



LAB

04 – Cloudera Machine Learning

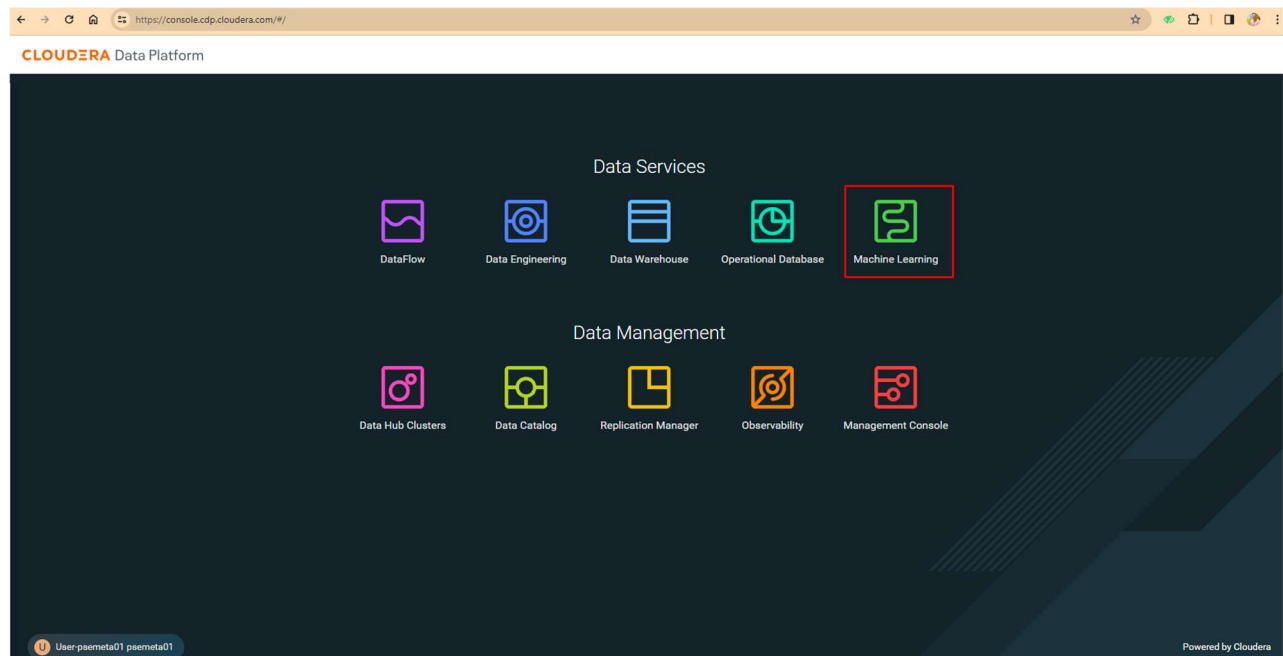
Data Lifecycle on CDP Public Cloud

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 ↕

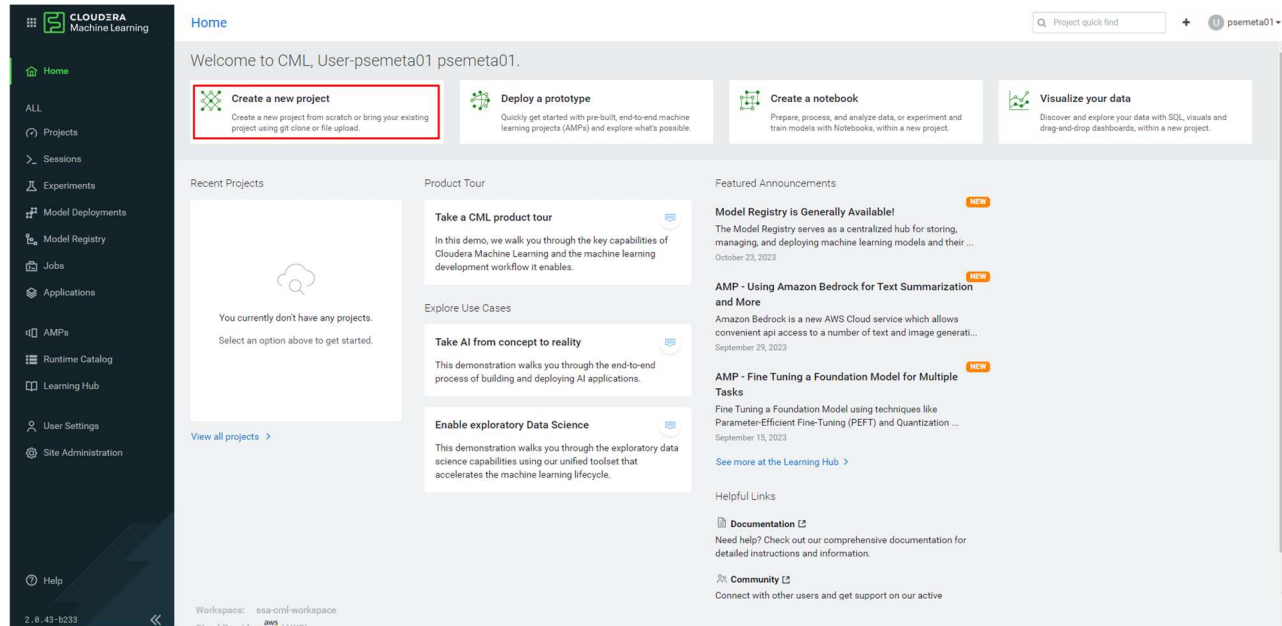
🔄 📄

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	⋮

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**.

