



LAB

04 – Cloudera Machine Learning

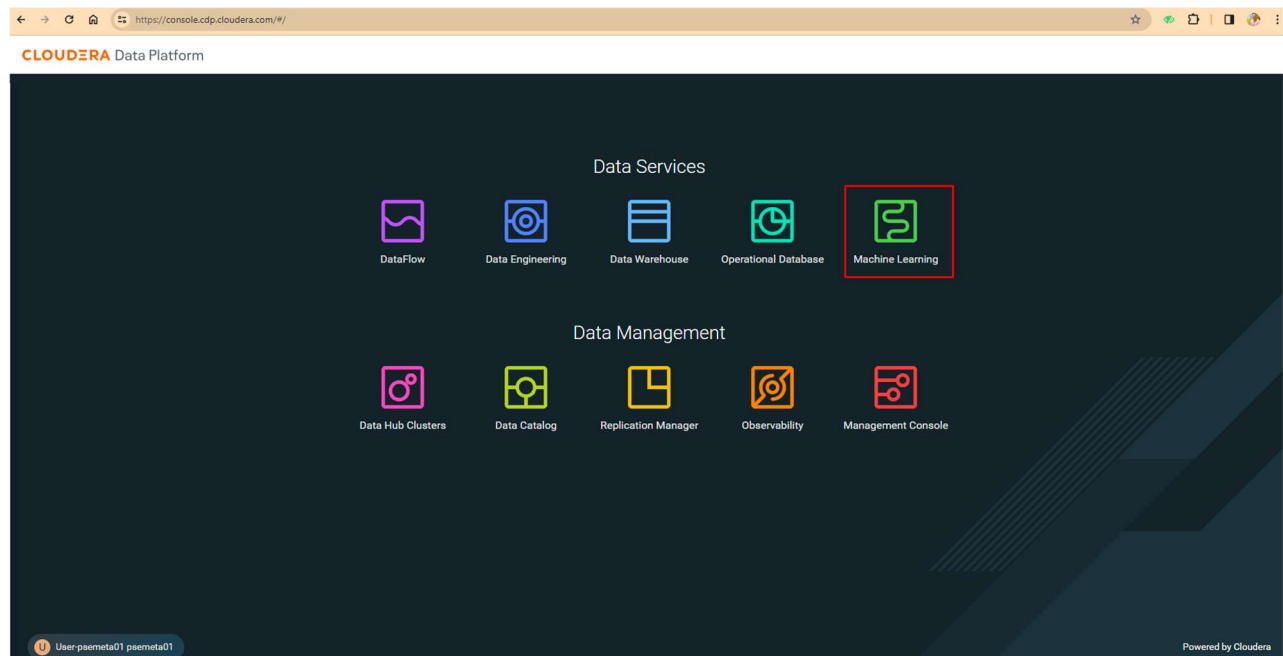
Data Lifecycle on CDP Public Cloud

Machine Learning Lab

Goals:

- Train a model to predict if a customer will churn.
- Deploy/expose model as REST API.

1. Click on Data Warehouse from CDP PC Home.




2. This is a screen to select a Workspace, which is compute resource allocation for Data Science related jobs. Click on the only Workspace that appears.

← → ↻ 🏠

https://console.us-west-1.cdp.cloudera.com/ml/#/

☆ 🌐 📄 🖨️ 👤 ⋮

 **CLOUDERA**
Machine Learning

Workspaces

Workspace Backups

Model Registries

Help

User-psemeta01 psemeta01

Machine Learning Workspaces


Environment

All

🔄

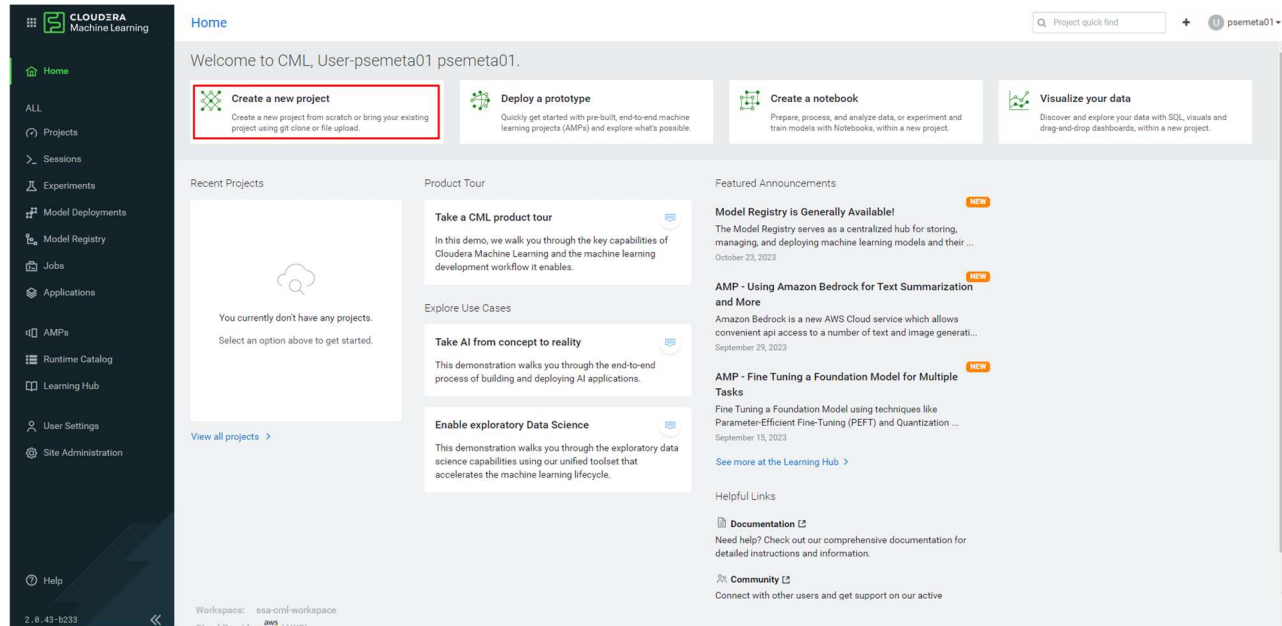
🔗

Provision Workspace

Status	Version	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	2.0.43	ssa-cml-workspace	psemeta-cdp-env	us-west-2	03/13/2024 9:51 AM +04	 AWS	<div>⋮</div>

Displaying 1 - 1 of 1 < 1 > 25 / page

3. Once in the Workspace, you should see the following interface. Here are the projects you have created. It is time to create a new project. Click on **Create a new project**.



4. Enter the following information to create a new project:

Project Name: <username>-Telco Churn (Example: **psemeta01-Telco Churn**)

Project Visibility: *Private*

Initial Setup, select **Git**

In the text field below HTTPS, enter the url of the git repo:

<https://github.com/DashDipti/TelcoChurn>

Keep the rest of the settings the same. Click the button **Create Project**.

CLUSTERA Machine Learning

New Project

Project Name: psemeta01-Telco Churn

Project Description:

Project Visibility: ☒ Private - Only added collaborators can view the project. ☐ Public - All authenticated users can view this project.

Initial Setup:

Provide the Git URL of the project to clone. Select the option that applies to your URL access.

