



Lab: Deployment in DSXL

June 19, 2018

Author: Elena Lowery elowery@us.ibm.com



Table of contents

Contents

Overview 1

Required software, access, and files 1

Deployment in DSX 1

Part 1: Test Assets that need to be deployed 2

Test Online Scoring 2

Create Batch Scripts and Test Batch Scoring..... 4

Create Evaluation Script and Test Evaluation..... 5

Part 2: Deploy project to production 8

Overview

In this lab you will learn how to deploy analytics assets in IBM Data Science Experience (DSX)

Required software, access, and files

- To complete this lab, you will need access to a DSX Local cluster.
- You will also need to download and unzip this GitHub repository:
https://github.com/elenalowery/DSX_Local_Workshop_12

Deployment in DSX

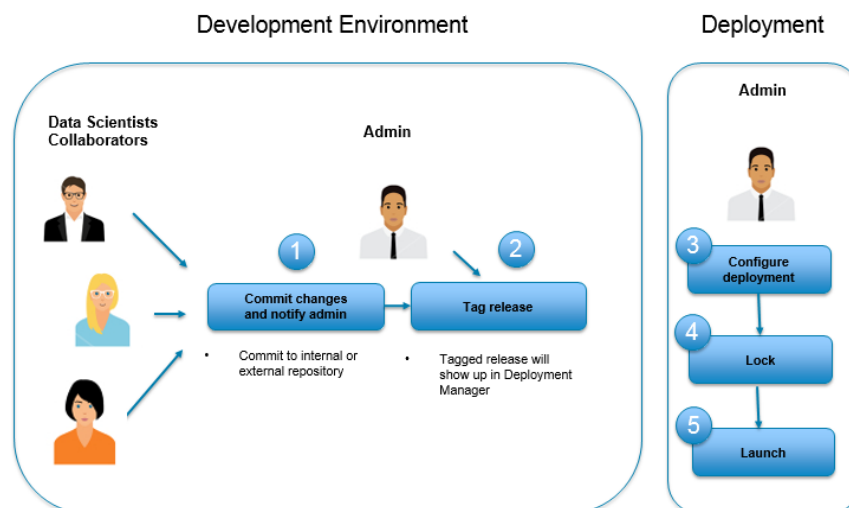
Deployment in DSX is accessed through **Deployment Manager**, which is available in the DSX UI for users with *admin* privilege.

Deployment provides the following capabilities:

- Deployment of models and scripts for online scoring
- Deployment of notebooks and scripts for batch scoring
- Deployment of Shiny applications
- Scheduling of model evaluation.

Some of the deployment tasks can be done in the development environment, but from the licensing perspective, these tasks should only be used for testing.

Deployment Manager provides important separation of the development and deployment tasks. Here's how the deployment workflow is implemented in DSX.



1. Data scientists collaborate on a project, and when they're done with development and testing, they notify the admin that the project is ready to be deployed in production.
2. The admin tags the project release and uses **Deployment Manager** UI to configure deployments.
3. When the project is launched, the REST APIs for deployed assets become available and all scheduled jobs will run as configured.

Part 1: Test Assets that need to be deployed

Note: testing is done in development environment.

Test Online Scoring

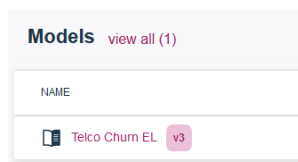
1. If you haven't already created a project from *DSX_Local_Workshop12.zip* file, follow instructions for *Use Case 1* in this repository https://github.com/elenalowery/DSX_Local_Workshop_12
2. If you haven't run through the *TelcoChurn* notebook, run through it so that you generate a model. The easiest way to do this is to open the notebook, scroll down to **Step 10**, click on it, then in the menu select **Cell -> Run all above**.

Save the model in the repository. If you wish, you can change the model name.

```
[ ]: model_name="Telco Churn EL"
save(model=model, name=model_name, test_data=test, al
```

Step 10: Deploy and Test model with UI

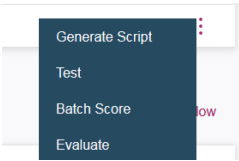
Navigate to the **Assets** view and make sure that the model has been created. Your model may have a different name and version.



3. Click on the vertical ellipses next to the model.

Notice that you have **Generate Script** option for the model. If you generate a script for online model deployment, you can add additional operations prior to invoking scoring. Generating a script is an **optional** task for online

scoring. If you don't generate a script, a default script will be used during deployment.

