

Feature Engineering and Model Deployments using Watson Studio Local <https://developer.ibm.com/patterns/mo...>

#datascience #machine-learning #watson-studio #jupyter-notebook #pca-analysis

56 commits2 branches0 packages0 releases7 contributorsApache-2.0

Branch: masterNew pull requestCreate new fileUpload filesFind fileClone or download

imgbot and Steve Martinelli [ImgBot] Optimize images (#30)

data	clear notebooks	
doc/source/images	[ImgBot] Optimize images (#30)	
notebooks	clear notebooks	
scripts	added readme steps	2 years ago
.gitignore	removed HDP references	2 years ago
.travis.yml	added flake8 tests for scripts (#10)	2 years ago
ACKNOWLEDGEMENTS.md	updated acknowledgements	2 years ago
CONTRIBUTING.md	initial cleanup	2 years ago
LICENSE	initial cleanup	2 years ago
MAINTAINERS.md	initial cleanup	2 years ago
README.md	Update links in README (#29)	16 months ago

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

<https://github.com/IBM/model-mgmt-on->

Open in DesktopDownload ZIP

README.md

Automated Feature Engineering and Model Scoring using IBM Watson Studio Local

The goal of this code pattern is to demonstrate how data scientists can leverage IBM's Watson Studio Local to automate the:

- Periodic extraction of features (used to train the machine learning model) from distributed datasets.
- Batch scoring of the extracted features on the deployed model.

To illustrate this, an example data science workflow which classifies 3 different wine categories from the chemical properties of those wines is used in this code pattern.

- For feature extraction, Principal component analysis (PCA) is applied on the wine classification dataset and two principal components are extracted.
- For the classification model, Logistic regression (a popular machine learning model) is applied on the extracted components to predict the wine categories.

What is PCA? Principal component analysis (PCA) is a popular dimensionality reduction technique which is used to reduce N number of numerical variables into few principal components that are used as features in the machine learning model. These principal components capture a major percentage of the combined variance effect of all the variables.

What is IBM Watson Studio Local? Watson Studio Local is an on premises solution for data scientists and data engineers. It offers a suite of data science tools that integrate with RStudio, Spark, Jupyter, and Zeppelin notebook technologies.

What is the IBM Watson Machine Learning? Watson Machine Learning is a Watson Studio Local tool that provides users the ability to create and train machine learning models. Users can also deploy their models to make them available to a wider audience.

This repo contains two Jupyter notebooks illustrating how to extract features and build a model on the [wine classification data set](#). The data contains a list of wines with their associated chemical features and assigned wine classification. There are three scripts in the repo to automate the feature extraction, score the extracted features and combine the two steps as one wrapper respectively.

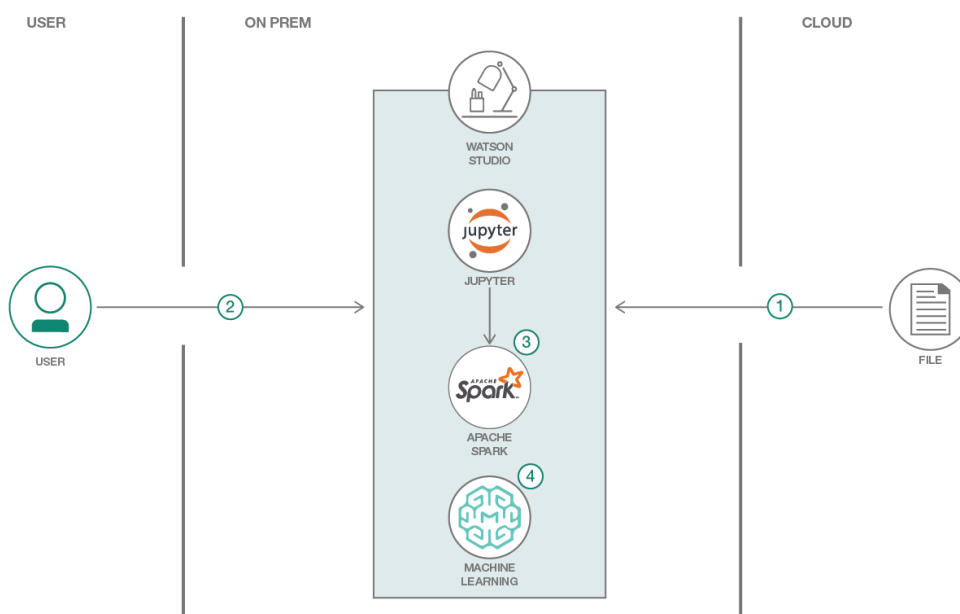
- The first notebook uses feature engineering techniques such as PCA and standard scaling to extract the features for the model development from a wine dataset.
- The second notebook trains, builds and saves a model that can be scored. The model can then be deployed and accessed remotely.
- The first script is simply a replica of the first notebook as the .py script which can be scheduled to extract the features periodically and save them with version numbers.
- The second script is used to score the set of extracted features on the deployed model and save the results with versioning.
- The third script is simply a wrapper script that runs the above two scripts one after the other and used to automate the feature extraction and model scoring in one run.

Using the Watson Machine Learning feature in Watson Studio, all three scripts are deployed as a service to automate feature extraction and scoring in production.

When you have completed this code pattern, you will understand how to:

- Use Watson Studio Local and to extract features using PCA and other techniques.
- Build, Train, and Save a model from the extracted features using Watson Studio Local.
- Use the Watson Machine Learning feature to deploy and access your model in batch and API mode
- Automate the feature extraction and model scoring using the scripts that are deployed as a service in batch and API mode.

Flow



1. The wine classification data set is loaded into Watson Studio Local as an asset.
2. The user interacts with Watson Studio Local to access assets such as Jupyter notebooks, python scripts, and data sets.
3. The Jupyter notebooks use Spark DataFrame operations to clean the dataset and use Spark MLlib to train a PCA classification model.
4. The model and accompanying scripts are deployed and run from Watson Machine Learning.

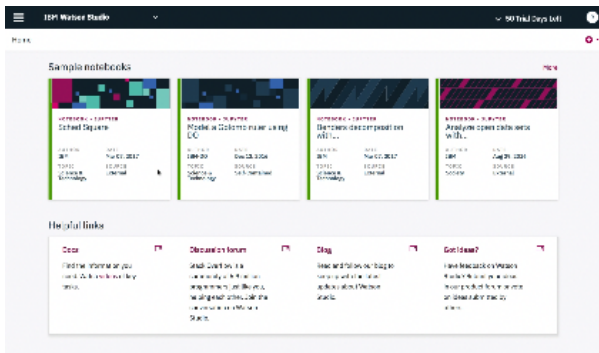
Included components

- **IBM Watson Studio Local:** An out-of-the-box on premises solution for data scientists and data engineers. It offers a suite of data science tools that integrate with RStudio, Spark, Jupyter, and Zeppelin notebook technologies.
- **Apache Spark:** An open-source, fast and general-purpose cluster computing system.
- **Jupyter Notebooks:** An open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.

Featured technologies

- **Artificial Intelligence:** Artificial intelligence can be applied to disparate solution spaces to deliver disruptive technologies.
- **Python:** Python is a programming language that lets you work more quickly and integrate your systems more effectively.

Watch the Video



Prerequisites

Install Watson Studio Local

Watson Studio Local provides a suite of data science tools, such as Spark and Jupyter notebooks, that are needed to complete this code pattern. Read the [Installation Guide](#) to install and configure your Watson Studio Local instance.

Steps

Follow these steps to setup the proper environment to run our notebooks locally.