The screenshot shows the 'New Project' form in the Cloudera Machine Learning interface. The form is titled 'New Project' and is located at the URL <https://ml-6e4b5f6-b95.psemeta.dp5i-5v1qz.cloudera.site/projects/new>. The form includes the following fields and options:

- Project Name:** A text field containing 'psemeta01-Telco Churn'.
- Project Description:** A text area for describing the project.
- Project Visibility:** Two radio buttons: 'Private - Only added collaborators can view the project.' (selected) and 'Public - All authenticated users can view this project.'
- Initial Setup:** Four buttons: 'Blank', 'Template', 'AMPs', and 'Git' (selected).
- Git URL:** A text field containing 'https://github.com/camposalex/TelcoChurn'. Below this field, there is a note: 'You are able to provide username/password. e.g. https://username:password@mygithub.com/my/repository'.
- HTTPS/SSH:** Two radio buttons: 'HTTPS' (selected) and 'SSH'.

At the bottom right of the form, there are two buttons: 'Cancel' and 'Create Project'.

Click on **Advanced Options**.

Select -

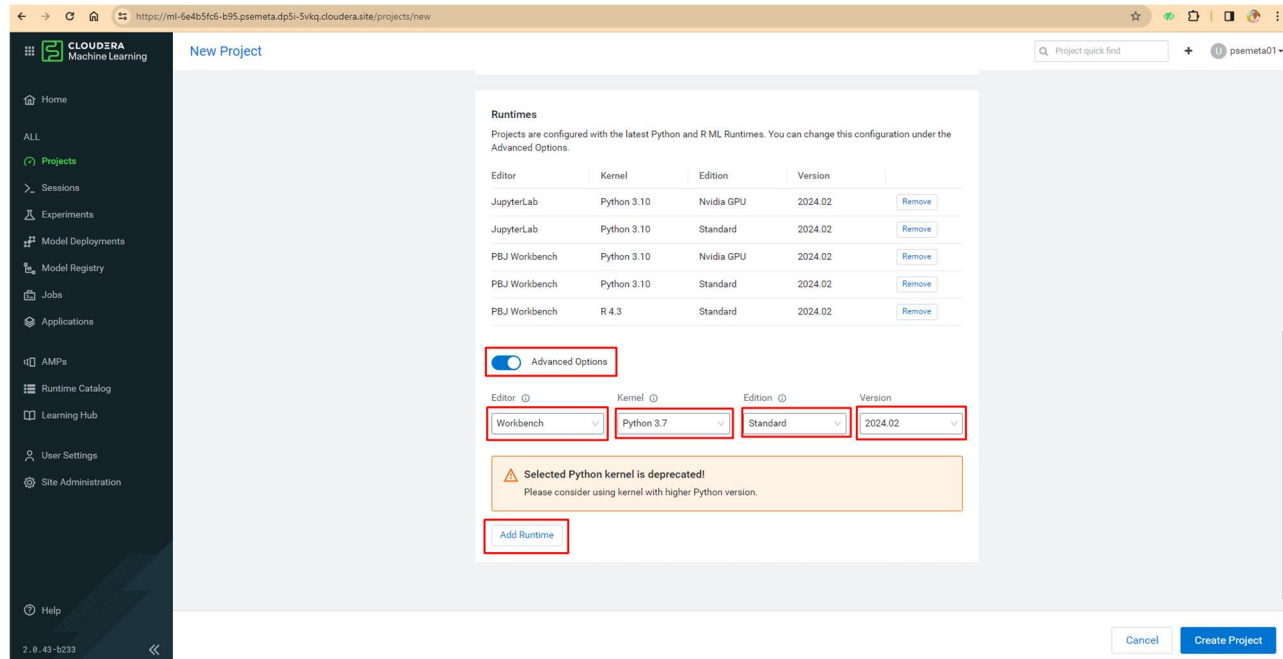
Editor – **Workbench**

Kernel – **Python 3.7**

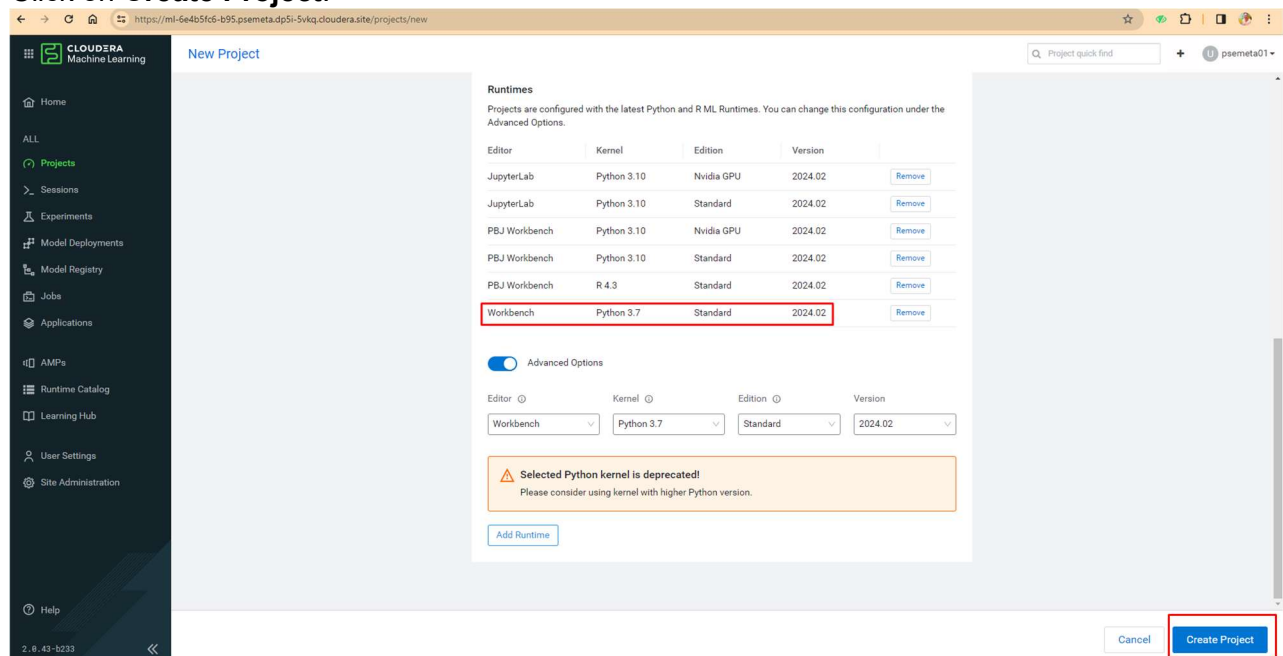
Edition – **Standard**

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.