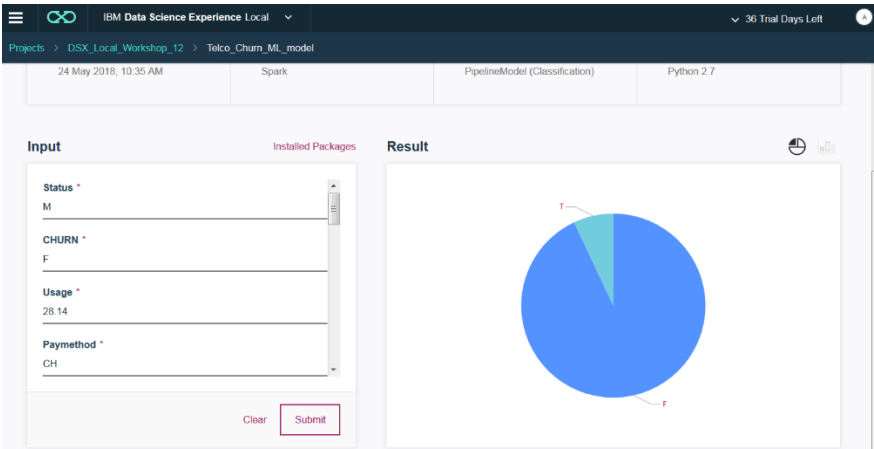


If you would like to take a look at the default script, click the **Generate Script** button. Take a note of the script name – later you can use it for deployment.

```
Projects > DSX_Local_Workshop_12 > Telco_Churn_ML_model_scoring_1527261222650.py
Telco_Churn_ML_model_scoring_1527261222650.py
1 # Copyright 2017, 2018 IBM. IPLA licensed Sample Materials.
2
```

- Click on the vertical ellipses next to the model and select **Test**.

The data for testing is prefilled because we save the training data with the model. You can test with the default data or change some values.



It's also possible to test with the "internal REST API", as shown in the *TelcoChurn_SparkML* notebook. The external REST API will be available when we deploy the model in the **Deployment Manager** (later in this lab).

Step 11: Test model with a REST API call (Optional)

This step demonstrates an "internal REST API" call to test the model (for an unpublished model syntax. An external REST call will have a different end point and will require authentication.

```
json_payload = [{
  "ID": 999,
  "Gender": "F"
}]
```

Create Batch Scripts and Test Batch Scoring

In order to create a batch scoring job in **Deployment Manager**, we first need to create a script in the development environment. A data scientist can also test a batch job.

1. Click on the ellipses next to the model and select **Batch Score**.
2. Fill out the required fields:
 - **Input data set** (Local): *new_customer_churn_data.csv*
 - **Output data set**: *ScoringResults.csv*. Make sure to provide .csv extension – otherwise you won't be able to preview and download the output.

Batch scoring script inputs

Execution Type *
DSX

Input data set *
new_customer_churn_data.csv

Output data set *
☒ Local file ☐ Remote data set
 ScoringResults.csv

32

3. Click on **Advanced Settings** and change the file name to *ScoreTelcoCustomers*. Click **Save**.

Advanced settings

File name *
ScoreTelcoCustomers

File extension *
.py

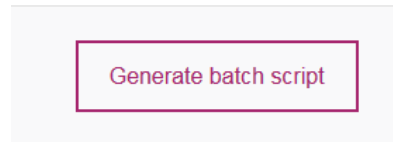
Job name *
TelcoChurnMLmodel-1527263985422

Job description
Job description

Cancel Save

31
19

- Click **Generate Batch Script**.



Note: the batch script can be edited. Examples of changes to the batch script are

- *Invoking another script (for example, ETL) prior to model scoring*
- *Changing input and output data sources.*

- Click **Run now** and wait till the status changes to *Success*. If you would like to look at the log files, you can click on the job id.

Runs run now						
ID	NAME	TARGET HOST	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT
1527264422-999	Run 1	Local instance	admin	25 May 2018, 11:07 AM	89	Success

You have finished testing batch scoring of a model in the development environment.

Create Evaluation Script and Test Evaluation

In order to create an evaluation job in **Deployment Manager**, we first need to create a script in the development environment. A data scientist can also test evaluation.

Note: In our example the "Evaluation data set" is subset of data used for modeling. We chose this approach for convenience and demonstration. In a production environment the "Evaluation data set" is the new set of historical data that's used to verify that the model is still accurate. This data set can be automatically uploaded to the data source that's used for evaluation either by a script in DSX or an external process.

- Click on the ellipses next to the model and select **Evaluate**.
- Select the data source for evaluation (*TelcoModelEval.csv* file which we generated in a notebook).

When we ran evaluation in the notebook we used *BinaryClassificationEvaluator* and *Area Under Roc Curve* as the metric. We suggest that you use the same values when creating the evaluation script.