1. [Clone the repo](#)
2. [Create project in IBM Watson Studio Local](#)
3. [Create project assets](#)
4. [Run the notebooks to create our model](#)
5. [Commit changes to Watson Studio Local Master Repository](#)
6. [Create release project in IBM Watson Machine Learning](#)
7. [Deploy our model as a web service](#)
8. [Deploy our scripts as a job](#)
9. [Bring deployments on-line](#)
10. [Gather API endpoints data for use in scripts](#)
11. [Modify scripts in Watson Studio Local](#)
12. [Run scripts locally to test](#)
13. [Manage your model with IBM Watson Machine Learning](#)

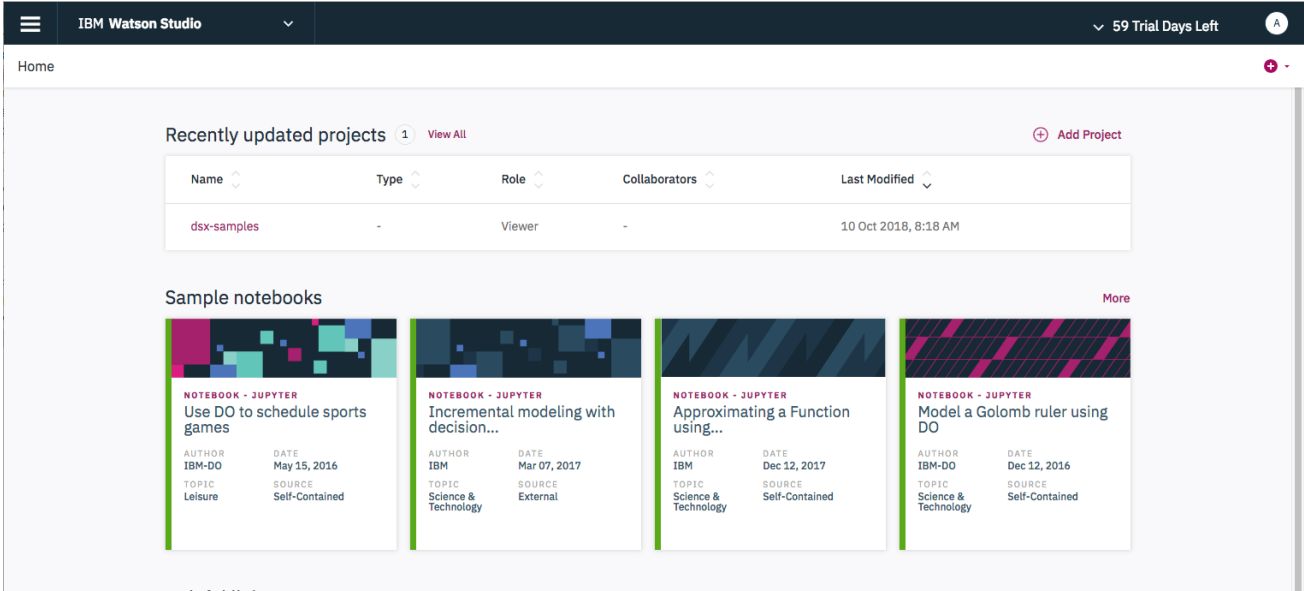
1. Clone the repo

```
git clone https://github.com/IBM/model-mgmt-on-watson-studio-local.git
```

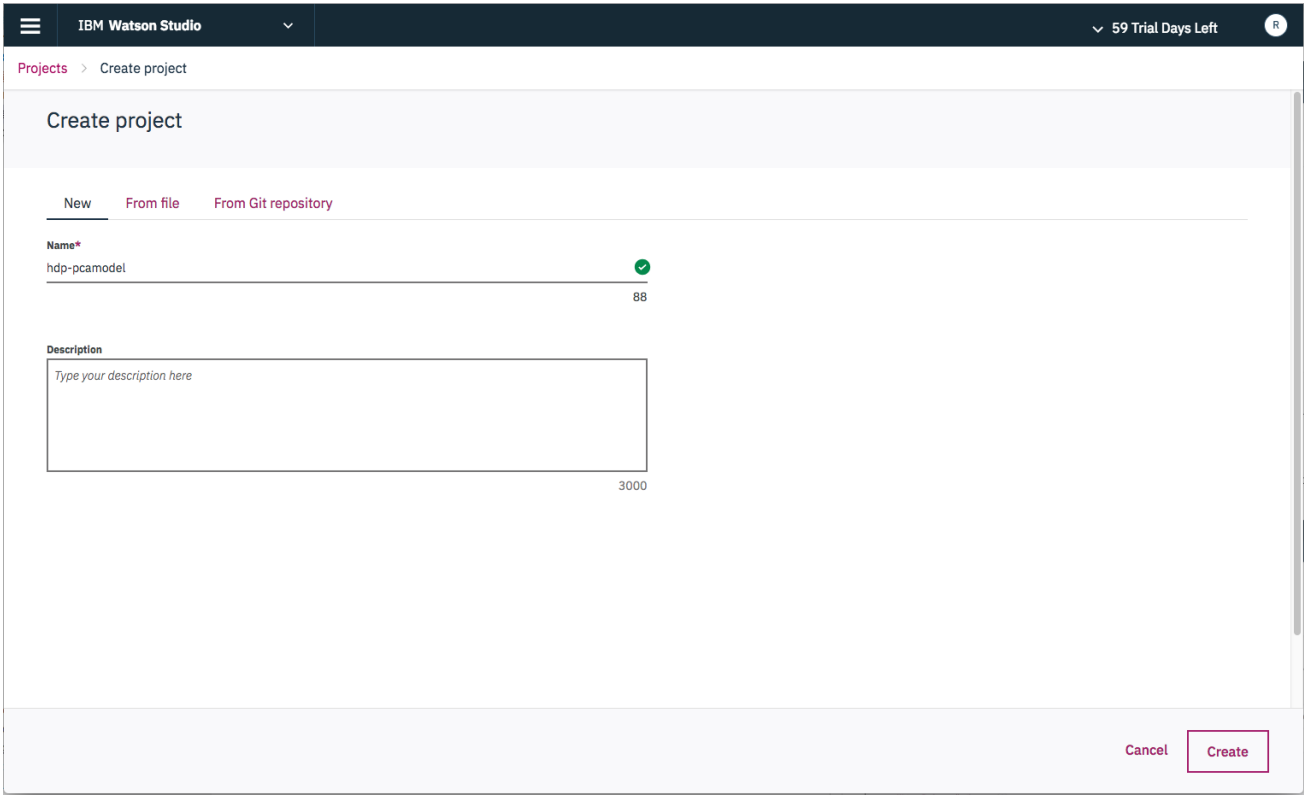
2. Create project in IBM Watson Studio Local

In Watson Studio Local, we use projects as a container for all of our related assets. To create a project:

- From the Watson Studio Local home page, select the **Add Project** button.



- Enter your project name and press the `Create` button.



3. Create project assets

Once created, you can view all of the project assets by selecting the `Assets` tab from the project's home page.

hdp-pcamodel

Assets 0 Data Sources 9 Jobs 0 Environments 7 Collaborators 1

Recent

- Data sets 0
- Notebooks 0
- RStudio 0
- Scripts 0
- Models 0
- Model groups 0
- Cognos dashboards 0
- Data Refinery flows 0
- Modeler flows 0
- Watson Explorer collections 0

Search by notebook name

Notebooks 0

+ Add Notebook

Name	Type	Environment	Last Modified
No notebooks available			

For our project, we need to add our notebooks and scripts. To add our notebooks:

- Select **Notebooks** in the project **Assets** list, then press the **Add Notebook** button.
- Enter a unique notebook name and use the **From URL** option to load the notebook from the github repo.
- Select **Environment** that runs under **Python v3.1** or greater.

NOTE: To make it easier to follow along with the remaining instructions, it is recommended that you set **Name** to match the name of the notebook in the uploaded file.

Create notebook

Blank From File From URL

Name*

pca-features

This name is valid 38

Description

Type your description here

500

Notebook URL

https://raw.githubusercontent.com/IBM/model-mgmt-on-watson-studio-local/master/notebooks/pca-features.ipynb

Environment*

Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1

Cancel Create

- Enter this URL:

<https://raw.githubusercontent.com/IBM/model-mgmt-on-watson-studio-local/master/notebooks/pca-features.ipynb>

- Repeat this step to add the second notebook, using the following URL:

<https://raw.githubusercontent.com/IBM/model-mgmt-on-watson-studio-local/master/notebooks/pca-modeling.ipynb>

To add our scripts:

- Select **Scripts** in the project **Assets** list, then press the **Add Script** button.

The screenshot shows the IBM Watson Studio interface for a project named 'hdp-pcamodel'. The top navigation bar includes the IBM Watson Studio logo, a dropdown menu, and a '59 Trial Days Left' indicator. The breadcrumb trail is 'Home > Projects > hdp-pcamodel'. The main content area shows the project details, including a 'Created by Rich Hagarty on 10 Oct 2018, 3:35 PM' timestamp. Below this, there are tabs for 'Assets' (2), 'Data Sources' (9), 'Jobs' (0), 'Environments' (7), and 'Collaborators' (1). The 'Assets' tab is active, displaying a list of assets. The 'Scripts' asset is highlighted, and the 'Add Script' button is visible. The 'Scripts' section shows a table with columns 'Name', 'Type', 'Path', and 'Last Modified', and a message 'No scripts available'.

- Click on the **From File** tab and then use the **Drag and Drop** option to load the script file from your local repo. The script name will be pre-populated with the name of the uploaded file. Select Python version 3.