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, and models. The 'New Session' button is highlighted with a red box.

Name	Size	Last Modified
flask	-	2 minutes ago
images	-	2 minutes ago
models	-	2 minutes ago

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.

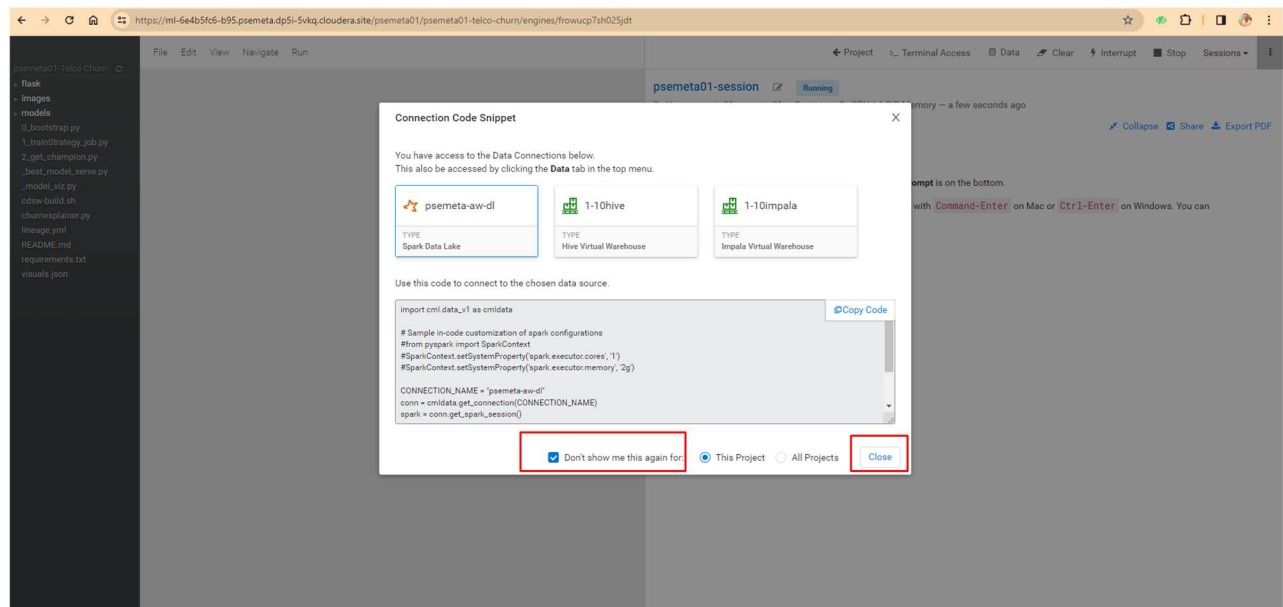
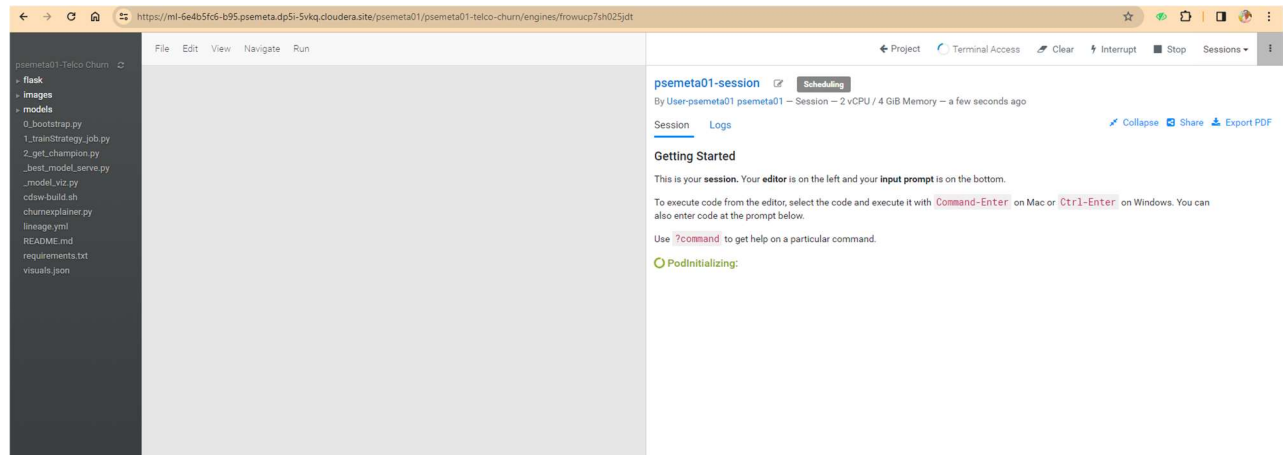
Click on **Start Session**.

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.