☒ HTTPS ☐ SSH

https://github.com/camposalex/TelcoChurn

You are able to provide username/password.
e.g. https://username:password@mygithub.com/my/repository

Cancel Create Project

Click on **Advanced Options**.

Select -

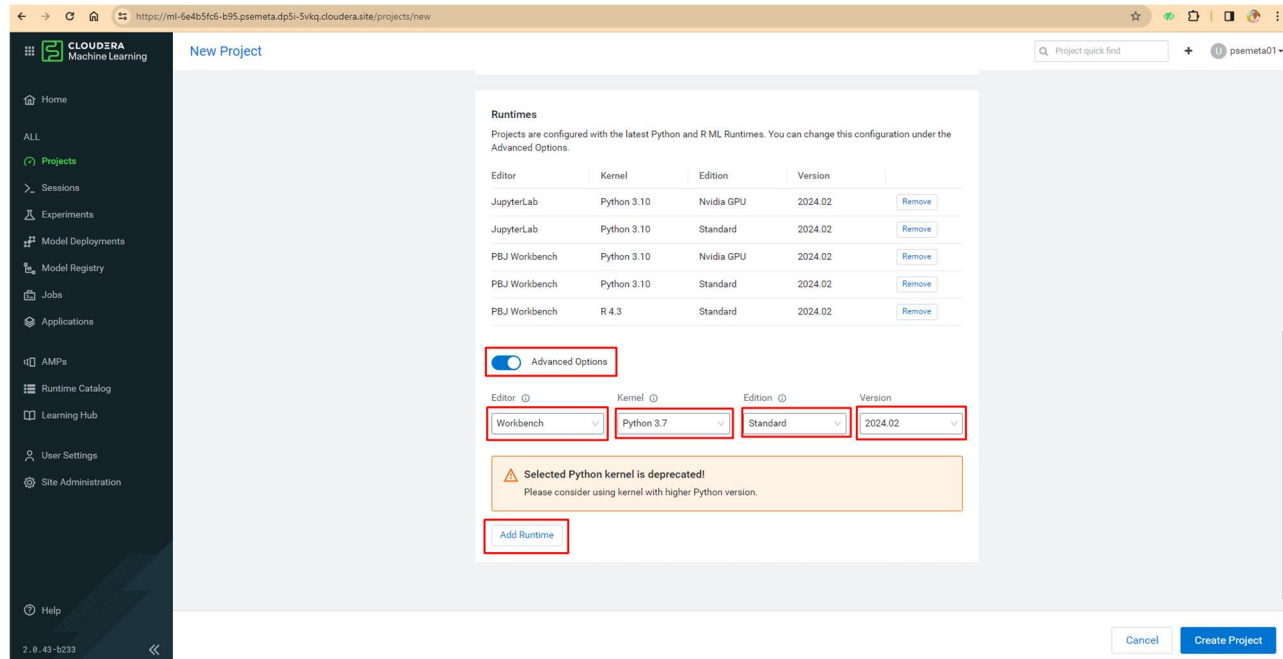
Editor – **Workbench**

Kernel – **Python 3.7**

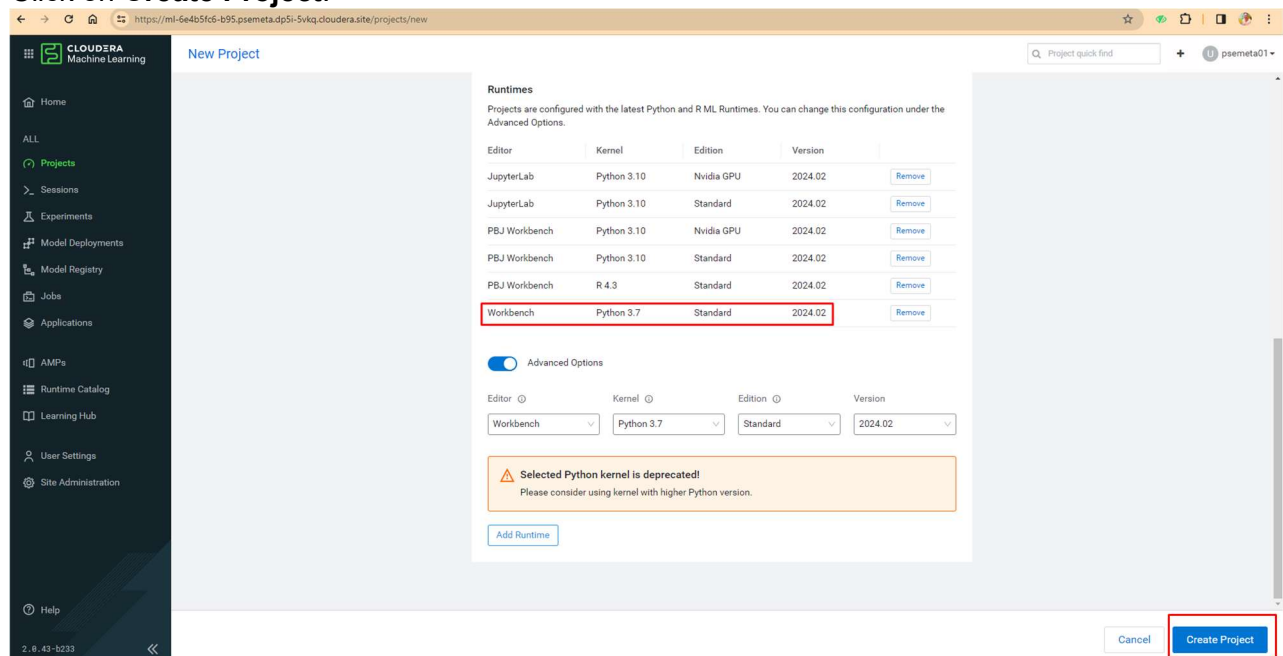
Edition – **Standard**

Version – **2024.02**

Click on **Add Runtime**.



The Runtime gets added as you can see below.
Click on **Create Project**.



5. Once the project is created, you should see the following screen:

Models, deploy and manage models as REST APIs to serve predictions.
Jobs, automate and orchestrate the execution of batch analytics workloads
Files, assets that are part of the project, such as files, scripts and code.

The screenshot shows the Cloudera Machine Learning interface for a project named 'psemeta01-Telco Churn'. The left sidebar contains navigation links for Home, All Projects, PROJECT (Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, Files, Collaborators, Project Settings, AMPs, Runtime Catalog, Learning Hub, User Settings, Site Administration, Help), and version information (2.0.43-b233). The main content area has a search bar and a 'New Session' button. Below this, there are sections for Models, Jobs, and Files. The Files section contains a table with columns for Name, Size, and Last Modified, listing various files like flask, images, models, and scripts. The 'New Session' button is highlighted in the top right corner.