Keep the default *Threshold* values.

Schedule evaluation script inputs

Input data set *

TelcoModelEval.csv

Evaluator *

Binary

Threshold metric *

Area under ROC Curve

Threshold *

0

1

Min: 0.30

Mid: 0.70

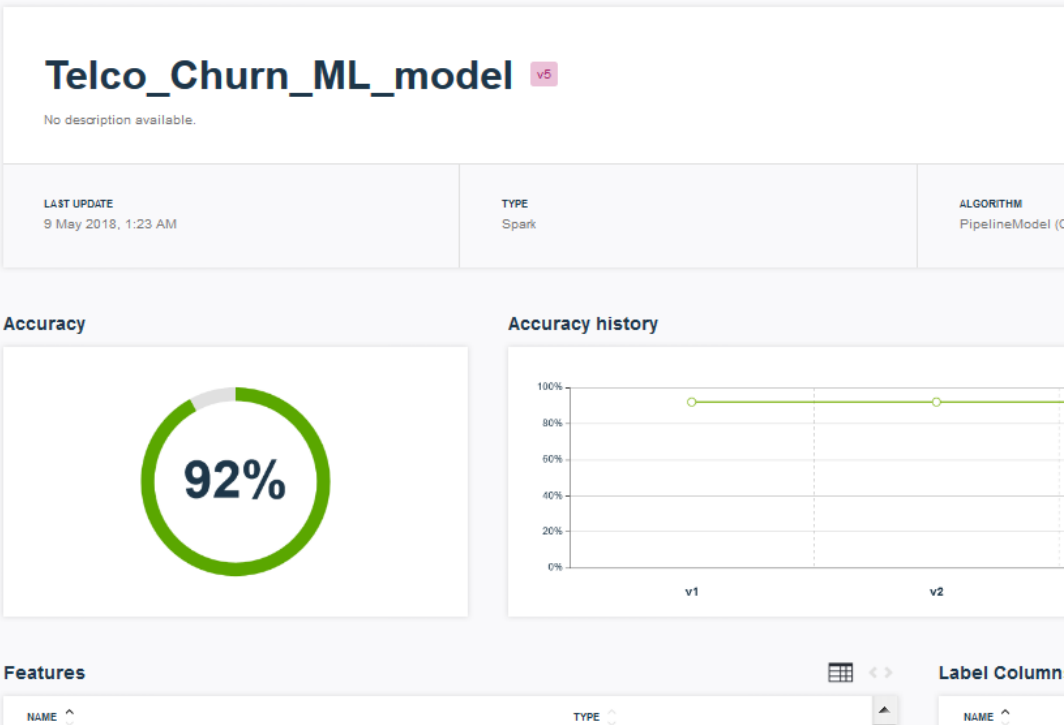
- Click on **Advanced Settings** and change the name of the script. For example, you can name it *TelcoChurnEvalScript*. Click **Save**.
- Click **Generate Evaluation Script**.
- Click **Run now**.
- Scroll down to review the results and wait till the run has finished.

Runs

run now

ID	NAME	TARGET HOST	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT
1529414331-999	Run 1	Local instance	admin	19 Jun 2018, 8:19 AM	89	<div>Success</div>

- Navigate to **Project** view and click on the model - you will see model evaluation results.