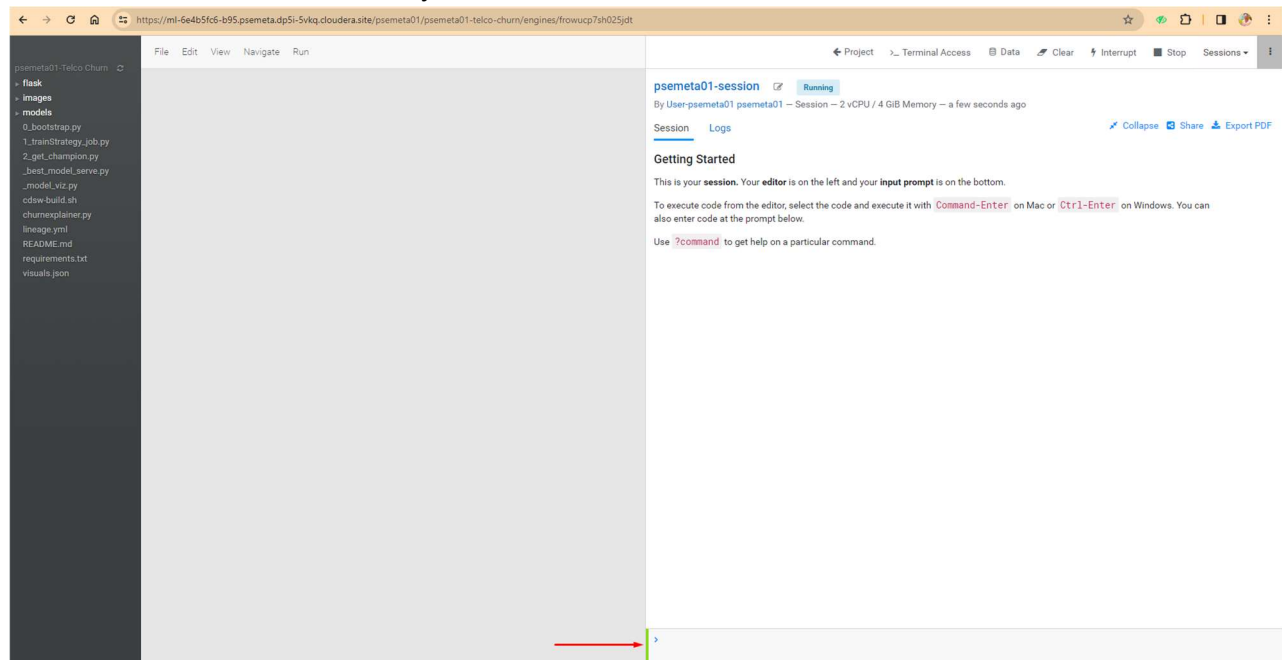
Editor	Kernel	Edition	Version
Workbench	Python 3.7	Standard	2024.02
- Warning:** A yellow box with a warning icon stating 'Selected Python kernel is deprecated! Please consider using kernel with higher Python version.'
- Enable Spark:** A toggle switch that is currently turned on.
- Spark Version:** 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 GB 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.




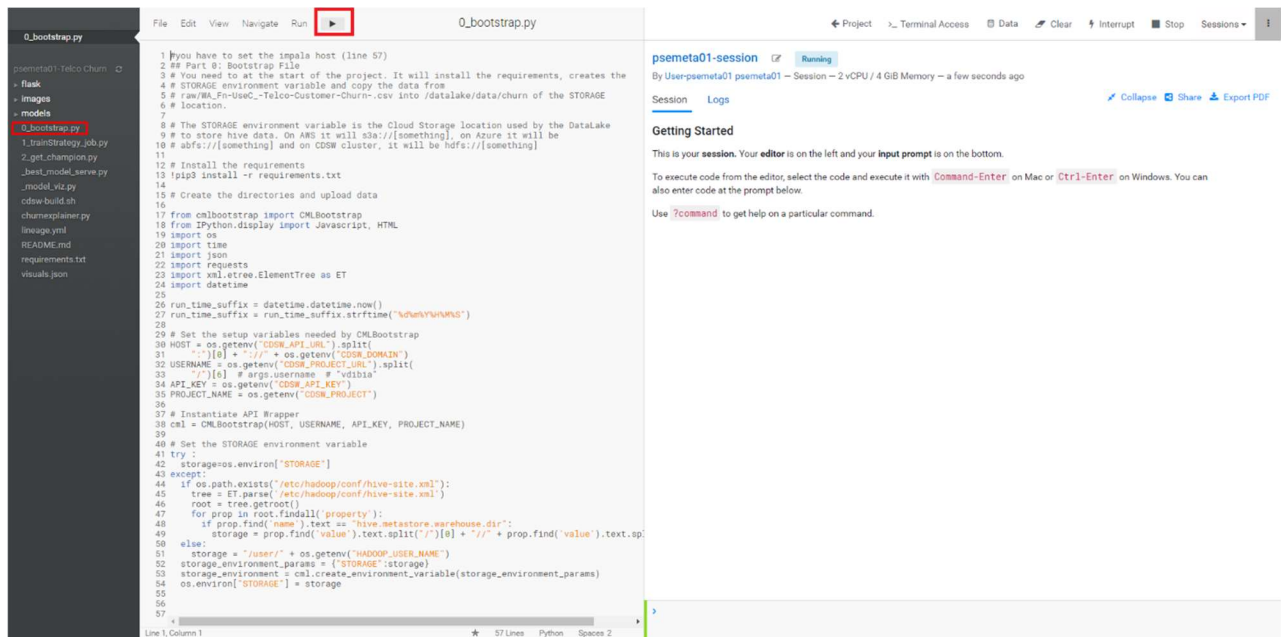
8. The editor/notebook located on the right side of the window will be in **Scheduling** status, and the bottom command bar flashing red. This means that CML is allocating computation for your session.

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.



