time. This gives us a precise measure of model decay, but the downside is that it could take months or years to get back the actuals. This is why DataRobot is applying its data science resources to data drift. | It is useful here to prompt the audience here for (a) how quickly they get back actuals and (b) how they store and process them (e.g. a database?) |
|  | (Optional) Click Add Deployment.  Upload airline\_training1k.csv as training and airline\_scoring1k.csv as scoring.  Assign:  Target = Delayed10  Prediction = Prediction  Date = ScheduledDepartureDateTime  Actual Reponse = Delayed10.  Name the model “External Logistic Regression v3” or something.  Create. Quickly go back to the Deployments inventory. | * We can also create deployments for models executed outside of DataRobot. All we need to do is provide the prediction request and response data, and optionally actual values to assess accuracy. This would typically be done through the API. * Once we map the fields up, this will create a deployment that gets the same model quality monitoring as DataRobot models. |  |
|  | Click the hamburger menu on a model and select share. | * We can share the deployment with anyone in our group--without giving them access to the underlying model or project. This lets them monitor status and stay informed about how a model is performing. |  |
|  | Conclusion. | * DataRobot is bringing transparent automation to a new phase of the data science process. * DataRobot Model Management can bring together disparate personas to keep models performing well in production. |  |

After the demo:

* Delete the deployment you created in step 1. This will improve performance in the future.
* Jot down any notes or any questions from the customer and email them to [tristan.spaulding@datarobot.com](mailto:tristan.spaulding@datarobot.com) . This will improve the product and the demos.

**Supporting Material**

4.4 FAQs [here](https://docs.google.com/document/d/1aLA-xMVh6qaNrA37AGDThgpH-iyS-IDYvWGdaAHrLtA/edit?ts=5b994e5b#heading=h.v1f0l04iw628).

Documentation [here](https://app.datarobot.com/docs/users-guide/deploy/mod-mgt/index.html).

Intro slides [here](http://digifi).