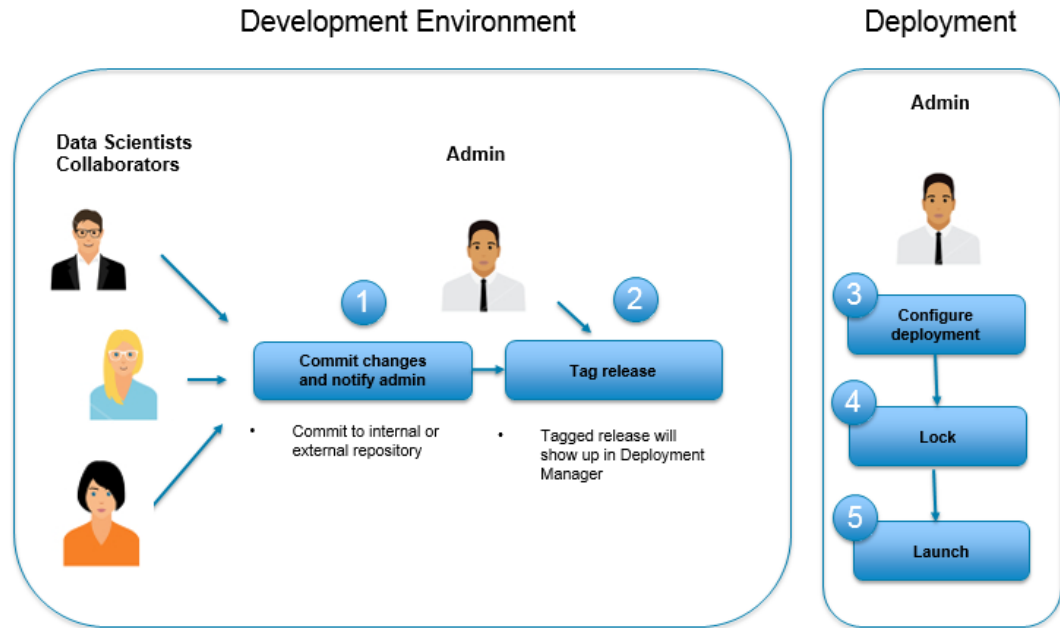


You have finished testing evaluation in the development environment.

Part 2: Deploy project to production

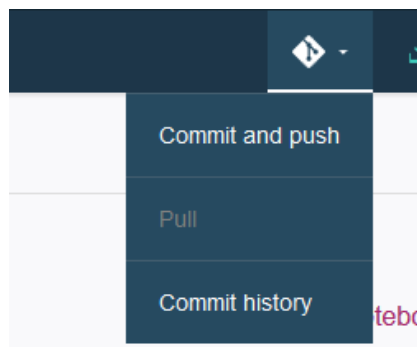
Deployment in DSX can be done by any user who has admin role for both DSX and the project. Many companies have a designated person who's responsible for deployment of analytics.

As a reminder, we will be following the following steps to deploy analytic assets to production.



1. Data scientists need to commit changes to the project. You already may have seen the “*Push and commit*” message close to the menu bar. You can invoke Commit by clicking on the **Git actions** icon in the top right corner. Select **Commit and push**.

Note: You have to be in the Project details view to see the icon.



2. Provide the *Commit message* about the changes. At this time don't specify the tag. Click **Commit and push**.

Commit and push

Remote repository location
DSX master repository

Files to commit and push (12)

<input checked="" type="checkbox"/>		models/Telco_Churn_ML_model/5/jobs.json
<input checked="" type="checkbox"/>		models/Telco_Churn_ML_model/.ScoreTelcoCustomers.py.link
<input checked="" type="checkbox"/>		models/Telco_Churn_ML_model/.TelcoChurnEvalScript.py.link
<input checked="" type="checkbox"/>		models/Telco_Churn_ML_model/5/evaluations.json

☒ All 3 modified 9 added 0 removed

Commit message*

Generated scripts for deployment

Create version tag for release ⓘ
Type tag name here

Cancel
Commit and push

3. Now we are assuming that a different person, an *admin*, is taking over deployment. Click on the **Git actions** icon and select **Commit History**. Notice that we can add a *tag* to the project.

A *tag* is used to identify a specific version of the project. There may be many versions of the assets in the project, but only specific versions should be used in production.

Branch: **Master**

June 19, 2018

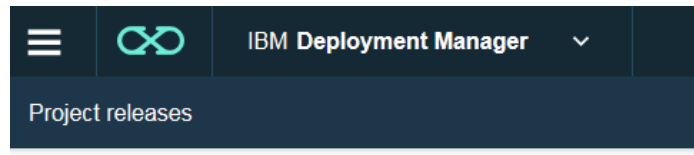
Commit: 75853e9 by **admin** at 8:35 AM

Tag Generated scripts for deployment

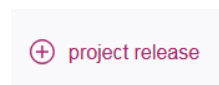
Enter tag name Cancel Save

4. Provide a tag, for example, *WorkshopRelease*, and click **Save**.

5. Navigate to the **Deployment Manager** view by selecting it from the main menu.



6. Click the **+** icon to create a project release.



7. Enter the required fields

- **Name:** *WorkshopRelease1*
- **Route:** *release1* (this string will be used in all URLs that are generated by deployment, that's why it can't have spaces, upper case characters, and special characters). *Important: if you're sharing a cluster, make the route unique, for example, add initials to release1.*
- **Source project:** the project that you want to deploy
- **Tag:** tag that you specified in the earlier steps.