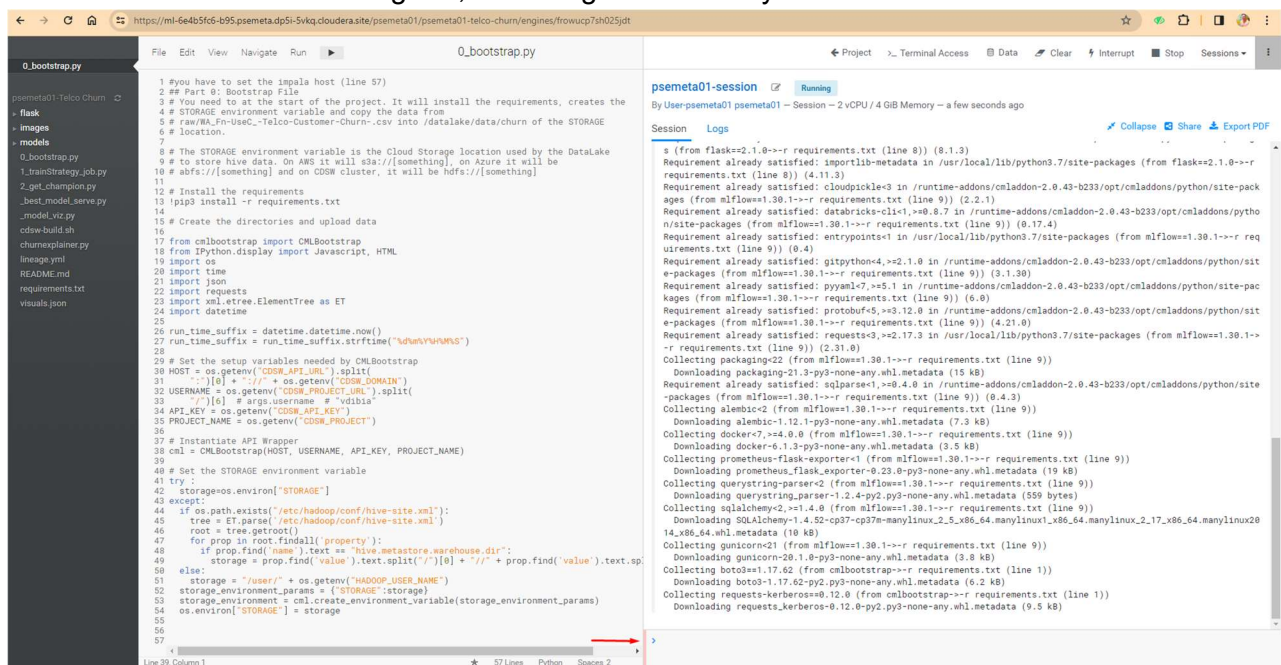
9. The first script/code to run is **0_bootstrap.py**. This Python code configures the libraries required for the project and integration with Lakehouse tables you populated before. Select (just

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.



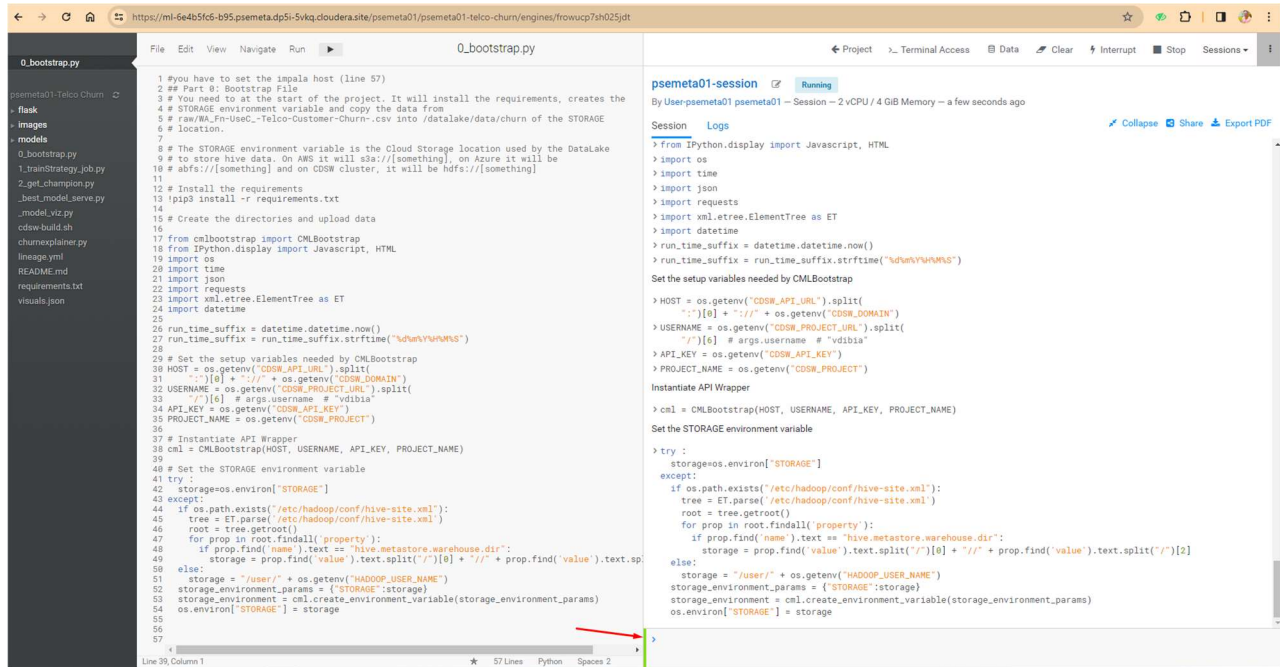
The screenshot shows the Databricks IDE interface. On the left, the file explorer lists files like '0_bootstrap.py', '1_trainStrategy_job.py', etc. The '0_bootstrap.py' file is selected. The main editor displays the Python code for '0_bootstrap.py'. On the right, the 'psemeta01-session' tab is active, showing the command output. The output indicates that the code is running successfully, with various dependencies being installed and the environment being set up.

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.