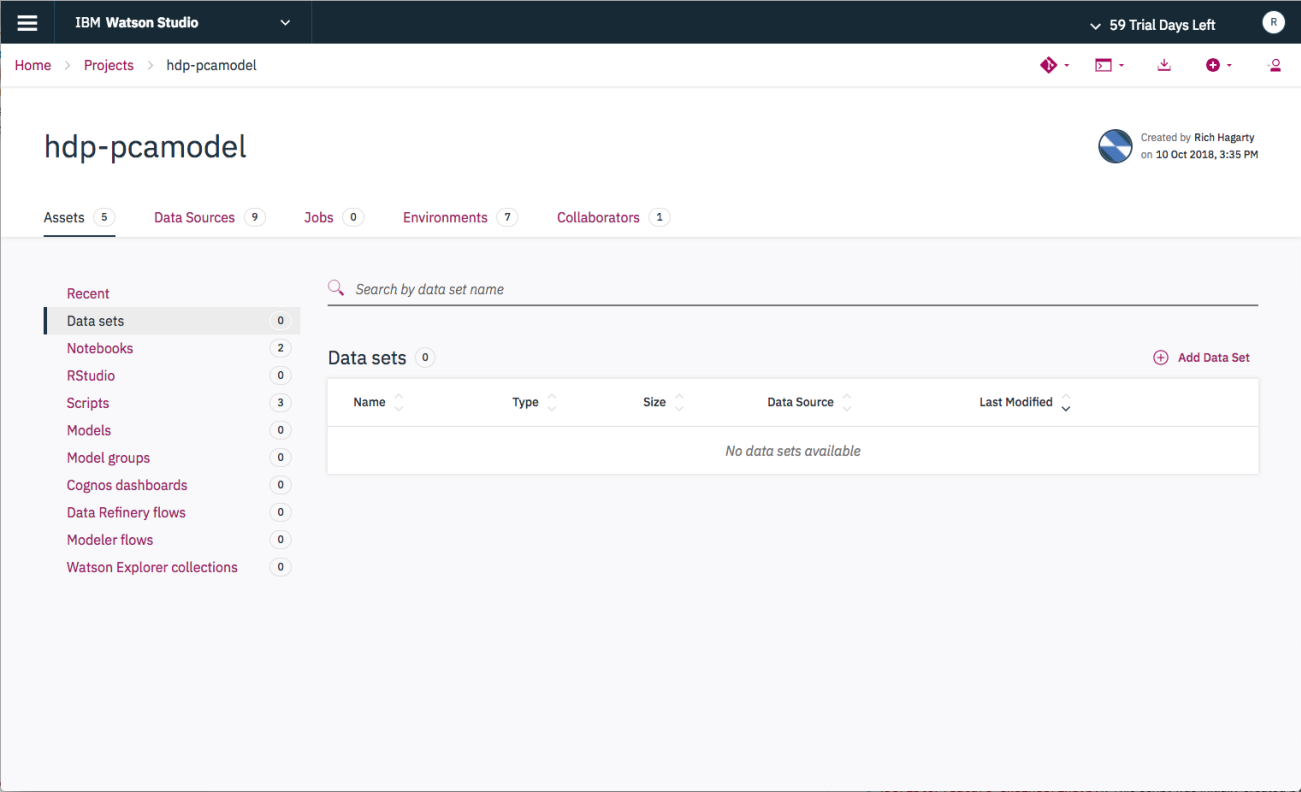
The screenshot shows the 'Create Script' dialog in IBM Watson Studio. The 'From File' tab is selected. The 'Name' field contains 'feature_engineering' and has a green checkmark indicating it is valid. The 'Description' field is empty. The 'Script File' field shows a green checkmark and the file name 'feature_engineering'. The 'Language' field is set to 'Python 3.5'. At the bottom right, there are 'Cancel' and 'Create' buttons.

- Add the following scripts:

```
scripts/feature_engineering.py
scripts/extract_and_score.py
scripts/model_scoring.py
```

To add our data set:

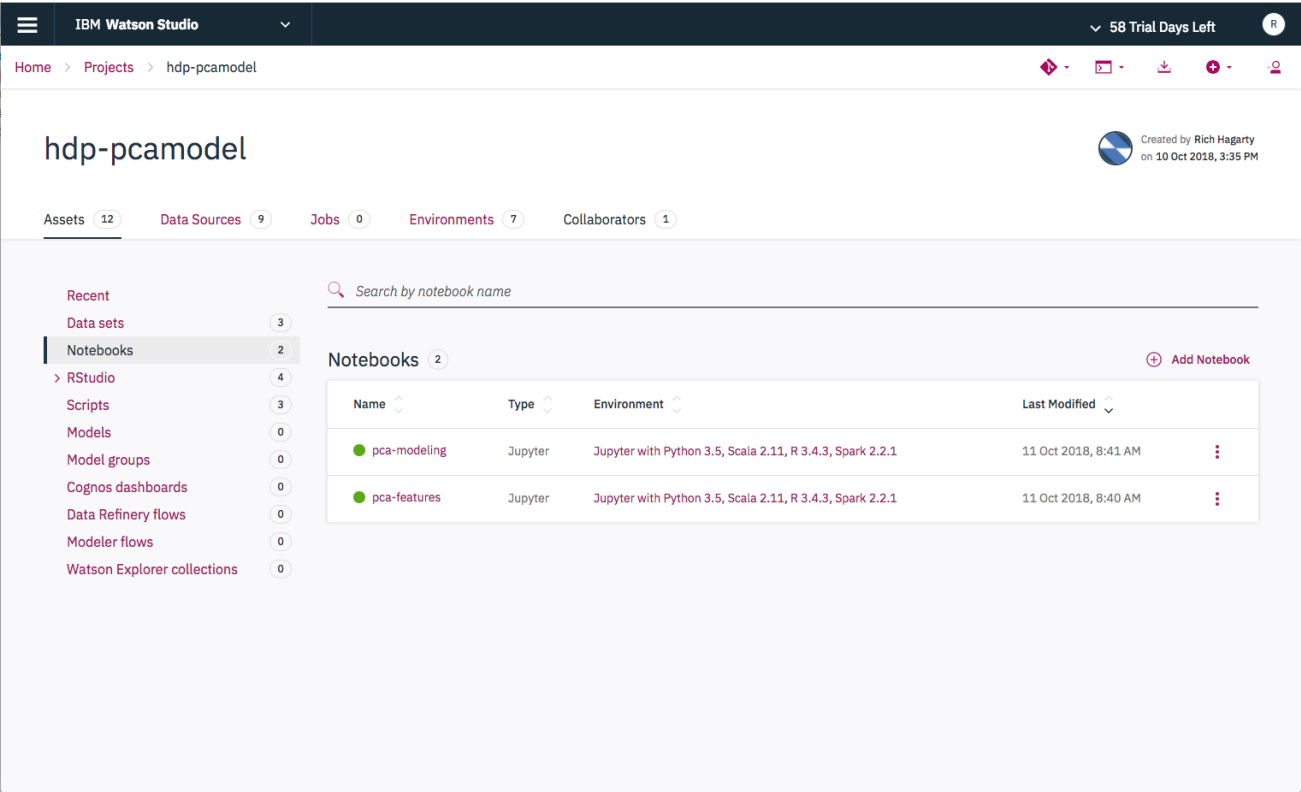
- Select `Data sets` in the project `Assets` list, then press the `Add Data Set` button.



From the `Local File` tab, click the `Select from your local file system` button to select the file `/data/Wine.csv` from your local repo.

4. Run the notebooks to create our model

To view our notebooks, select `Notebooks` in the project `Assets` list.



First, some background on how executing a notebooks:

When a notebook is executed, what is actually happening is that each code cell in the notebook is executed, in order, from top to bottom.

Each code cell is selectable and is preceded by a tag in the left margin. The tag format is `In [x]:`. Depending on the state of the notebook, the `x` can be:

- A blank, this indicates that the cell has never been executed.
- A number, this number represents the relative order this code step was executed.
- A `*`, which indicates that the cell is currently executing.

There are several ways to execute the code cells in your notebook:

- One cell at a time.
 - Select the cell, and then press the `Play` button in the toolbar.
- Batch mode, in sequential order.
 - From the `Cell` menu bar, there are several options available. For example, you can `Run All` cells in your notebook, or you can `Run All Below`, that will start executing from the first cell under the currently selected cell, and then continue executing all cells that follow.
- At a scheduled time.
 - Press the `Schedule` button located in the top right section of your notebook panel. Here you can schedule your notebook to be executed once at some future time, or repeatedly at your specified interval.