Create project release

From DSX From repository From file

Name *

WorkshopRelease1

84

Route * ⓘ

release1

18

Source project *

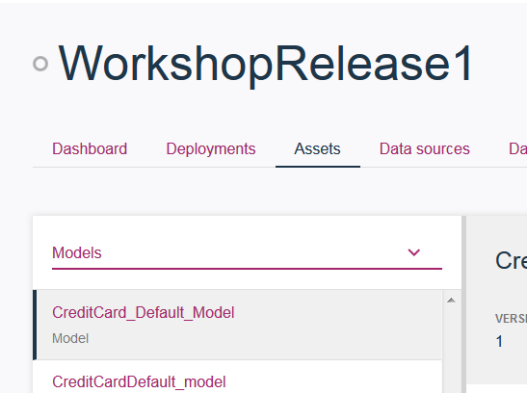
DSX_Local_Workshop_12

Tag *

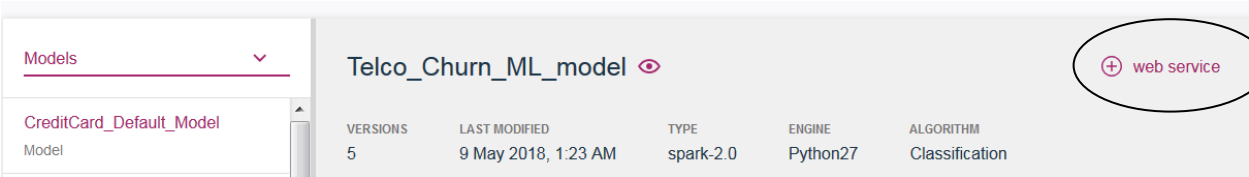
WorkshopRelease

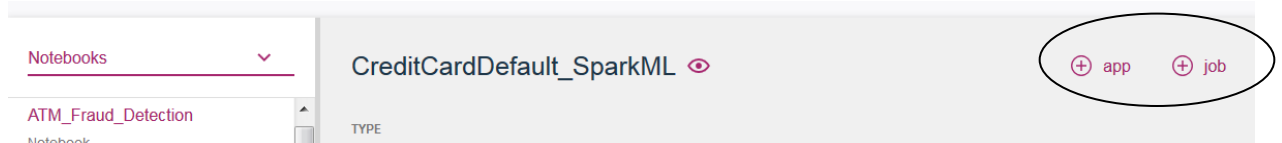
Click **Create**. This will take a few minutes because DSX is making a copy of all assets in the project.

8. The default view shows all assets that are a part of the project. Notice that you can filter them by type if you select the drop down.



9. Select different asset types (models, notebooks, scripts, etc.), and notice how deployment options (icons in the right corner) change.





DSX automatically determines applicable deployment type for each asset.

- **Models** can be deployed as Web services for **online scoring** (invoked with a REST API). To deploy a model for batch scoring, the “script” deployment option is used. The script is generated on the development side, as we have done in *Part 1* of the lab.
- **Scripts** can be deployed for **online** or **batch scoring**. DSX automatically determines if the script can be used for online scoring (a script generated by DSX or a script that follows the specific format for online scoring). DSX supports deployment of any R or Python script. Documentation explains the required script format for online scoring: <https://content-dsxlocal.mybluemix.net/docs/content/local-dev/dsxl-scripts-as-web-services.html>
- **Model evaluation** is a type of **batch script deployment**.
- **Notebooks** can be deployed for **batch scoring** or as **an application**. Batch scoring of notebooks can be used to perform many functions: scoring, model refresh, data preparation, etc. Publishing a notebook as “an application” makes it available as an HTML page that can be accessed with specified level of security.
- **Shiny applications** can be deployed as **an application**. Publishing Shiny as “an application” makes it available as an HTML page that can be accessed with specified level of security.
- **SPSS flows** can be deployed for batch scoring.

10. Before we configure deployment, let’s review properties of the *release*.

- On the **Dashboard** tab you will see results of evaluations. The evaluations shown here are both evaluations done in the development environment and configured in deployment. We will configure an evaluation later in this lab.

WorkshopRelease1

Dashboard Deployments Assets Data sources Data sets Active environments

Current model metrics

Recent model evaluations

MODEL NAME	EVALUATION TIME	PERFORMANCE
Telco_Churn_ML_model	19 Jun 2018, 8:19 AM	✓ Good
Telco_Churn_ML_model	3 May 2018, 1:11 PM	✓ Good
Telco_Churn_ML_model	24 Apr 2018, 5:06 PM	✓ Good

- The **Deployments** tab is empty at this time because we haven't configured any deployments yet.

Dashboard Deployments Assets Data sources Data sets Active environments

NAME ASSET TYPE VISIBILITY DATE STARTED

*No deployments found. Go to **assets** to configure and deploy.*

- The **Assets** and **Data Sources** tabs show the same assets and data sources as the development side.
- Active environments** is also empty because we haven't configured and launched the project yet.

11. We will start with configuring online deployment. On the **Assets** tab, select **Models** in the dropdown, then click on *Telco_Churn_ML_model*.