Name	Size	Last Modified
flask	-	a few seconds ago
images	-	a few seconds ago
models	-	a few seconds ago
0_bootstrap.py	1.95 kiB	a few seconds ago
1_trainStrategy_job.py	17.80 kiB	a few seconds ago
2_get_champion.py	508 B	a few seconds ago
_best_model_serve.py	2.74 kiB	a few seconds ago
_model_viz.py	4.21 kiB	a few seconds ago
cdsw-build.sh	44 B	a few seconds ago
churnexplainer.py	6.69 kiB	a few seconds ago
lineage.yml	610 B	a few seconds ago
README.md	11.97 kiB	a few seconds ago
requirements.txt	198 B	a few seconds ago
utilsafe.json	781.07 kiB	a few seconds ago

This Telco Churn project consists of running three scripts. The way of execution is through a session, which is the allocation of isolated compute resources for each user. For this, you must click on the blue button **New Session**, located in the upper right.

This screenshot is similar to the one above, showing the same project page. The 'New Session' button in the top right corner is highlighted with a red rectangular box.

6. Make sure to select the values as shown in the screenshot. We select the previously added Runtime.

Slide the **Enable Spark** option and then select **Spark 3.2.3-CDE.xxxxx**.

Note: Spark 3.x is needed for Iceberg related tables and will give error if not chosen.

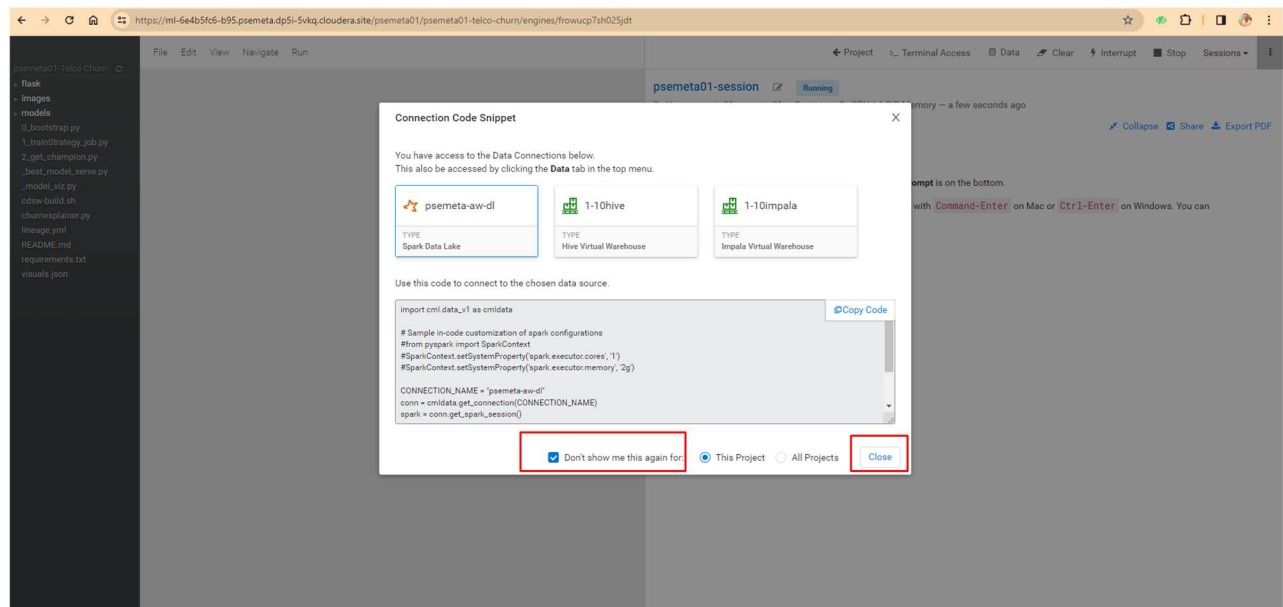
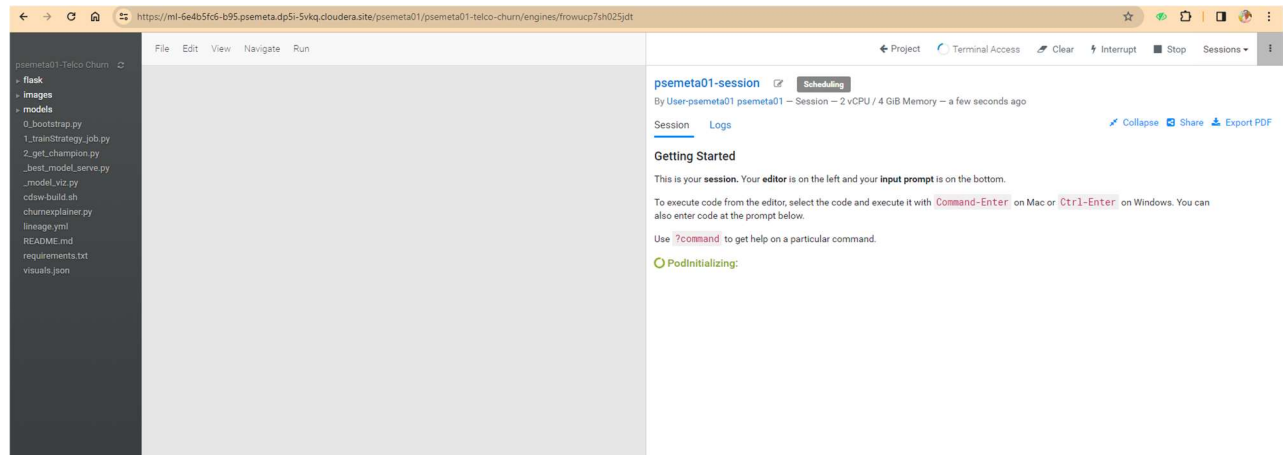
Click on **Start Session**.

The screenshot shows the 'Start A New Session' form in the Cloudera Machine Learning interface. The form is titled 'Start A New Session' and is located at the URL <https://ml-6e4b5fc6-b95.psemeta.dp5i-5vkq.cloudera.site/psemeta01/psemeta01-telco-churn/sessions/new>. The form includes the following fields and options:

- Session Name:** A text input field containing 'psemeta01-session'.
- Runtime:** A table with columns: Editor, Kernel, Edition, and Version.