To run a notebook, simply click on the notebook name from the `Notebooks` list.

- Run the `pca-features` notebook first. It reads in and transforms the wine data set. It also creates data files that will be required by the next notebook. These `csv` data files can be viewed by selecting `Data sets` from the project `Assets` list.
- Run the `pca-modeling` notebook, which generates and saves our data model.

Once the model is created, you can view it by selecting `Models` in the project `Asset` list. Note that it is given a default version number.

The screenshot shows the IBM Watson Studio web interface. At the top, there's a header with the IBM Watson Studio logo and a trial status indicator. Below the header, the breadcrumb navigation shows 'Home > Projects > hdp-pcamodel'. The main content area is titled 'hdp-pcamodel' and includes a sub-header with 'Created by Rich Hagarty on 10 Oct 2018, 3:35 PM'. Below this, there's a navigation bar with tabs: 'Assets' (13), 'Data Sources' (9), 'Jobs' (0), 'Environments' (7), and 'Collaborators' (1). The 'Assets' tab is active, and within it, the 'Models' section is selected, showing a count of 1. A search bar labeled 'Search by model name' is present. Below the search bar, a table lists the models. The table has columns for 'Name', 'Type', 'Status', and 'Last Modified'. One model is listed: 'PCAModel v1', of type 'SciKit Learn', with status 'Trained', and last modified on '11 Oct 2018, 8:43 AM'. To the left of the table, there's a sidebar with a 'Recent' section listing various assets like 'Data sets', 'Notebooks', 'RStudio', 'Scripts', and 'Models' (which is highlighted with a count of 1). Other categories like 'Model groups', 'Cognos dashboards', 'Data Refinery flows', 'Modeler flows', and 'Watson Explorer collections' are also listed with counts of 0.

Name	Type	Status	Last Modified
PCAModel v1	SciKit Learn	Trained	11 Oct 2018, 8:43 AM

Note: After executing the notebooks, you may be wondering why we just didn't combine all of the code into just a single notebook. The reason is simply to separate out the data processing steps from the model creation steps. This allows us to process any new data in the future without effecting our current model. In fact, this is exactly what should be done with any new data - score it against the current model first to determine if the results are still acceptable. If not, we can then run the second notebook to generate a new model.

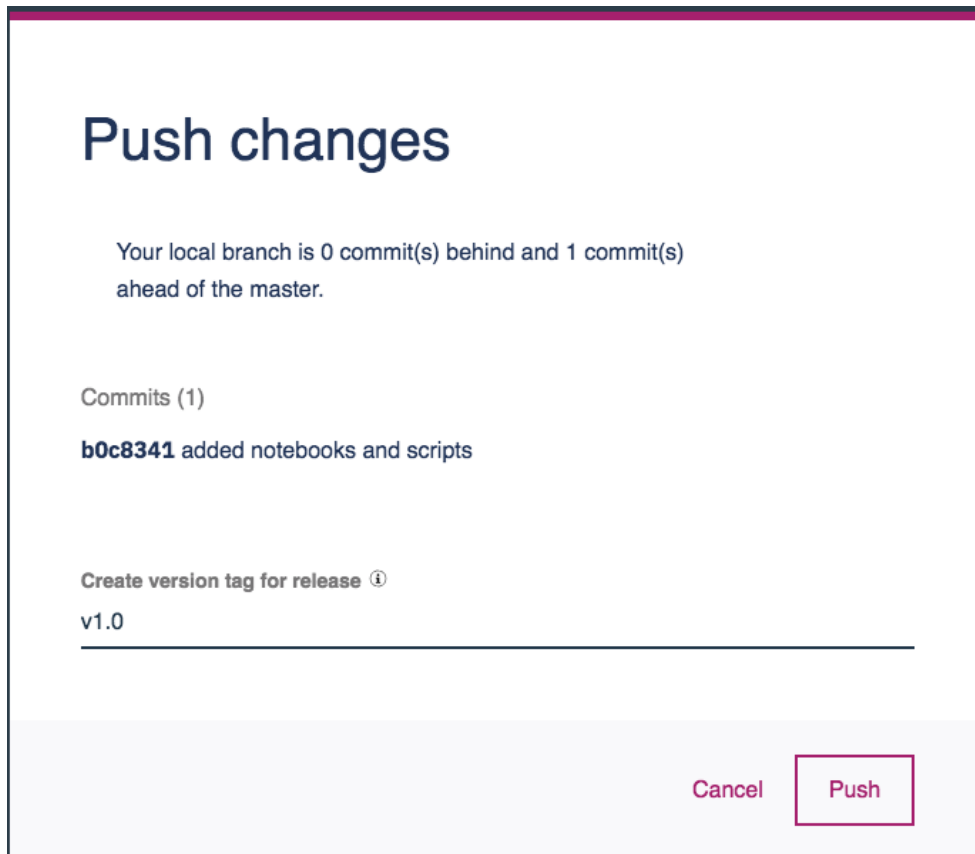
As you will see later, running the first notebook will be done by running a script in our project (`scripts/feature_engineering.py`). This script was initially created by loading the `pca-features` notebook into Jupyter, then exporting the notebook cells into a `python` script (use the menu options `File -> Download as -> Python (.py)`). We only had to modify the script slightly to include some code to handling data versioning.

5. Commit changes to Watson Studio Local Master Repository

After making changes to your project, you will be occasionally reminded to commit and push your changes to the Watson Studio Local Master Repository.

Changes made — You have local changes that you can **commit**.

Now that we have added our notebooks and scripts, and generated our model, let's go ahead and do that. Commit and push all of our new assets, and set the version tag to `v1.0` .



6. Create release project in IBM Watson Machine Learning

IBM Watson Machine Learning provides the mechanism to deploy our model as a web service. It manages `Project Releases` , which we will now create.

- Launch the IBM Watson Machine Learning tool by selecting it from the main drop-down menu on the Watson Studio Local home page.



- From the `Project releases` page, press the `Add Project Release` button.

IBM Watson Machine Learning

58 Trial Days Left

R

Project releases

Search by project release name

Project releases 0

Add Project Release

Name	Deployments	Members	Source Project	Last Updated
------	-------------	---------	----------------	--------------

- Select our previously committed project from the `Source project` drop-down list, and select the version tag you assigned to the project. Give the release a `Name` and a `Route` (which can be any random string), and the press `Create`.

IBM Watson Machine Learning

58 Trial Days Left

R

Project releases > Create project release

Create project release

From Watson Studio

From repository

From file

Name *

pca-model

91

Route * ⓘ

richh

21

Source project *

hdp-pcamodel

Tag *

v1.0

Cancel

Create

- If you click on the `Assets` tab, you will see all of the assets associated with the project.

IBM Watson Machine Learning

58 Trial Days Left

Project releases > pca-model

pca-model

Created by Rich Hagarty on Thu Oct 11 2018

Dashboard

Deployments

Assets

Data sources

Data sets

Active environments

Workers

Members

Search by asset name

All

extract_and_score.py
Script

feature_engineering.py
Script

model_scoring.py
Script

pca-features
Notebook

pca-modeling
Notebook

PCAModel
Model

extract_and_score.py

LANGUAGE: Python, CATEGORY: Other, TYPE: Other, RUNTIME: dsx-scripted-ml-python2

+

 job

+

 web service

Name	Asset	Type	Visibility	Date Started	Availability
No deployments found.					

7. Deploy our model as a web service

- Select the model from the list of Assets associated with our project. From the model details panel, press the web service button.

IBM Watson Machine Learning

58 Trial Days Left

Project releases > pca-model

pca-model

Created by Rich Hagarty on Thu Oct 11 2018

Dashboard

Deployments

Assets

Data sources

Data sets

Active environments

Workers

Members

Search by asset name

All

extract_and_score.py
Script

feature_engineering.py
Script

model_scoring.py
Script

pca-features
Notebook

pca-modeling
Notebook

PCAModel
Model

PCAModel

VERSIONS: 1, LAST MODIFIED: 11 Oct 2018, 8:43 AM, TYPE: scikit-learn-0.19, ENGINE: Python35, ALGORITHM: Classification

+

 web service

Name	Asset	Type	Visibility	Date Started	Availability
No deployments found.					

- On the model deployment screen, provide a unique name, reserve some CPUs and memory, then press Create .

IBM Watson Machine Learning

58 Trial Days Left

Project releases > pca-model > Deploy PCAModel as a web service

Deploy PCAModel as a web service

Name *

pcamodel

18

URL

https://yc375-master-1.fyre.ibm.com/dmodel/v1/richh/pyscript/pcamodel

Model version *

Use latest version

Web service environment *

Python 3.5 - Script as a Service

☒ Reserve CPU cores

1

☒ Reserve GB Memory

2

Replicas

1

Cancel

Create

8. Deploy our scripts as a job

- From the details panel for the `extract_and_score.py` script, press the `job` button.