WorkshopRelease1

Dashboard Deployments Assets Data sources Data sets Active environments

Search by asset name

Models

- CreditCard_Default_Model Model
- CreditCardDefault_model Model
- SPSS_ResponseChannel Model
- Telco Churn Zeppelin Model
- Telco_Churn_ML_model Model**

Telco_Churn_ML_model

web service

VERSIONS	LAST MODIFIED	TYPE	ENGINE	ALGORITHM
5	9 May 2018, 1:23 AM	spark-2.0	Python27	Classification

NAME ASSET TYPE VISIBILITY DATE STARTED AVAILABILITY

No deployments found.

12. Click the **web service** button. Fill out the required fields.

- **Name:** *telcochurn1*. Notice that the name gets appended to the URL (REST endpoint), that's why it has to be lowercase with no special characters.
- **Model version:** you can select different versions of the model
- **Web service environment:** should be the same as the environment in which model was built (selected by default)
- **Reserve resources:** checking this option will provide dedicated CPUs and memory to online deployment
- **Replicas:** number of environments that host the service - select at least 2 for high availability.

Name *
telcochurn1 ✓

URL
https://169.55.181.211/dmodel/v1/release1/pyscript/telcochurn1 15

Model version *
Use latest version ▼

Web service environment *
Python 2.7 - Script as a Service ▼


Reserve resources
☐

Replicas
2 ▲ ▼

Click **Create**.

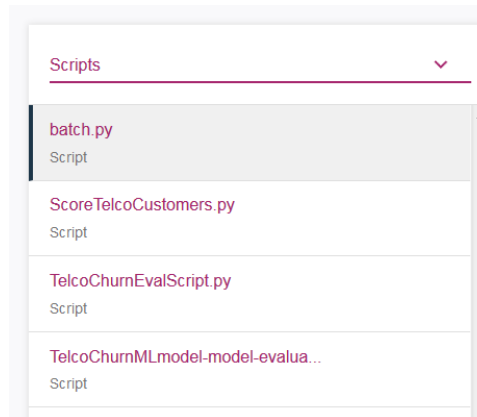
The **REST endpoint** is displayed in the model details. This endpoint won't be live until we launch the project, which we will do after we configure all other deployments.

Overview **API**

POST https://169.55.181.211/dsvc/v1/release1/pyscript/telcochurn1/score 

13. Next, we will configure batch scoring and model evaluation.

On the **Assets** tab select the **Scripts** from the dropdown. The two scripts that we created in the development environment are *ScoreTelcoCustomer.py* (used for batch scoring) and *TelcoChurnEvalScript.py* (used for model evaluation).



14. Select *ScoreTelcoCustomers.py* and click on **+job** button. Fill out the required fields.

- **Name:** *telcobatch1*. The name gets added to the REST endpoint. Batch jobs can be invoked with a REST API. Unlike online scoring, data is not passed in – the data sources that are defined in the script are used.
- **Type:** batch scoring
- **Worker:** *Jupyter with Python 2.7, Scala 2.11, R 3.4.3*. The worker should be the same runtime environment as was used for running the script in development.

Scroll down and review the rest of the fields (default values can be used).

Project releases > WorkshopRelease1 > Deploy batch.py as a job

Deploy batch.py as a job

Name *

telcobatch

16

URL

https://169.55.181.211/djob/v1/release1/telcobatch

Description

Job description

300

Type *

Batch scoring

▼

Worker *

Jupyter with Python 2.7, Scala 2.11, R 3.4.3

▼

Tenant host *

If you specified a schedule for invoking the job, it will take effect when the project is launched. You will also be able to invoke it “on demand” by selecting it in the **Deployments** view. We will complete this step later in the lab.

- Repeat the same steps to create a model evaluation job. The only difference is the type – *Model Evaluation*.

Deploy TelcoChurnEvalScript.py as a job

Name *

telcoeval

17

URL

https://169.55.181.211/djob/v1/release1/telcoeval

Description

Job description

300

Type *

Model evaluation

▼

Worker *

Jupyter with Python 2.7, Scala 2.11, R 3.4.3

▼

We don't have an option to modify data source (in our example .csv files) when we deploy assets for batch scoring and evaluation. If we used a remote data source (for example, a database table or a file in HDFS), then we would be able to modify input/output by modifying the data source definition.

If you click on the **Deployments** tab now, you'll see that every deployment is in *Disabled* status because we haven't launched the project.