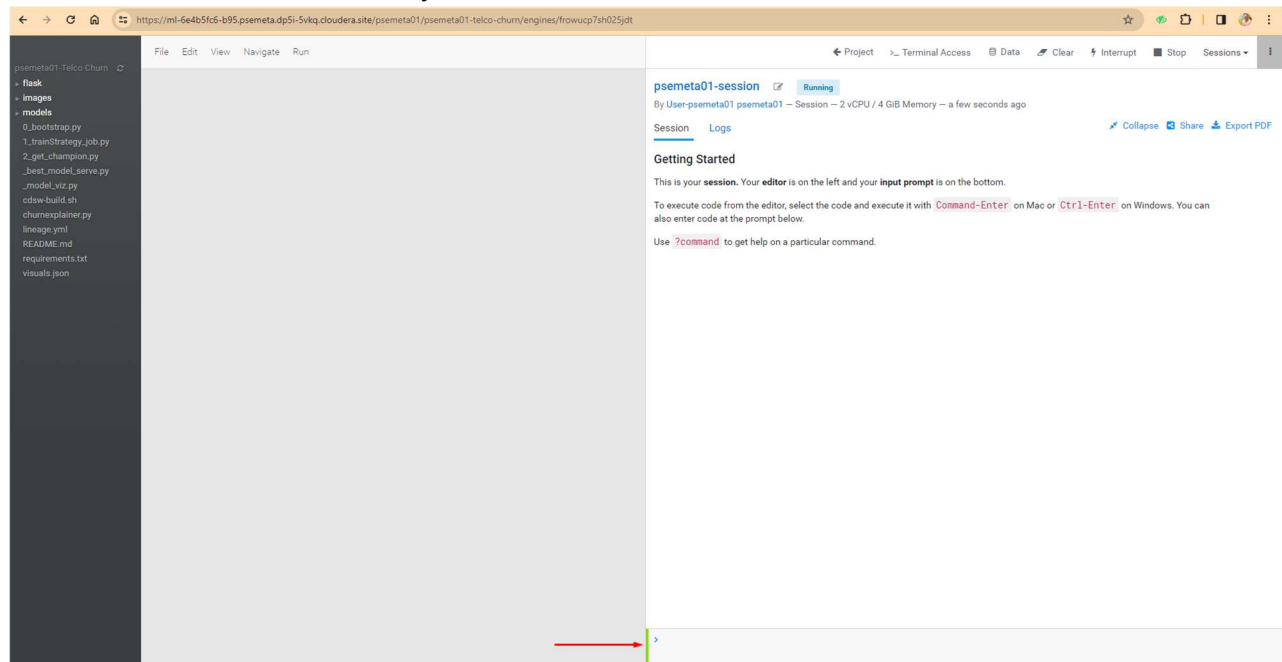
Editor	Kernel	Edition	Version
Workbench	Python 3.7	Standard	2024.02
- Warning:** A yellow box with a warning icon stating 'Selected Python kernel is deprecated! Please consider using kernel with higher Python version.'
- Enable Spark:** A toggle switch that is currently turned on.
- Spark Version:** A dropdown menu showing 'Spark 3.2.3 - CDE 1.19.2 - HOTFIX-2'.
- Runtime Image:** A text input field containing 'docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2024.02.1-b4'.
- Resource Profile:** A dropdown menu showing '2 vCPU / 4 GiB Memory'.
- Buttons:** 'Cancel' and 'Start Session' buttons at the bottom right.

7. When you start a session for the first time, it will ask if you want to use a data connection. This project does not need this type of connection. Mark the check of **Don't show me this again for**, and then click the button **Close**, so this window will not appear anymore.




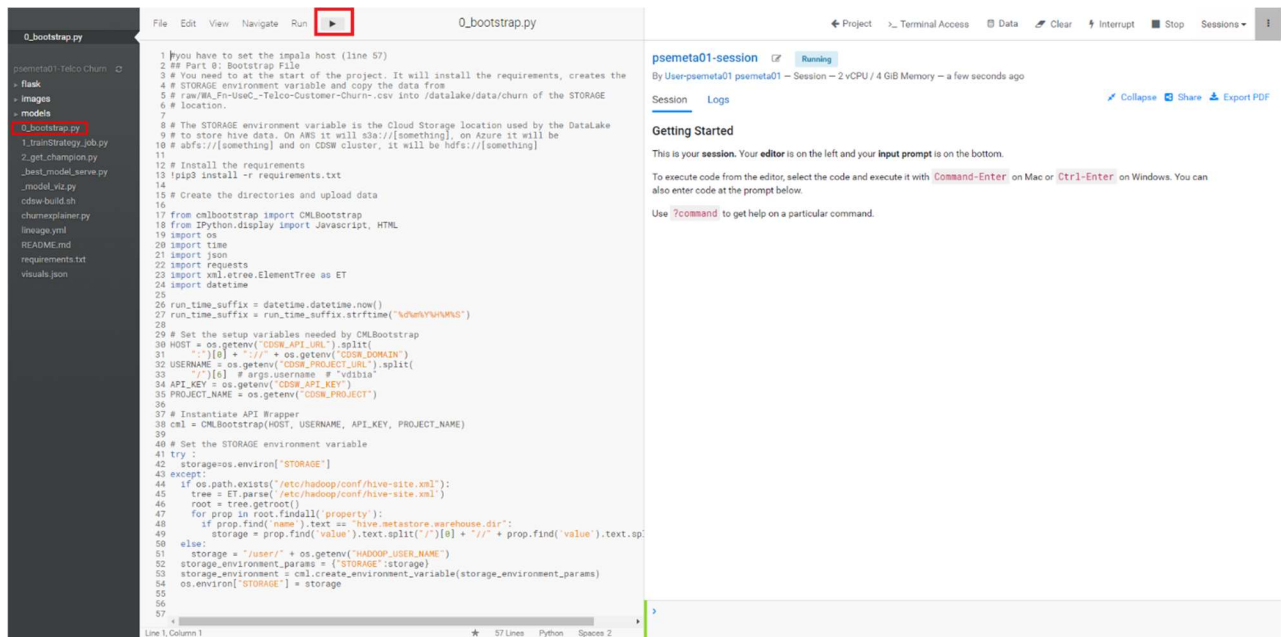
8. The editor/notebook located on the right side of the window will be in **Scheduling** status, and the bottom command bar flashing red. This means that CML is allocating computation for your session.

After a few seconds, the status changes to **Running**, and the command bar to green. This means that the session is ready to run code.

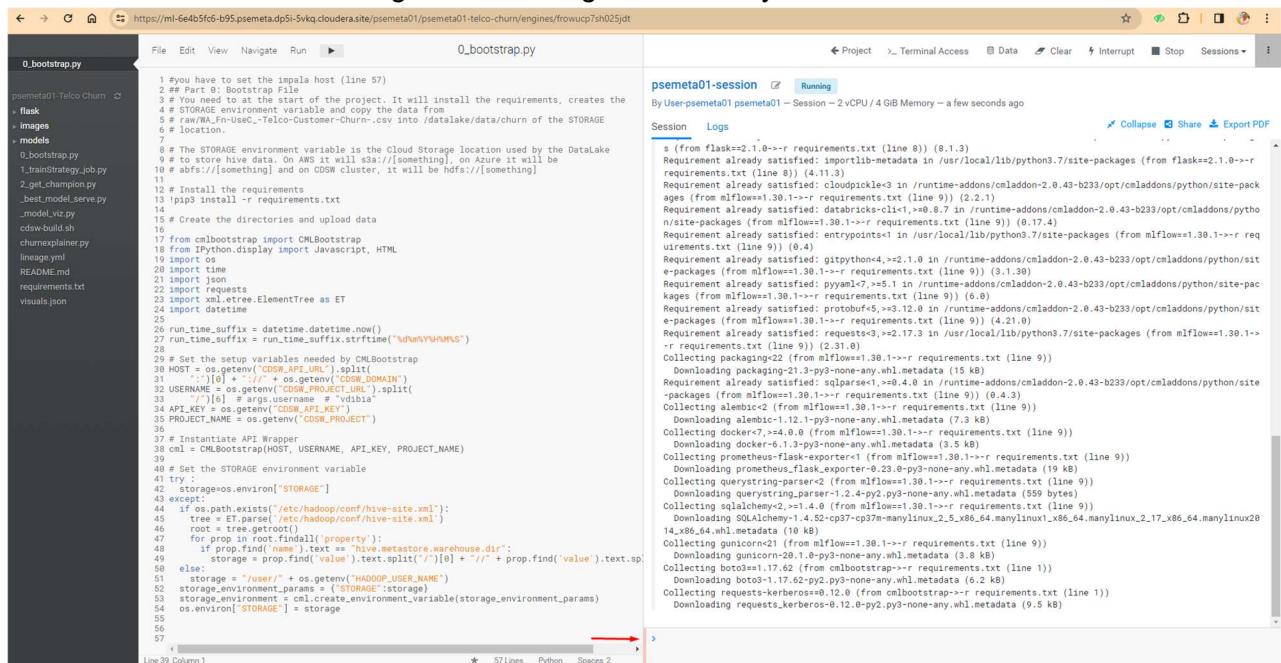


9. The first script/code to run is **0_bootstrap.py**. This Python code configures the libraries required for the project and integration with Lakehouse tables you populated before. Select (just

one click) the file in the bar located on the left side of the interface, this will make the code appear in the editor. Once the file is selected, click on the button  to run the code.