IBM Watson Machine Learning

58 Trial Days Left

Project releases > pca-model

pca-model

Created by Rich Hagarty on Thu Oct 11 2018

Dashboard

Deployments

Assets

Data sources

Data sets

Active environments

Workers

Members

Search by asset name

All

extract_and_score.py
Script

feature_engineering.py
Script

model_scoring.py
Script

pca-features
Notebook

pca-modeling
Notebook

PCAModel
Model

extract_and_score.py

LANGUAGE

Python

CATEGORY

Other

TYPE

Other

RUNTIME

dsx-scripted-ml-python2

job

web service

Name	Asset	Type	Visibility	Date Started	Availability
No deployments found.					

- On the script deploy screen, provide a job name, set the type to `Script run`, add `v1` as a command line argument, then press `Create`.

IBM Watson Machine Learning

58 Trial Days Left

Project releases

pca-model

Deploy extract_and_score.py as a job

Deploy extract_and_score.py as a job

Name *

extract-and-score

9

URL

https://yc375-master-1.fyre.ibm.com/djob/v1/richh/extract-and-score

Description

Job description

300

Type *

Script run

Worker *

Jupyter with Python 3.5, Scala 2.11, R 3.4.3

Target host *

Local instance

Environment variables +

VARIABLE_1=value 1

100

Command line arguments +

v1

98

Scheduled to run

On demand

Every

day

at

12:00 AM

Cancel

Create

Repeat these steps for the remaining 2 scripts.

9. Bring deployments on-line

If you select the `Deployments` tab from the project page, you will notice that all of the deployments are listed as disabled.

Project releases > pca-model

Created by Rich Hagarty on Thu Oct 11 2018

Dashboard Deployments Assets Data sources Data sets Active environments Workers Members

Search by deployment name

Name	Asset	Type	Visibility	Date Started	Availability
pcamodel	PCAModel v1	Web service	—	11 Oct 2018, 9:51 AM	Disabled
model-scoring	model_scoring.py	Job	—	11 Oct 2018, 1:21 PM	Disabled
feature-engineering	feature_engineering.py	Job	—	11 Oct 2018, 1:20 PM	Disabled
extract-and-score	extract_and_score.py	Job	—	11 Oct 2018, 1:19 PM	Disabled

To bring the deployments on-line, press the **Launch** button icon, which is the left-most icon listed at the top of the page. Once you complete the action, you should see the following.

Project releases > pca-model

Created by Rich Hagarty on Thu Oct 11 2018

Dashboard Deployments Assets Data sources Data sets Active environments Workers Members

Search by deployment name

Name	Asset	Type	Visibility	Date Started	Availability
pcamodel	PCAModel v1	Web service	—	11 Oct 2018, 9:51 AM	Enabled
model-scoring	model_scoring.py	Job	—	11 Oct 2018, 1:21 PM	Enabled
feature-engineering	feature_engineering.py	Job	—	11 Oct 2018, 1:20 PM	Enabled
extract-and-score	extract_and_score.py	Job	—	11 Oct 2018, 1:19 PM	Enabled

Note: you may have to manually **Enable** the model deployment by using the menu options listed on the right side of the model row in the deployments table.

10. Gather API endpoints data for use in scripts

Some of our scripts will need access to our deployed model and, in some cases, to the other deployed scripts.

Here is a quick overview of what our scripts do and require:

- **feature_engineering.py** - this script performs the same function as our `pca-features` notebook. It reads in data from the wine dataset, then applies machine learning techniques to transform the data. The output will be two data files - `features` and `target`. Note that these data files will have a version tag appended to them so that their contents are not over-written every time that this script is run. This script does not require access to our model or other scripts.
- **model_scoring.py** - this script is a batch processor. It reads the `features` data file and scores each feature, one at a time, by running them through our model. The output will be a `scoring_output` data file. Note that the data file will have a version tag appended to it so that its contents are not over-written every time that this script is run. This script requires access to our deployed model.
- **extract_and_score.py** - this script was created for convenience, and can be used instead of running the previous scripts separately. It invokes the `feature_engineering` script first, then when complete, it invokes the `model_scoring` script. This script requires access to the other deployed scripts.

To access the deployed model or scripts, we need to gather the associated API endpoints and an authorization token. All of these values are available from the deployment pages for each of the objects. The endpoints will take on the following format:


- model endpoints will end in `/score`.
- script endpoints will end in either `/trigger`, `/status`, or `/cancel` (which corresponds to the actions: `start`, `status`, and `stop`).

To get the endpoint for our deployed model, click on the model from the `Assets` tab of the project page.

The screenshot shows the IBM Watson Machine Learning interface. At the top, there's a header with the IBM logo, 'IBM Watson Machine Learning', and a dropdown menu. On the right, it says '58 Trial Days Left' and a user icon. Below the header, the breadcrumb trail is 'Project releases > pca-model > pcamodel'. The main content area is titled 'pcamodel' and shows it was 'Created by Rich Hagarty on 11 Oct 2018, 9:51 AM'. There are two tabs: 'Overview' (selected) and 'API'. The 'Overview' tab displays several key metrics:

- ENDPOINT:** POST `https://yc375-master-1.fyre.ibm.com/dmodel/v1/richh/pyscript/pcamodel/score` (with a copy icon)
- DEPLOYMENT TOKEN:** (with a copy icon)
- TYPE:** Web service
- ASSET:** PCAModel v1
- ALLOCATED CPU:** 1.0 cores
- ALLOCATED MEMORY:** 2.0 GB
- REQUESTS:** 0

 Below this, there are two sections: 'Request' and 'Response'. The 'Request' section has a 'Function name' field with 'score' and a 'Body' field with a JSON input: `{\"input_json\": [{\"0\": 2.1469878236, \"1\": -1.0167515378}]}`. The 'Response' section is empty. There is a 'Generate Code' button in the top right of the 'Response' section.

The endpoint is listed at the top of the page. Both the `Endpoint` and the `Deployment Token` can be saved to the clipboard by clicking on their respective  icons.

Repeat this step to retrieve the endpoints for both the `feature_engineering` and `model_scoring` scripts.

Note: You will only need to get one copy of the `Deployment Token`. It will be the same for all deployments within this project.

11. Modify scripts in Watson Studio Local

Once we have gathered our deployment endpoints and deployment token, we need to go back to `Watson Studio Local` mode so that we can modify and test our scripts.