WorkshopRelease1

Launch

Dashboard

Deployments

Assets

Data sources

Data sets

Active environments

Search by deployment name

NAME	ASSET	TYPE	VISIBILITY	DATE STARTED	AVAILABILITY
telcoeval	TelcoChurnEvalScript.py	Job	—	19 Jun 2018, 4:06 PM	<div>✖ Disabled</div>
telcochurn1	default_spark.py	Web service	—	19 Jun 2018, 11:15 AM	<div>✖ Disabled</div>
telcobatch	batch.py	Job	—	19 Jun 2018, 12:41 PM	<div>✖ Disabled</div>

16. In this step we will deploy a notebook as an application.

In the **Assets** tab select **Notebooks**. Click on any notebook and click **+app**. Provide name and select the desired security setting.

Deploy TelcoChurn_SparkML.jupyter.ipynb as an app

Name *

notebook1

17

URL

https://169.55.181.211/dapp/v1/release1/jupyter/notebook1

Shared with *

Anyone with the link

Any authenticated user

Deployment admin

The URL for the notebook will be "live" (accessible) after we launch the release.

17. Now that we created a few deployments, we can launch the release by clicking the **Launch** button in the top right corner.

Launching the release will:

- Start all environments that will be used for deployment
- Enable the REST endpoints
- Enable URLs for “applications” (notebooks and Shiny)
- Enable schedules (if they are configured)
- Enable on-demand invocation of jobs.

Notice that when you click Launch, everything with the exception of online (Web service) deployment is immediately enabled.

✓ WorkshopRelease1 was successfully brought online.

WorkshopRelease1

Dashboard Deployments Assets Data sources Data sets Active environments Search by deployment name

NAME	ASSET	TYPE	VISIBILITY	DATE STARTED	AVAILABILITY	
telcoeval	TelcoChurnEvalScript.py	Job	—	19 Jun 2018, 4:06 PM	✓ Enabled	⋮
telcochurn1	default_spark.py	Web service	—	19 Jun 2018, 11:15 AM	✗ Disabled	⋮
telcobatch	batch.py	Job	—	19 Jun 2018, 12:41 PM	✓ Enabled	⋮
notebook1	TelcoChurn_SparkML_jupyter.ipynb	App	Anyone with the link	19 Jun 2018, 5:14 PM	✓ Enabled	⋮

Click **Enable** from the menu (the ellipses in the row for the Web services deployment). Online deployment environment takes some time to start. When it's started, the status will change to *Enabled*.

WorkshopRelease1

Dashboard Deployments Assets Data sources Data sets Active environments Search by deployment name

NAME	ASSET	TYPE	VISIBILITY	DATE STARTED	AVAILABILITY	
telcoeval	TelcoChurnEvalScript.py	Job	—	19 Jun 2018, 4:06 PM	✓ Enabled	⋮
telcochurn1	default_spark.py	Web service	—	19 Jun 2018, 11:15 AM	✓ Enabled	⋮
telcobatch	batch.py	Job	—	19 Jun 2018, 12:41 PM	✓ Enabled	⋮
notebook1	TelcoChurn_SparkML_jupyter.ipynb	App	Anyone with the link	19 Jun 2018, 5:14 PM	✓ Enabled	⋮

18. Now that all URLs and endpoints are active, we can test them.

- To test the Notebook URL, click on the ellipses next to notebook deployment, select **Share Endpoint**, and copy and paste it to a new browser window.

Check Python version. This notebook is implemented for Python 2.7.x. Not all cells may work in other versions of Python.

```
In [1]: import platform
print(platform.python_version())
```

2.7.13

Predicting Customer Churn in Telco

In this notebook you will learn how to build a predictive model with Spark machine learning API (SparkML) and deploy it for scoring in Machine Learning (ML).

This notebook walks you through these steps:

- To test the batch job, click on the deployment (row in the table), which will bring you to deployment details. Click **Start** from the dropdown menu, then click **Submit**. This issues the REST request to invoke the batch job. Verify that it was successful.

Request

Start

Environment variables +

Command line arguments +

Response

```

1 {
2   "description": "Successfully started the deployed job's run.",
3   "result": {
4     "_messageCode_": "success",
5     "jobExecution": {
6       "args": [],
7       "env": [],
8       "jobName": "telcobatch",
9       "remoteOptions": [],
10      "result": "Waiting",
11      "runId": "1529449386-990",
12      "runName": "telcobatch",
13      "startTime": 0,
14      "user": "990"
15    },
16    "message": "success"
17  }
18 }
```

If you click on the **Overview** tab of the deployment details, you will see the job runs that were invoked during testing.