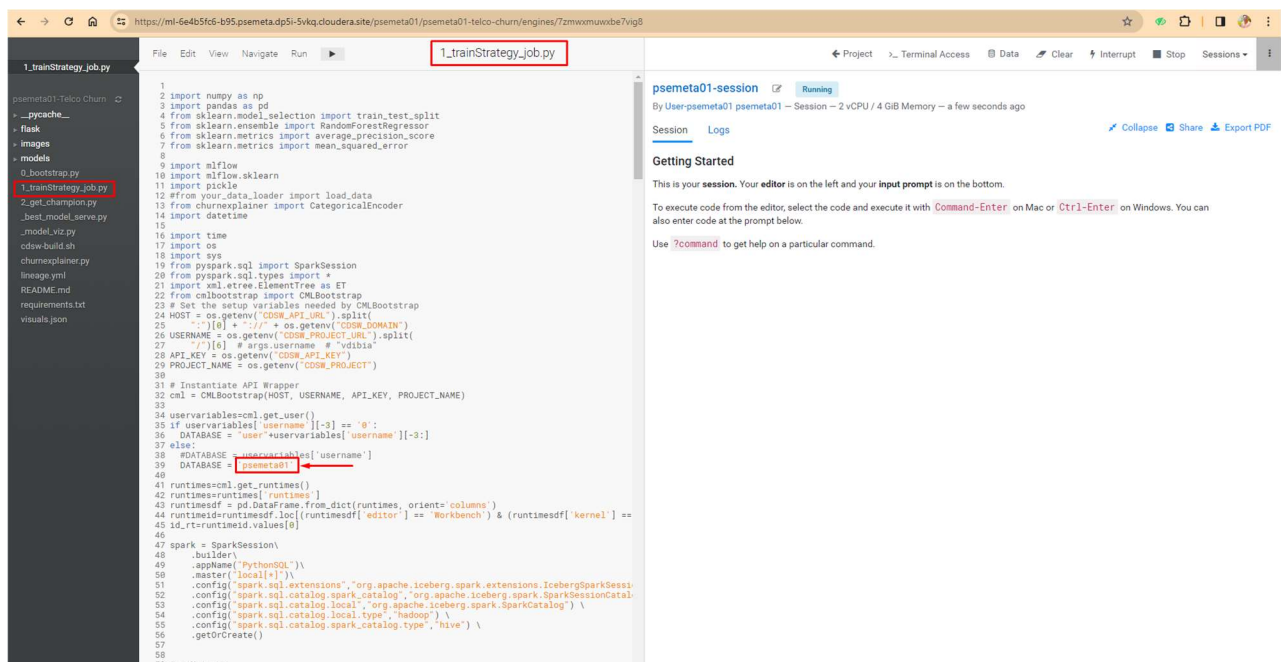


When you start execution, you will see code output on the right side of the interface, and the bottom command bar flashing red, indicating that it is busy.




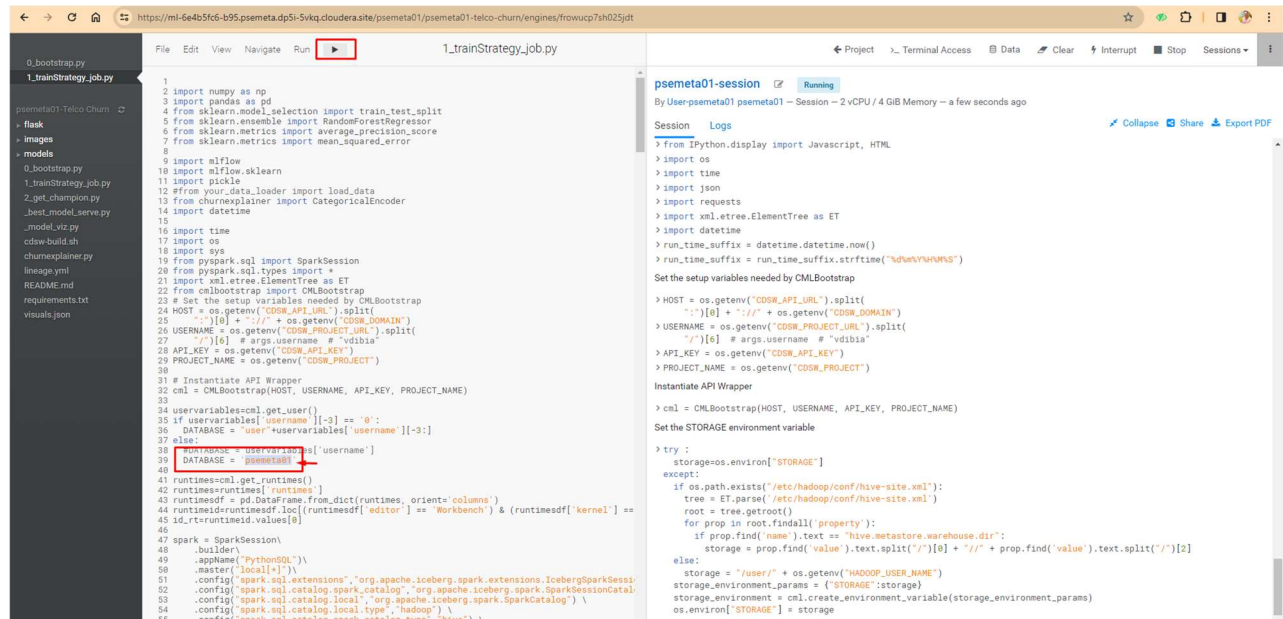
[illegible]

In the code change the line that has the value of **DATABASE** to your username ex – **psemeta01**



The screenshot below shows the changed name for the database.