This screenshot shows the Databricks IDE interface during the execution of the '0_bootstrap.py' script. The file explorer on the left shows the file is selected. The main editor displays the Python code. On the right, the 'psemeta01-session' tab is active, showing the command output. The output indicates that the code is running successfully, with various dependencies being installed and the environment being set up. The bottom command bar is flashing red, indicating that it is busy.

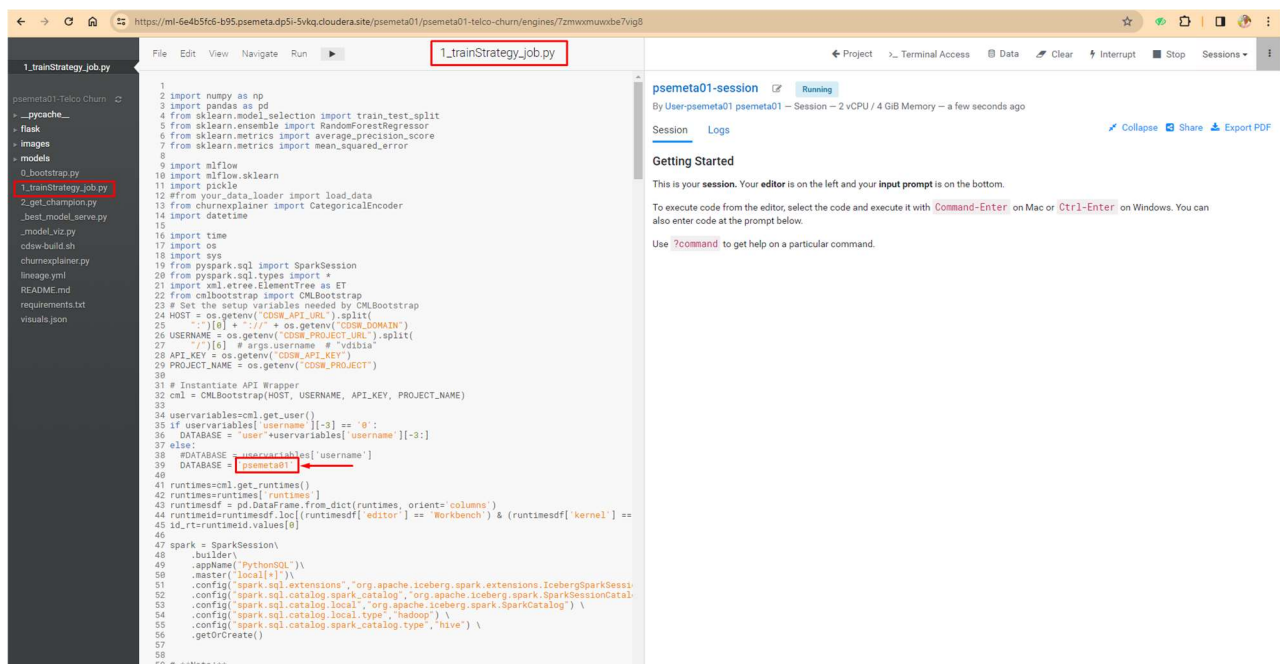
The green command bar indicates that the execution of the code has been finished. This bootstrap code takes 3-4 minutes to run.



```
1 #You have to set the impala host (line 57)
2 # Part 0: Bootstrap File
3 # You need to at the start of the project. It will install the requirements, creates the
4 # STORAGE environment variable and copy the data from
5 # raw/NA_Fn-UseC-Telco-Customer-Churn.csv into /datalake/data/churn of the STORAGE
6 # location.
7
8 # The STORAGE environment variable is the Cloud Storage location used by the DataLake
9 # to store hive data. On AWS it will s3a://[something], on Azure it will be
10 # abfs://[something] and on CDWS cluster, it will be hdfs://[something]
11
12 # Install the requirements
13 pip3 install -r requirements.txt
14
15 # Create the directories and upload data
16
17 from clbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import time
21 import json
22 import requests
23 import xml.etree.ElementTree as ET
24 import datetime
25
26 run_time_suffix = datetime.datetime.now()
27 run_time_suffix = run_time_suffix.strftime("%d%h%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDWS_API_URL").split(
31     "://")[0] + "://" + os.getenv("CDWS_DOMAIN")
32 USERNAME = os.getenv("CDWS_PROJECT_URL").split(
33     "://")[0] + "://" + os.getenv("CDWS_PROJECT")
34 API_KEY = os.getenv("CDWS_API_KEY")
35 PROJECT_NAME = os.getenv("CDWS_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try:
42     storage=os.environ["STORAGE"]
43 except:
44     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
45         tree = ET.parse("/etc/hadoop/conf/hive-site.xml")
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split("/")[-1] + "/" + prop.find('value').text.split("/")[-1]
50 else:
51     storage = "/user/" + os.getenv("HADOOP_USER_NAME")
52     storage_environment_params = {"STORAGE":storage}
53     storage_environment = cml.create_environment_variable(storage_environment_params)
54     os.environ["STORAGE"] = storage
55
56
57
```


10. The second script/code to run is **1_trainStrategy_job.py**. This Python code will create the Experiment to run the model with three different hyper parameters and records the precision. 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.