ID	NAME	TARGET HOST	TRIGGERED BY	STARTED AT	DURATION (s)	RESULT
1529449386-990	telcobatch	Local instance	—	19 Jun 2018, 6:03 PM	78	Success
1529448958-990	telcobatch	Local instance	—	19 Jun 2018, 5:54 PM	77	Success
1529448269-990	telcobatch	Local instance	—	19 Jun 2018, 5:44 PM	85	Success

Optionally, you can click the **Generate Code** button, which shows the curl command for invoking the REST endpoint, and run it from ssh.

```

sh-4.2# curl -k -X POST \
> https://169.55.181.211/djob/v1/releasel/telcobatch/trigger \
> -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybWVudXN1IjoicmVsZWZ2ZTEiLCJpYXQiOiE1Mjk0Mjc3Mz19.T_jWzJ1tcw6MNbn52fZ-bCotCiWJ5MiyskSWznnncJw3uB8nCWePgJS4DzomEfkBULjiFBUNnIrEw93-HK-lKBchRxcgSF5TRQVPQid-GJ6F3KFU7U0d6_nFN5qcZr48qw2D1DQfyik0nkdTFYsIsfo0na-rD1iFIFy1U3IzWW5x-8J8rf1fX9Zj47KKW_vZs9xYWJEptfaqwzeNkAU4vCbDkFIz8p6FkuD3qSrxIzWbsUXbJ4pOMRyp9jF55KMZO9eaUbyqWSE-FUPHBD4y1qinr2aa-cDCbLln1x7CXtnrSboEtOxk-RCTbjs9praWCfy-qplkiKJ8Wre7Es_8zYaA' \
> -H 'Cache-Control: no-cache' \
> -H 'Content-Type: application/json' \
> -d '{"env": [], "args": []}'
{"description": "Successfully started the deployed job's run.", "result": {"_messageCode_": "success", "jobExecution": {"args": [], "env": [], "jobName": "telcobatch", "remoteOptions": [], "result": "Waiting", "runId": "1529449672-990", "runName": "telcobatch", "startTime": 0, "user": "990"}, "message": "success"}}sh-4.2#

```

- In this section we will test evaluation. Click on the **Dashboard** tab and review evaluation details (for example, timestamps for recent evaluations). This list will change after we run evaluation.

Switch to **Deployments** tab and click on the evaluation batch job.

Similar to batch job testing, select **Start** from the dropdown, then select **Submit**.

Navigate back to the **Dashboard** tab and scroll down to **Recent job runs** table. The evaluation job will run for 1-2 minutes. When it's done, the **Recent model evaluations** table will be updated.

Recent job runs					
ID	NAME	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT
1529451810-990	telcoeval	telcoeval	19 Jun 2018, 6:43 PM	In progress	Running

Recent model evaluations		
MODEL NAME	EVALUATION TIME	PERFORMANCE
Telco_Churn_ML_model	19 Jun 2018, 6:45 PM	Good
Telco_Churn_ML_model	3 May 2018, 4:41 PM	Good
Telco_Churn_ML_model	24 Apr 2018, 5:06 PM	Good

- In this section we will test online scoring. Click on Web service deployment and switch to the API tab. In the *Request* field, change every "INSERT VALUE" string to 0 (zero, without quotes).

Request

Function name *

score

Body *

```
{
  "input_json_str":
  [{"Status": "M", "CHURN": "F", "Usage": 28.14, "Paymethod": "CH", "Gender": "F", "Age": 25.14, "RatePlan": 1, "Children": 2, "LongDistanceBilltype": "Intl_discount", "CarOwner": "N", "Dropped": "INSERT_VALUE", "EstIncome": 52004.8, "International": "INSERT_VALUE", "LongDistance": 5.03, "Local": 23.11, "ID": 14, "LocalBilltype": "Budget"}]}
```

Request should look similar to this:

Request

Function name *

score

Body *

```
{
  "input_json_str":
  [{"Status": "M", "CHURN": "F", "Usage": 28.14, "Paymethod": "CH", "Gender": "F", "Age": 25.14, "RatePlan": 1, "Children": 2, "LongDistanceBilltype": "Intl_discount", "CarOwner": "N", "Dropped": 0, "EstIncome": 52004.8, "International": 0, "LongDistance": 5.03, "Local": 23.11, "ID": 14, "LocalBilltype": "Budget"}]}
```

Click **Submit**. Response is displayed.

Response

```
1 {
2   "result": [
3     [{"probabilities": [{"0.9343804275553431, 0.0656195724446568}], "classes": [{"F", "T"}], "predictions": [{"F"}]},
4     ""
5   ]
6 }
```

Optionally, you can click the **Generate Code** button, which shows the curl command for invoking the REST endpoint, and run it from ssh.

If you navigate back to deployment Overivew, you'll notice that invocation metrics have changed



You have finished the **Deployment in DSX** lab.

In this lab you completed the following steps:

- Tested assets in development environment
 - Created deployment definitions
 - Launched and tested deployment.
-