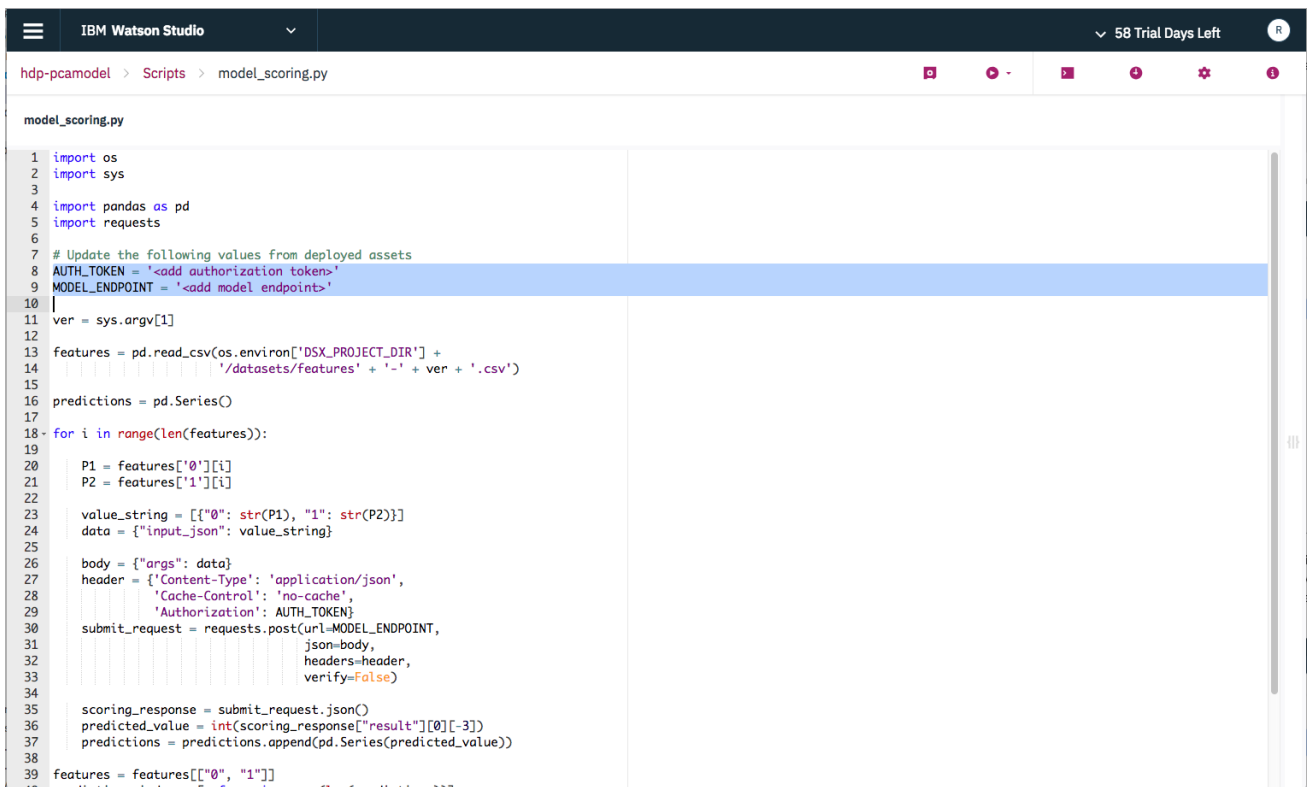


Go to the **Assets** list for the project, and select **Scripts**. Click on a script file to open it up in edit mode.

The scripts we will be modifying are those that reference deployment objects. These are:

```
scripts/model_scoring.py
scripts/extract_and_score.py
```

- For the `model_scoring` script, substitute in the token and endpoint values, then save the new version of the script.



- For the `extract_and_score` script, substitute in the token and endpoint values. The endpoint values are for the `feature_engineering` and `model_scoring` deployment scripts.

Important: The endpoints will end with the string `/trigger`. Delete that portion of the URL as the script will append either `/trigger` or `/status` to the endpoints, as needed.

- Finish by saving the new version of the script by clicking on the **Save** icon located at the top right side of the page.


```

1 # Trigger the feature extraction from wine dataset
2
3 import requests
4 import sys
5 import time
6
7 # Update the following values from deployed assets
8 AUTH_TOKEN = '<add authorization tokens>'
9 MODEL_SCORING_ENDPOINT = '<add model_scoring script endpoints>'
10 FEATURE_ENGINEERING_ENDPOINT = '<add feature_engineering script endpoints>'
11
12 args = sys.argv[1]
13
14 body = {"env": [], "args": [args]}
15
16 header = {'Content-Type': 'application/json',
17           'Cache-Control': 'no-cache',
18           'Authorization': AUTH_TOKEN}
19
20 url_ef = FEATURE_ENGINEERING_ENDPOINT + '/trigger'
21
22 extract_features = requests.post(url=url_ef,
23                                 json=body,
24                                 headers=header,
25                                 verify=False)
26
27 response_ef = extract_features.json()
28 print(response_ef)
29
30 # Check the status of the extract features request
31
32 result_ef = response_ef["result"]
33
34 jobId_ef = str(result_ef['jobExecution']['runId'])
35 print(jobId_ef)
36
37 status_ef = str(result_ef['jobExecution']['result'])
38 print(status_ef)
39
40

```

12. Run scripts locally to test

To avoid having to go back and forth between Watson Studio Local and Watson Machine Learning (which includes re-deploying and creating new release versions), make sure the scripts run locally in Watson Studio Local first.

Scripts can be run from either the detail panel for the script, or from the script editor.

1. Run as a job from the script detail panel:

- From the project page, click on **Scripts** in the **Assets** list.

hdp-pcamodel

Assets 13 Data Sources 9 Jobs 0 Environments 7 Collaborators 1

Recent

- Data sets 3
- Notebooks 2
- > RStudio 4
- Scripts 3**
- Models 1
- Model groups 0
- Cognos dashboards 0
- Data Refinery flows 0
- Modeler flows 0
- Watson Explorer collections 0

Search by script name

Scripts 3

Name	Type	Path	Last Modified
extract_and_score.py	Python	/scripts/extract_and_score.py	11 Oct 2018, 1:57 PM
model_scoring.py	Python	/scripts/model_scoring.py	11 Oct 2018, 1:44 PM
feature_engineering.py	Python	/scripts/feature_engineering.py	10 Oct 2018, 3:58 PM

+ Add Script

Start with the script `feature_engineering`. Use the menu bar on the right side of the script row to **Create Job** and run the script.

In the **Create Job** run panel, provide a unique name and make sure you use the following options:

- Type: Script Run
- Worker: Python 3
- Source asset: /scripts/feature_engineering.py
- Command line arguments: v1
- Scheduled to run: On demand

After you press the **Create** button, you will see the run panel.

featurerun Created by Rich Hagarty on 11 Oct 2018, 2:02 PM

TYPE	WORKER	TARGET HOST	SCHEDULED TO RUN
Script run	Jupyter with Python 3.5, Scala 2.11, R 3.4.3	Local instance	On demand

Source asset

```

1 # coding: utf-8
2
3 import os
4 import sys
5
6 ##### Importing the libraries
7 import pandas as pd
8 from sklearn.preprocessing import StandardScaler
9 from sklearn.decomposition import PCA
10
11 # Setting Dynamic Variables
12 target_v1 = os.getenv('target_v1')
13 features_v1 = os.getenv('features_v1')

```

Default environment variables

Variable	Value

Default command line arguments

Array Index	Argument

If you scroll down a bit, you will see a **Run Now** button. Click it to start the script. Again, you will be presented with a dialog that requires you give it a run name. The rest of the values will be defaulted to values you already set, so they do not need to be modified. Click the **Run** button to run the script.

You will be transferred back to the run panel where you can see the status (listed under **Duration**) and a tail of the log file. Once completed successfully, you should see 2 new files listed under **Data sets** in the **Assets** list.

hdp-pcamodel Created by Rich Hagarty on 10 Oct 2018, 3:35 PM

Assets (15) Data Sources (9) Jobs (1) Environments (7) Collaborators (1)

Recent

- Data sets (5)
- Notebooks (2)
- RStudio (4)
- Scripts (3)
- Models (1)
- Model groups (0)
- Cognos dashboards (0)
- Data Refinery flows (0)
- Modeler flows (0)
- Watson Explorer collections (0)

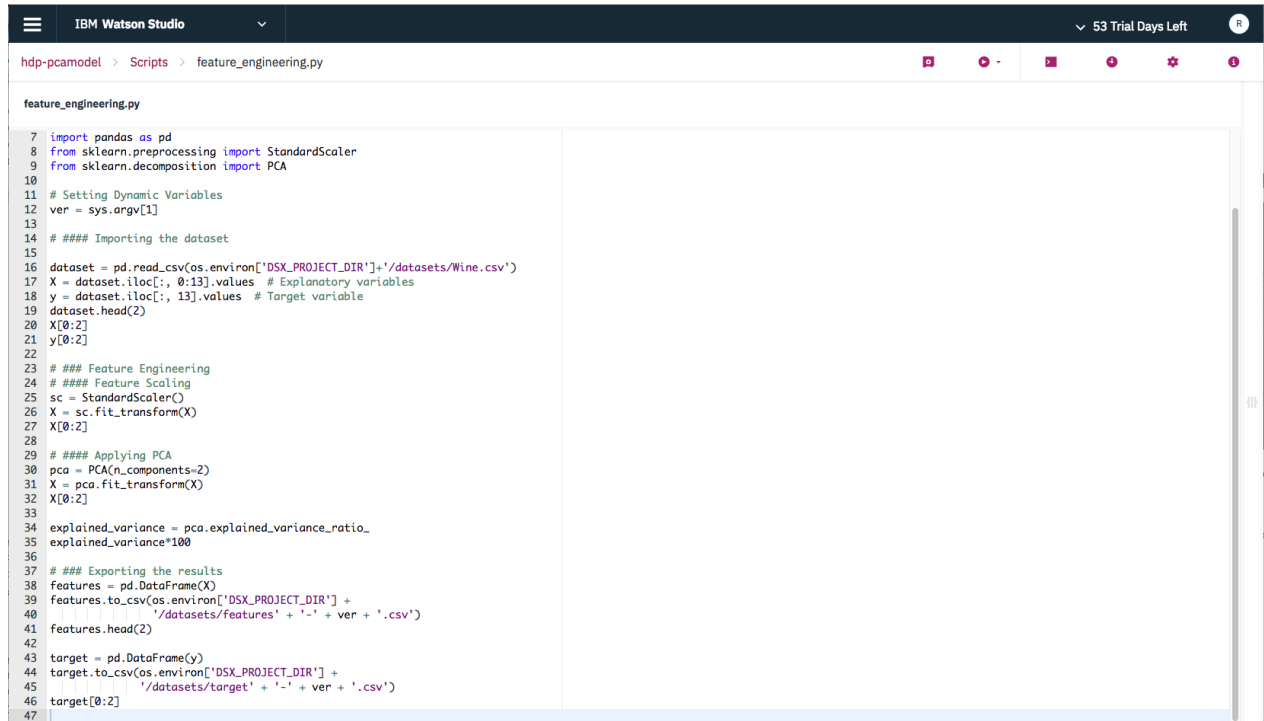
Data sets (5)