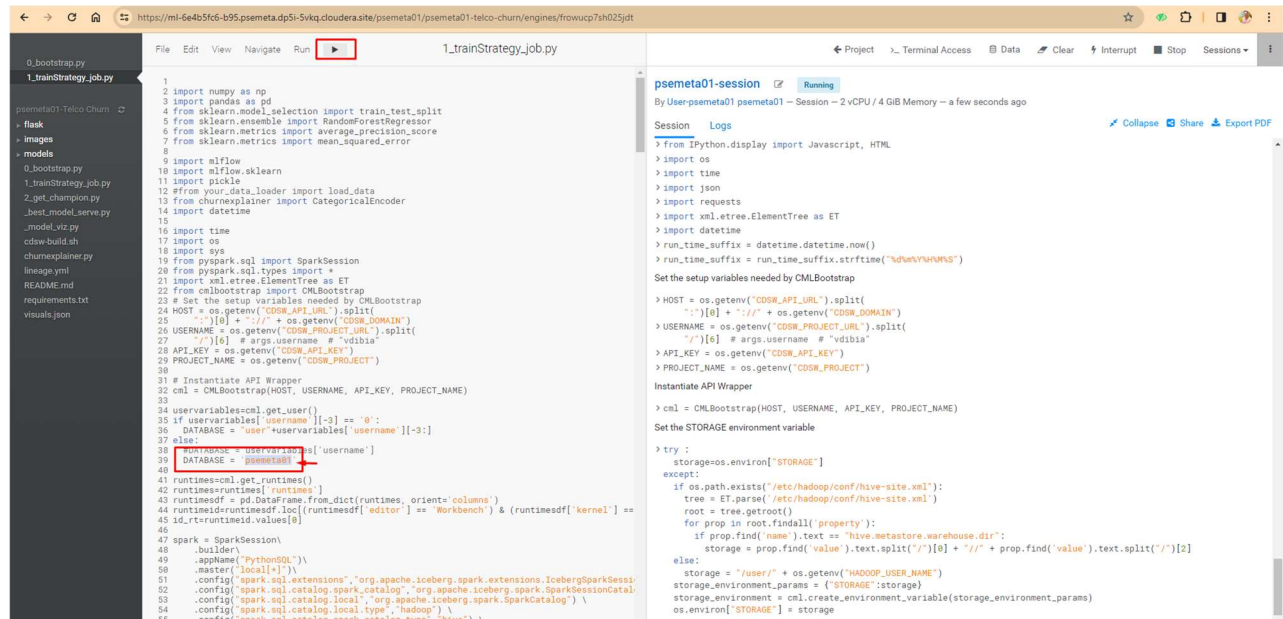
In the code change the line that has the value of **DATABASE** to your username ex – **psmeta01**



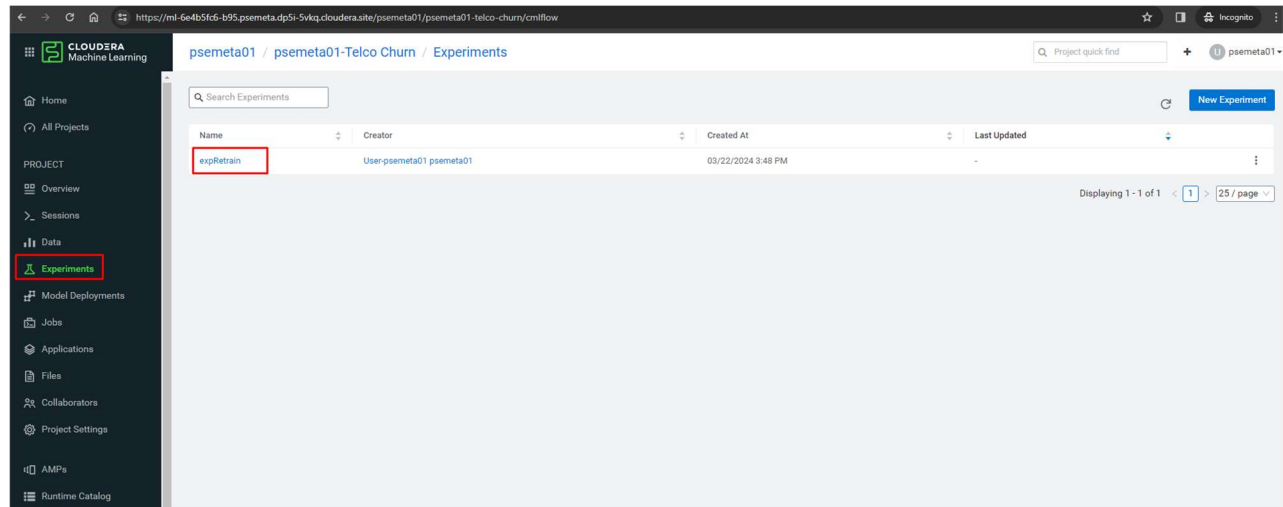
```
1
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.ensemble import RandomForestRegressor
6 from sklearn.metrics import average_precision_score
7 from sklearn.metrics import mean_squared_error
8
9 import mlflow
10 import mlflow.sklearn
11 import pickle
12 #from your_data_loader import load_data
13 from chumexplainer 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 Import +
21 import xml.etree.ElementTree as ET
22 from clbootstrap import CMLBootstrap
23 # Set the setup variables needed by CMLBootstrap
24 HOST = os.getenv("CDWS_API_URL").split(
25     "://")[0] + "://" + os.getenv("CDWS_DOMAIN")
26 USERNAME = os.getenv("CDWS_PROJECT_URL").split(
27     "://")[0] + "://" + os.getenv("CDWS_PROJECT")
28 API_KEY = os.getenv("CDWS_API_KEY")
29 PROJECT_NAME = os.getenv("CDWS_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'+user_variables['username']
39     DATABASE = 'psmeta01'
40
41 runtimes=cml.get_runtimes()
42 runtimes=runtimes[runtimes]
43 runtimesdf = pd.DataFrame.from_dict(runtimes, orient='columns')
44 runtimesdf=runtimesdf.loc[(runtimesdf['editor'] == 'Workbench') & (runtimesdf['kernel'] ==
45 id_rtruntimesdf.values[0])
46
47 spark = SparkSession\
48     .builder\
49     .appName("pythonSQL")\
50     .master("local[*]")\
51     .config("spark.sql.extensions", "org.apache.iceberg.spark.extensions.IcebergSparkSession")\
52     .config("spark.sql.catalog.spark_catalog", "org.apache.iceberg.spark.SparkSessionCatalog")\
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")\
56     .getOrCreate()
57
58
59 # **Note:**
```