Click on the button  to run the code. Once the execution is finished (approximately 1-2 minutes).



```
0_bootstrap.py
1_trainStrategy_job.py
psemeta01-Telco Churn
+ flask
+ images
+ models
0_bootstrap.py
1_trainStrategy_job.py
2_get_champion.py
3_best_model_serve.py
_model_viz.py
cdw-build.sh
churnexplainer.py
lineage.yml
README.md
requirements.txt
visuals.json
```

```
1
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.ensemble import RandomForestRegressor
6 from sklearn.metrics import average_precision_score
7 from sklearn.metrics import mean_squared_error
8
9 import mlflow
10 import mlflow.sklearn
11 import pickle
12 #from your_data_loader import load_data
13 from churnexplainer import CategoricalEncoder
14 import datetime
15
16 import time
17 import os
18 import sys
19 from pyspark.sql import SparkSession
20 from pyspark.sql.types import *
21 import xml.etree.ElementTree as ET
22 from cmlbootstrap import CMLBootstrap
23 # Set the setup variables needed by CMLBootstrap
24 HOST = os.getenv("CDGW_API_URL").split(
25     "://")[0] + "://" + os.getenv("CDGW_DOMAIN")
26 USERNAME = os.getenv("CDGW_PROJECT_URL").split(
27     "://")[0] # args.username # "vdbia"
28 API_KEY = os.getenv("CDGW_API_KEY")
29 PROJECT_NAME = os.getenv("CDGW_PROJECT")
30
31 # Instantiate API Wrapper
32 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
33
34 user_variables=cml.get_user()
35 if user_variables['username'][-3] == '0':
36     DATABASE = 'user'+user_variables['username'][-3:]
37 else:
38     DATABASE = user_variables['username']
39     DATABASE = 'psemeta01'
40
41 runtimes=cml.get_runtimes()
42 runtimes=runtimes['runtimes']
43 runtimesdf = pd.DataFrame.from_dict(runtimes, orient='columns')
44 runtimeid=runtimesdf.loc[(runtimesdf['editor'] == 'Workbench') & (runtimesdf['kernel'] ==
45     id_rruntimeid.values[0])
46
47 spark = SparkSession
48     .builder()
49     .appName("PythonSQL")\
50     .master("local[*]")\
51     .config("spark.sql.extensions", "org.apache.iceberg.spark.extensions.IcebergSparkSessi
52     .config("spark.sql.catalog.spark_catalog", "org.apache.iceberg.spark.SparkSessionCatal
53     .config("spark.sql.catalog.local", "org.apache.iceberg.spark.SparkCatalog") \
54     .config("spark.sql.catalog.local.type", "hadoop") \
55     .config("spark.sql.catalog.spark_catalog.type", "hive") \
```

psemeta01-session Running

By User:psemeta01 psemeta01 - Session - 2 vCPU / 4 GiB Memory - a few seconds ago

Session Logs [Collapse](#) [Share](#) [Export PDF](#)

```
> from IPython.display import Javascript, HTML
> import os
> import time
> import json
> import requests
> import xml.etree.ElementTree as ET
> import datetime
> run_time_suffix = datetime.datetime.now()
> run_time_suffix = run_time_suffix.strftime("%d-%m-%Y-%H-%M-%S")

Set the setup variables needed by CMLBootstrap

> HOST = os.getenv("CDGW_API_URL").split(
    "://")[0] + "://" + os.getenv("CDGW_DOMAIN")
> USERNAME = os.getenv("CDGW_PROJECT_URL").split(
    "://")[0] # args.username # "vdbia"
> API_KEY = os.getenv("CDGW_API_KEY")
> PROJECT_NAME = os.getenv("CDGW_PROJECT")

Instantiate API Wrapper

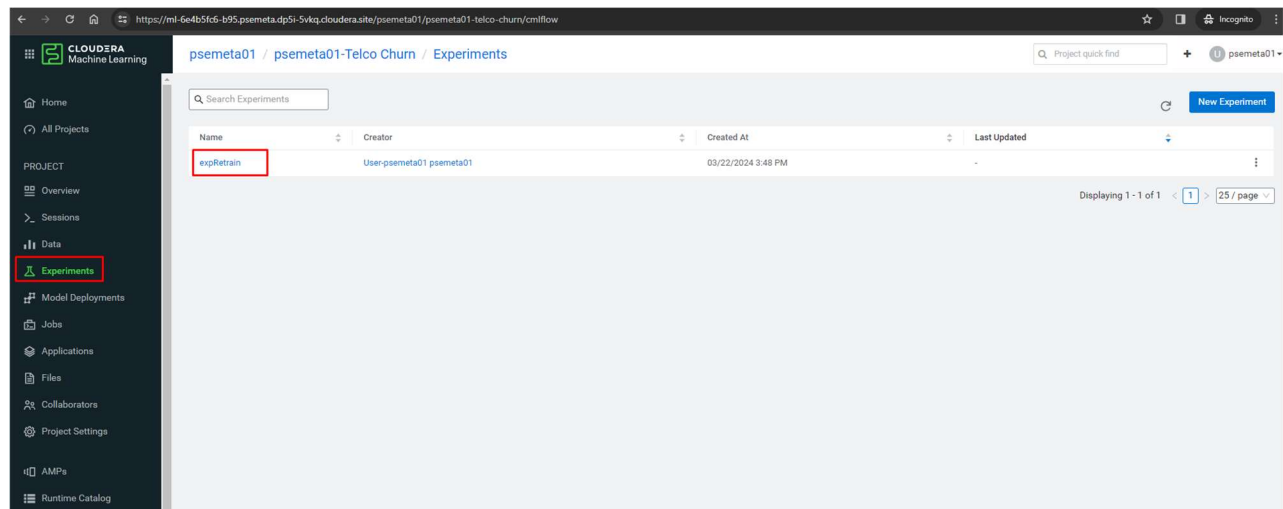
> cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)

Set the STORAGE environment variable

> try :
    storage=os.environ["STORAGE"]
except:
    if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
        tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
        root = tree.getroot()
        for prop in root.findall('property'):
            if prop.find('name').text == "hive.metastore.warehouse.dir":
                storage = prop.find('value').text.split("/")[-1] + "/" + prop.find('value').text.split("/")[-2]
    else:
        storage = "/user/" + os.getenv("HADOOP_USER_NAME")
        storage_environment_params = ("STORAGE":storage)
        storage_environment = cml.create_environment_variable(storage_environment_params)
        os.environ["STORAGE"] = storage
```


Once it runs successfully as indicated by green bar below click on the button **Project**, located in the upper right bar of the session to go back to the project home.

11. Once back in project home, click on the **Experiments** option, from the left menu, and then on **expRetrain** in the list of Experiments that appears.



12. On this screen you will see the three runs of this experiment. Look at the last column, where **precision** attribute displays. This is the precision that each hyper parameter is delivering.

The screenshot shows the Cloudera Machine Learning interface. The left sidebar contains navigation options: Home, All Projects, PROJECT (Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, Files, Collaborators, Project Settings, AMPs, Runtime Catalog, Learning Hub, User Settings). The main panel displays the 'Experiment' details for 'expRetrain' (ID: vaid-ln4b-9etu-mfgg). Below the experiment details, there are 'Runs (3)' listed with a search bar 'metrics.mae < 1 and p...'. A table shows the runs with columns: Status, Start Time, Run Name, Duration, User, Source, Version, Models, Parameters (algo, compute, dataset), Metrics (precision), and Tags (engineID). The 'precision' column is highlighted with a red box, showing values 0.8518098598... and 0.8669739750....