Name	Type	Size	Data Source	Last Modified
target-v1.csv	CSV	961 B	Local file	11 Oct 2018, 2:05 PM
features-v1.csv	CSV	7.29 KB	Local file	11 Oct 2018, 2:05 PM
target.csv	CSV	961 B	Local file	11 Oct 2018, 8:40 AM
features.csv	CSV	7.29 KB	Local file	11 Oct 2018, 8:40 AM
Wine.csv	CSV	11.19 KB	Local file	10 Oct 2018, 4:18 PM

NOTE: The created data files (`target` and `features`) will have a version tag appended to their name. This matches the command line argument we passed into the script.

2. Run interactively from the script editor

- from the project page, click on `Scripts` in the `Assets` list, then click on the `feature_engineering` script in the list. This will bring up the script editor.

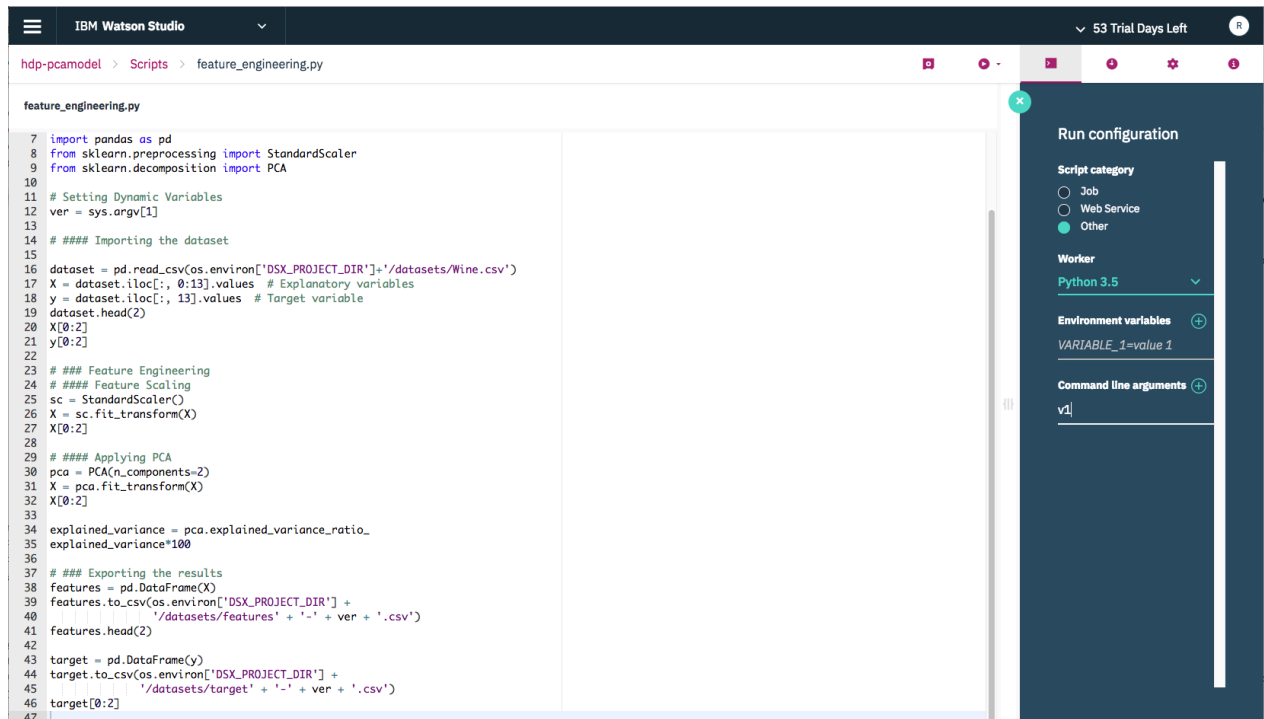


```

feature_engineering.py
7 import pandas as pd
8 from sklearn.preprocessing import StandardScaler
9 from sklearn.decomposition import PCA
10
11 # Setting Dynamic Variables
12 ver = sys.argv[1]
13
14 ##### Importing the dataset
15
16 dataset = pd.read_csv(os.environ['DSX_PROJECT_DIR']+'/datasets/Wine.csv')
17 X = dataset.iloc[:, 0:13].values # Explanatory variables
18 y = dataset.iloc[:, 13].values # Target variable
19 dataset.head(2)
20 X[0:2]
21 y[0:2]
22
23 ##### Feature Engineering
24 ##### Feature Scaling
25 sc = StandardScaler()
26 X = sc.fit_transform(X)
27 X[0:2]
28
29 ##### Applying PCA
30 pca = PCA(n_components=2)
31 X = pca.fit_transform(X)
32 X[0:2]
33
34 explained_variance = pca.explained_variance_ratio_
35 explained_variance*100
36
37 ### Exporting the results
38 features = pd.DataFrame(X)
39 features.to_csv(os.environ['DSX_PROJECT_DIR'] +
40               '/datasets/features' + '-' + ver + '.csv')
41 features.head(2)
42
43 target = pd.DataFrame(y)
44 target.to_csv(os.environ['DSX_PROJECT_DIR'] +
45              '/datasets/target' + '-' + ver + '.csv')
46 target[0:2]
47

```

Click the `Run` configuration button in the tool bar and then add `v1` as a command line argument.



```

feature_engineering.py
7 import pandas as pd
8 from sklearn.preprocessing import StandardScaler
9 from sklearn.decomposition import PCA
10
11 # Setting Dynamic Variables
12 ver = sys.argv[1]
13
14 ##### Importing the dataset
15
16 dataset = pd.read_csv(os.environ['DSX_PROJECT_DIR']+'/datasets/Wine.csv')
17 X = dataset.iloc[:, 0:13].values # Explanatory variables
18 y = dataset.iloc[:, 13].values # Target variable
19 dataset.head(2)
20 X[0:2]
21 y[0:2]
22
23 ##### Feature Engineering
24 ##### Feature Scaling
25 sc = StandardScaler()
26 X = sc.fit_transform(X)
27 X[0:2]
28
29 ##### Applying PCA
30 pca = PCA(n_components=2)
31 X = pca.fit_transform(X)
32 X[0:2]
33
34 explained_variance = pca.explained_variance_ratio_
35 explained_variance*100
36
37 ### Exporting the results
38 features = pd.DataFrame(X)
39 features.to_csv(os.environ['DSX_PROJECT_DIR'] +
40               '/datasets/features' + '-' + ver + '.csv')
41 features.head(2)
42
43 target = pd.DataFrame(y)
44 target.to_csv(os.environ['DSX_PROJECT_DIR'] +
45              '/datasets/target' + '-' + ver + '.csv')
46 target[0:2]
47

```

Run configuration

Script category

- ☐ Job
- ☐ Web Service
- ☒ Other

Worker

Python 3.5

Environment variables

VARIABLE_1=value 1

Command line arguments

v1

Drag the sizing icon located on the right side of the panel to open up the console log. Then start the script by clicking on the `Run -> Run as job` button in the tool bar.

The script will begin executing and displaying output to the console log.

The screenshot shows the IBM Watson Studio interface. On the left, a Python script named `feature_engineering.py` is displayed. The script imports `pandas` and `sklearn` libraries, loads a dataset from `Wine.csv`, performs feature engineering (scaling and PCA), and exports the results to `features.csv` and `target.csv`. On the right, the execution output is shown, indicating a successful run with the command `Run 1539731003-1003`. The output details the file path `/var/run/secrets/kubernetes.io` and provides metadata such as size, blocks, IO block, device, inode, links, access, modify, change, and birth times.

Using either method, repeat this process to run the `model_scoring` script, and then the `extract_and_score` script.

NOTE: The `extract_and_score` script will fail when it attempts to check the status of the `model_scoring` script. Remember, it invokes the other scripts by calling their endpoints which are deployed in Watson Machine Learning. The problem is that those deployed scripts do not have the updated endpoint information yet, and won't until we push our changes back up to the Watson Machine Learning release model (which we will do next).