Status	Start Time	Run Name	Duration	User	Source	Version	Models	Parameters	Metrics	Tags
Success	2024-03-22 03:48:57	run_4856_0	6.6s	psemeta01	ipython3	49930d	sklearn	random forest local telco-churn	0.8518098598...	os2m170aexjh...
Success	2024-03-22 03:49:04	run_4856_1	4.1s	psemeta01	ipython3	49930d	sklearn	random forest local telco-churn	0.8732466313...	os2m170aexjh...
Success	2024-03-22 03:49:08	run_4856_2	4.5s	psemeta01	ipython3	49930d	sklearn	random forest local telco-churn	0.8669739750...	os2m170aexjh...

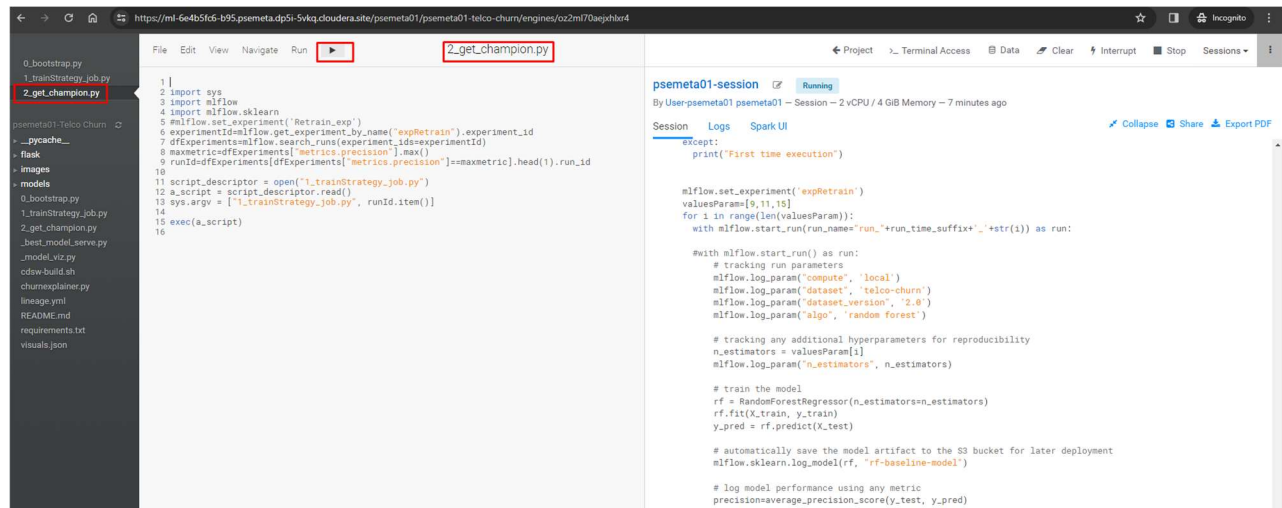
13. Let's go back to the session to run the last code. Since sessions run in Kubernetes containers, it's very easy to get back to where we were. Click on the option **Sessions** from the left menu, and later the only session that will appear in the list.

If you didn't name your session when you started it should be called *Untitled Session*.

The screenshot shows the Cloudera Machine Learning interface with the 'Sessions' tab selected in the left sidebar. The main panel displays the 'Sessions' page for 'psemeta99' (Telco Churn). It includes a 'Creator' dropdown set to 'All' and a 'Show Running Only' toggle. A table lists the sessions with columns: Status, Session, Kernel, Creator, Created At, and Duration. A single session is listed: 'psemeta99-session' (Python 3.7 Workbench Standard) created by 'User-psemeta99 psemeta99' on 09/06/2024 12:18 PM, running since 7m 30s. The session name is highlighted with a red box.

Status	Session	Kernel	Creator	Created At	Duration
Running	psemeta99-session	(Python 3.7 Workbench Standard)	User-psemeta99 psemeta99	09/06/2024 12:18 PM	Running since 7m 30s

14. The third and last script/code to run is **2_get_champion.py**. This Python code takes the hyper parameter of the execution of the Experiment with the better precision and deploys two Models as REST API, one to be integrated in Data Visualization and another for unit use for calls. Select (just one click) the file in the bar located on the left side of the interface, this will make the code appear in the editor. Once the file is selected, click on the button  to run the code.



The screenshot displays the AWS SageMaker console interface. On the left, the file explorer shows a directory structure with files like `0_bootstrap.py`, `1_trainStrategy_job.py`, and `2_get_champion.py`, which is currently selected. The central pane shows the Python code for `2_get_champion.py`, which imports `sys`, `mlflow`, and `sklearn`, and contains logic for setting up an experiment, training a model, and saving it. The right pane shows the execution console for a session named `psemeta01-session`, which is in a `Running` state. The console output shows the start of the execution with `print('First time execution')`.

After a few seconds, you will see the following message “**Deploying ModelViz...**” repeated several times, and the bottom command bar will be red initially and then will become green on successful deployment.

```
1 import sys
2 import mlflow
3 import mlflow.sklearn
4
5 mlflow.set_experiment('Retrain_exp')
6 experimentId=mlflow.get_experiment_by_name('expRetrain').experiment_id
7 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
8 maxmetric=dfExperiments['metrics.precision'].max()
9 runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
10
11 script_descriptor = open('1_trainStrategy_job.py')
12 a_script = script_descriptor.read()
13 sys.argv = ['1_trainStrategy_job.py', runId.item()]
14
15 exec(a_script)
16
```

2024/03/22 11:48:56 INFO mlflow.tracking.fluent: Experiment with name 'expRetrain' does not exist. Creating a new experiment.

First time execution

> import sys

> import mlflow

> import mlflow.sklearn

mlflow.set_experiment('Retrain_exp')

> experimentId=mlflow.get_experiment_by_name('expRetrain').experiment_id

> dfExperiments=mlflow.search_runs(experiment_ids=experimentId)

> maxmetric=dfExperiments['metrics.precision'].max()

> runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id

> script_descriptor = open('1_trainStrategy_job.py')

> a_script = script_descriptor.read()

> sys.argv = ['1_trainStrategy_job.py', runId.item()]

/usr/local/bin/python3:1: FutureWarning: 'item' has been deprecated and will be removed in a future version

#!/usr/local/bin/python3.7

> exec(a_script)

Creating new model for visualization

Workspace URL: https://modelservice.ml-6e4b5fc6-b95.psemeta.dp5i-5vkq.cloudiera.site/model

ModelViz Access Key: m0b3rkbwz2c56kg97ucyzge55t29b

Deploying ModelViz.....

```
1 import sys
2 import mlflow
3 import mlflow.sklearn
4
5 mlflow.set_experiment('Retrain_exp')
6 experimentId=mlflow.get_experiment_by_name('expRetrain').experiment_id
7 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
8 maxmetric=dfExperiments['metrics.precision'].max()
9 runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
10
11 script_descriptor = open('1_trainStrategy_job.py')
12 a_script = script_descriptor.read()
13 sys.argv = ['1_trainStrategy_job.py', runId.item()]
14
15 exec(a_script)
16
```

test

By User-psemeta01 psemeta01 - Session - 2 vCPU / 4 GB Memory - a few seconds ago

Session Logs Spark UI

> maxmetric=dfExperiments['metrics.precision'].max()

> runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id

> script_descriptor = open('1_trainStrategy_job.py')

> a_script = script_descriptor.read()

> sys.argv = ['1_trainStrategy_job.py', runId.item()]

/usr/local/bin/python3:1: FutureWarning: 'item' has been deprecated and will be removed in a future version

#!/usr/local/bin/python3.7

> exec(a_script)

/home/cdsw/.local/lib/python3.7/site-packages/sklearn/ensemble/base.py:158: DeprecationWarning: 'np.int' is a deprecated alias for the builtin 'int'. To silence this warning, use 'int' by itself. Doing this will not modify any behavior and is safe. When replacing 'np.int', you may wish to use e.g. 'np.int64' or 'np.int32' to specify the precision. If you wish to review your current use, check the release note link for additional information. Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

dtype=np.int)

Creating new model for visualization

Workspace URL: https://modelservice.ml-6e4b5fc6-b95.psemeta.dp5i-5vkq.cloudiera.site/model

ModelViz Access Key: m1qn6bxiwont284ezovfcsrt6p6g8b

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

Deploying ModelViz.....

ModelViz is deployed

Creating new model

ModelOps Access Key: mrfdj9ry8cidswoe28jyna9vc7lk7aw7

Deploying ModelOps.....

Deploying ModelOps.....

Deploying ModelOps.....

Deploying ModelOps.....

Model is deployed

After about 2 minutes, the last message should be "Model is deployed", and the bar will be green. It means that the Deployment of the two Models is complete. Click on the button **Project**, located in the upper right bar of the session to return to the home page of the project.

0_bootstrap.py

Telco Churn

0_bootstrap.py

0_bootstrap.py

1_trainStrategy_job.py

2_get_champion.py

+ _pycache_

..best_model_serve.py

..model_viz.py

cdsw-build.sh

chumexplainer.py

+ flask

+ images

lineage.yml

+ models

README.md

requirements.txt

visuals.json

File Edit View Navigate Run ▶

2_get_champion.py

1
2 import sys
3 import mlflow
4 import mlflow.sklearn
5 #mlflow.set_experiment('Retrain_exp')
6 experimentId=mlflow.get_experiment_by_name('expRetrain').experiment_id
7 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
8 maxmetric=dfExperiments['metrics.precision'].max()
9 runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
10
11 script_descriptor = open("1_trainStrategy_job.py")
12 a_script = script_descriptor.read()
13 sys.argv = ["1_trainStrategy_job.py", runId.item()]
14
15 exec(a_script)
16

Line 1, Column 1

★ 16 Lines Python Spaces 2

Project Terminal Access Data Clear Interrupt Stop Sessions

Untitled Session Running
By Test10 User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago

Session Logs Spark UI Collapse Share Export PDF

> import sys
> import mlflow
> import mlflow.sklearn

mlflow.set_experiment('Retrain_exp')

> experimentId=mlflow.get_experiment_by_name('expRetrain').experiment_id
> dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
> maxmetric=dfExperiments['metrics.precision'].max()
> runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
> script_descriptor = open("1_trainStrategy_job.py")
> a_script = script_descriptor.read()
> sys.argv = ["1_trainStrategy_job.py", runId.item()]

/usr/local/bin/ipython3:1: FutureWarning: 'item' has been deprecated and will be removed in a future version
#! /usr/local/bin/python3.7

> exec(a_script)

Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lt4ul28lkvb7mbdph9gi
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Model is deployed
Creating new model for visualization
New model created with access key mli7u8em8ypcxly6xid1c4a8g17q3foi
Deploying Model.....
Deploying Model.....
Deploying Model.....
Model is deployed

15. Once on the home page of the project, you will see the Models displayed, which are two. Click on the one that starts with **ModelViz_**.

The screenshot shows the Databricks project home page. On the left is a dark sidebar with navigation options: All Projects, PROJECT (Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, Files, Collaborators, Project Settings, AMPs, Runtime Catalog, Learning Hub, User Settings, Site Administration, Help), and a version indicator 2.8.43-b233. The main content area is titled 'Models' and contains a table with two rows:

Model Deployment	Source	Status	Replicas	CPU	Memory	Last Deployed	Actions
ModelOpsChum_psemeta01	_bestL...	Deployed	1 / 1	1	2.00 GiB	Mar 22, 2024, 04:28 PM	Stop
ModelViz_psemeta01	_model...	Deployed	1 / 1	1	2.00 GiB	Mar 22, 2024, 04:28 PM	Stop

Below the table, there is a 'Jobs' section stating 'This project has no jobs yet. Create a new job to document your analytics pipelines.' and a 'Files' section with a table of files:

Name	Size	Last Modified
__pycache__	-	7 minutes ago
flask	-	13 minutes ago
images	-	13 minutes ago
models	-	13 minutes ago
0_bootstrap.py	1.95 kiB	13 minutes ago
1_trainStrategy_job.py	17.80 kiB	8 minutes ago
2_get_champion.py	508 B	13 minutes ago
_best_model_serve.py	2.74 kiB	13 minutes ago
_model_viz.py	4.21 kiB	13 minutes ago
cdsw-build.sh	44 B	13 minutes ago

16. Here you will see Model information and settings in the Overview tab.

The screenshot shows the 'ModelViz_psemeta01' Overview tab. The left sidebar is the same as in the previous screenshot, with 'Model Deployments' highlighted. The main content area has tabs for Overview, Deployments, Builds, Monitoring, Logs, and Settings. The 'Overview' tab is active, showing a description of the model as a 'visualization a given model prediction'. Below this is a 'Sample Code' section with a code editor showing a curl command for a POST request to a model service. To the right of the code editor is a 'Model Details' section with a table of model information:

Source	Code
Model Id	5
Model CRN	cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/4b5f5d45-a746-458f-ab03-20505ace5ee9
Deployment Id	6
Deployment CRN	cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/3be502e6-3807-43ac-b343-82c4deed21ab
Build Id	5
Build CRN	cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/e181b700-736d-41af-9a11-dbb98afdd62
Deployed By	psemeta01
Comment	Initial revision.
Runtime Image	Python 3.7 (Standard)
File	_model_viz.py
Function	predict

Below the code editor is a 'Sample Response' section showing an empty JSON object {}. At the bottom, it indicates the workspace is 'ssa-cml-workspace' and the cloud provider is 'AWS'.

The screenshot shows the Dapr CLI interface. On the left is a dark sidebar with navigation links: Home, All Projects, PROJECT, Overview, Sessions, Data, Experiments, Model Deployments (highlighted in green), Jobs, Applications, Files, Collaborators, Project Settings, AMPs, Runtime Catalog, Learning Hub, User Settings, Site Administration, and Help. The main area displays the output of the 'test' command. At the top, the 'Input' is shown as a JSON object: `{ "input": "No" }`. Below it, the 'Result' section shows a 'Status' of 'success' (indicated by a green dot icon) and a 'Response' containing a JSON object: `{ "data": { "colnames": ["result"], "coltypes": ["INT"], "rows": [[0], [0]] } }`. At the bottom, the 'Replica ID' is 'modelviz-osemeta01-5-6-79885c4685-f9znv' and the 'Workspace' is 'ssa-cnd/workspace'.

```
url - https://modelservice.ml-6e4b5fc6-b95.psemeta.dp5i-5vkq.cloudera.site/model  
accessKey - miqnv6bxiwontz84ozoxfyc5rt6p6grb
```

[illegible]