Once you have verified the scripts, commit and push the changes to the Watson Studio Master Repository, as described above in [Step #5](#). Make sure you bump the version number.

13. Manage your model with IBM Watson Machine Learning

Launch IBM Watson Machine Learning by selecting it from the main drop-down menu on the Watson Studio Local home page.



First we need to update our release project to grab all of the latest versions of our scripts.

- From the Watson Machine Learning home page, click on our project tile.
- From the row of icons listed in the page banner, click on the `Update` icon.
- From the update screen, use the `Source` project drop-down menu to select our Watson Studio Local project. Then select the version tag associated with our latest commit.

Now that all of our assets are updated, we can actually manage our model by tracking its use and performance.

To get the ball rolling, let's start by running our `extract_and_score` script. As explained previously, this will read in the current wine csv file and transform it, then score each wine against our model.

- From the `Asset` page, select the `extract_and_score` script. From the detail page, click on the script name to bring up the script launch page. Then click on the `API` tab.

IBM Watson Machine Learning

58 Trial Days Left

Project releases > pca-model > extract-and-score

extract-and-score

Created by Rich Hagarty on 11 Oct 2018, 1:19 PM

ENDPOINT

POST https://yc375-master-1.fyre.ibm.com/djob/v1/richh/extract-and-score/trigger

DEPLOYMENT TOKEN

TYPE

Job

ASSET

extract_and_score

ALLOCATED CPU

Unallocated

ALLOCATED MEMORY

Unallocated

TARGET HOST

Local instance

Overview

API

Request

Start

Response

<> Generate Code

Environment variables

+

Command line arguments

+

v1

- Note that the Command line arguments value is set to v1 . You can set this to anything you want, but it will be appended to the file names generated by this script. This is how to avoid overriding the data from previous runs of this script.
- To run the script, click on the Overview tab and scroll down to the Runs section of the page. Then click the run now button.
- From the start dialog, enter a name, modify the command line argument if needed, then click the Run button.

Run extract-and-score

Name *

extract-and-score-job

79

Target host *

Local instance

Environment variables

+

VARIABLE_1=value 1

Command line arguments

+

v1

Cancel

Run

You can view the status of the job from the same panel. To see the associated log file, click the `View logs` menu option located on the right side of the job row.

IBM Watson Machine Learning

53 Trial Days Left

Project releases > pca-model > extract-and-score

extract-and-score

Created by Rich Hagarty on 16 Oct 2018, 3:22 PM

ENDPOINT
POST https://yc375-master-1.fyre.ibm.com/djob/v1/richh/extract-and-score/trigger

DEPLOYMENT TOKEN

TYPE
Job

ASSET
extract_and_score

ALLOCATED CPU
Unallocated

ALLOCATED MEMORY
Unallocated

TARGET HOST
Local instance

Overview

API

Default environment variables

Default command line arguments

Runs

run now

Id	Name	Target Host	Triggered By	Started At	Duration (S)	Result
1539733734-990	extract-and-score	Local instance	—	16 Oct 2018, 4:48 PM	58	Success

From the main dashboard, you can also see that the script completed successfully, and also note that in the case of the `extract-and-score` script, three separate jobs were launched (the main script, and the two invoked scripts).

IBM Watson Machine Learning

53 Trial Days Left

Project releases > pca-model

pca-model

Created by Rich Hagarty on Tue Oct 16 2018

Dashboard

Deployments

Assets

Data sources

Data sets

Active environments

Workers

Members

Current model metrics

Recent model evaluations

Top deployments by requests

Recent job runs

0 evaluated models

0 Good
0 Fair
0 Poor

Model Name

Evaluation Time

Performance

no model evaluations found

Deployment

Type

Date Deployed

Requests

pca-model

Web service

11 Oct 2018, 2:15 PM

73

Id

Name

Triggered By

Started At

Duration (S)

Result

1539733753-990

model-scoring

model-scoring

16 Oct 2018, 4:49 PM

22

Success

1539733737-990

feature-engineering

feature-engineering

16 Oct 2018, 4:48 PM

10

Success

1539733734-990

extract-and-score

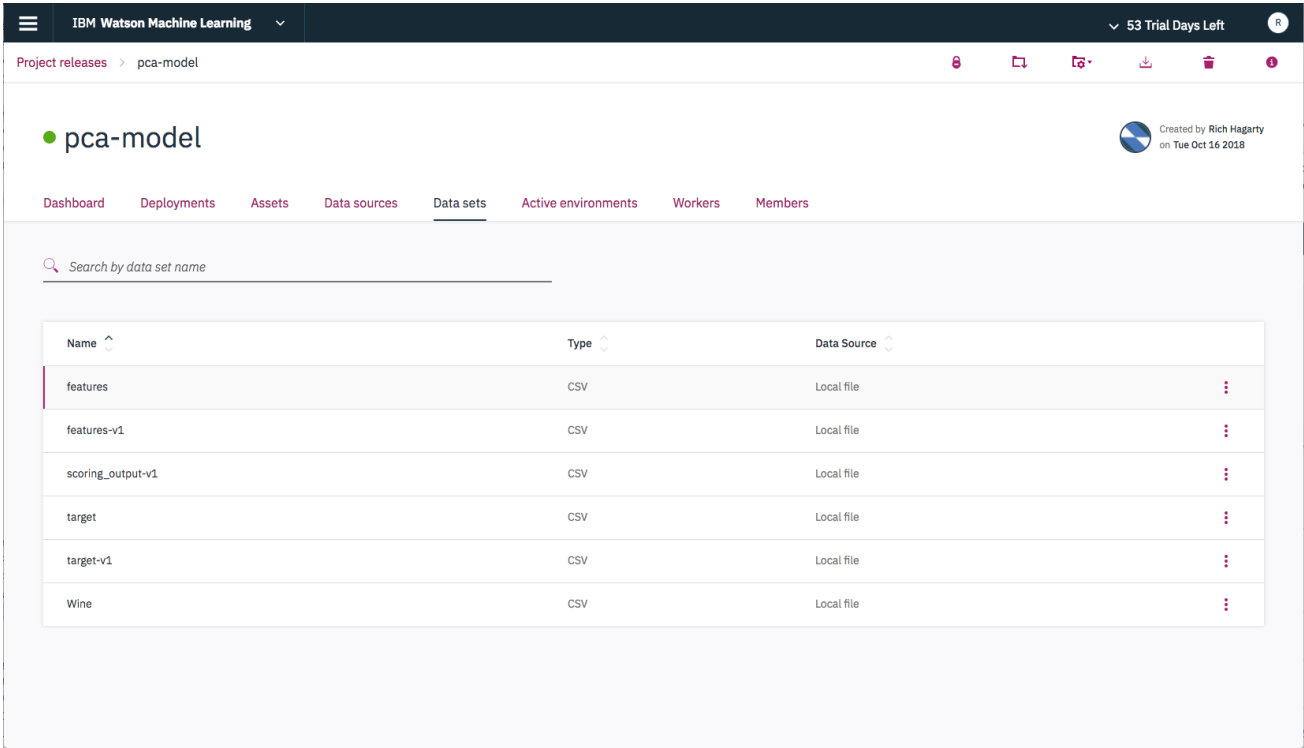
extract-and-score

16 Oct 2018, 4:48 PM

58

Success

If the script completes succesfully, there should be a three versioned data files that should be added as `Data sets` assets in the project. The `scoring_output-v1` file contains the final wine classification scores.



Sample output

Here are links to each of the notebooks with example output:

- [pca-features](#)
- [pca-modeling](#)

Links

- [IBM Watson Studio Local guided demo: Build, train, and deploy a machine-learning model without coding](#)

Learn more

- Data Analytics Code Patterns:** Enjoyed this Code Pattern? Check out our other [Data Analytics Code Patterns](#)
- AI and Data Code Pattern Playlist:** Bookmark our [playlist](#) with all of our Code Pattern videos
- Watson Studio:** Master the art of data science with IBM's [Watson Studio](#)
- Spark on IBM Cloud:** Need a Spark cluster? Create up to 30 Spark executors on IBM Cloud with our [Spark service](#)

License

This code pattern is licensed under the Apache Software License, Version 2. Separate third party code objects invoked within this code pattern are licensed by their respective providers pursuant to their own separate licenses. Contributions are subject to the [Developer Certificate of Origin, Version 1.1 \(DCO\)](#) and the [Apache Software License, Version 2](#).

[Apache Software License \(ASL\) FAQ](#)