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).



```
1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestRegressor
5 from sklearn.metrics import average_precision_score
6 from sklearn.metrics import mean_squared_error
7
8 import mlflow
9
10 import os
11 import sys
12 from pyspark.sql import SparkSession
13 from pyspark.sql.types import *
14 import xml.etree.ElementTree as ET
15
16 # Set the setup variables needed by CMLBootstrap
17 HOST = os.getenv("CDGW_API_URL").split(
18     "/"
19 )[0] + "://" + os.getenv("CDGW_DOMAIN")
20 USERNAME = os.getenv("CDGW_PROJECT_URL").split(
21     "/"
22 )[0] + "/" + os.getenv("CDGW_PROJECT")
23 API_KEY = os.getenv("CDGW_API_KEY")
24 PROJECT_NAME = os.getenv("CDGW_PROJECT")
25
26 # Instantiate API Wrapper
27 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
28
29 # Set the setup variables needed by CMLBootstrap
30 user_variables = cml.get_user_variables()
31 if user_variables["username"][-3] == "0":
32     DATABASE = "user" + user_variables["username"][-3:]
33 else:
34     DATABASE = user_variables["username"]
35     DATABASE = "psmetadb"
36
37 # Set the setup variables needed by CMLBootstrap
38 runtimes = cml.get_runtimes()
39 runtimesdf = pd.DataFrame.from_dict(runtimes, orient="columns")
40 runtimeid = runtimesdf.loc[(runtimesdf["editor"] == "Workbench") & (runtimesdf["kernel"] == "Python3")]
41 runtimeid = runtimeid.values[0]
42
43 # Create SparkSession
44 builder = SparkSession.builder()
45     .appName("PythonSQL")
46     .master("local[*]")
47     .config("spark.sql.extensions", "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions")
48     .config("spark.sql.catalog.spark_catalog", "org.apache.iceberg.spark.SparkSessionCatalog")
49     .config("spark.sql.catalog.local", "org.apache.iceberg.spark.SparkCatalog")
50     .config("spark.sql.catalog.local.type", "hadoop")
51     .config("spark.sql.catalog.spark_catalog.type", "hive")
52     .getOrCreate()
```

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.

Experiment

Experiment Name: expRetrain
 Experiment ID: vaid-ln4b-9etu-mfgg
 Artifact Location: /home/cdsw/experiments/vaid-ln4b-9etu-mfgg

> Notes

Runs (3)

Q metrics.mae < 1 and p...

Status	Start Time	Run Name	Duration	User	Source	Version	Models	Parameters	compute	dataset	Metrics	Tags
✓	2024-03-22 03:48:57	run_4856_0	6.6s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8518098598...	oz2mi70aejkh...
✓	2024-03-22 03:49:04	run_4856_1	4.1s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8732466313...	oz2mi70aejkh...
✓	2024-03-22 03:49:08	run_4856_2	4.5s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8669739750...	oz2mi70aejkh...

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*.

Experiment


Experiment Name: expRetrain
 Experiment ID: vaid-ln4b-9etu-mfgg
 Artifact Location: /home/cdsw/experiments/vaid-ln4b-9etu-mfgg

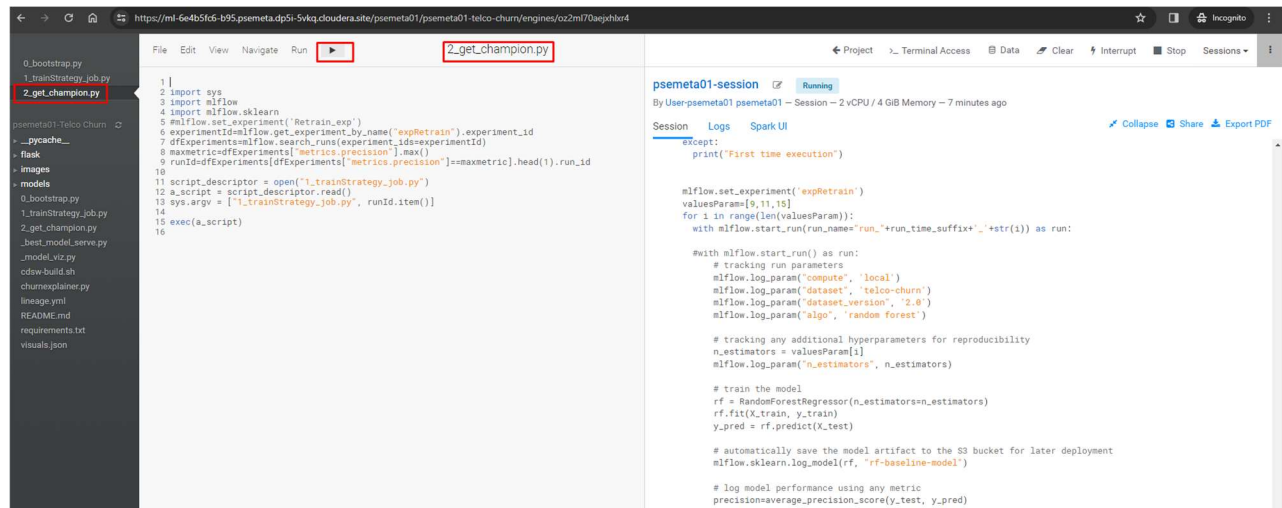
> Notes

Runs (3)


Q metrics.mse < 1 and p...

Status	Start Time	Run Name	Duration	User	Source	Version	Models	Parameters	compute	dataset	Metrics	Tags
✓	2024-03-22 03:48:57	run_4856_0	6.6s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8518098598...	oz2mi70aejkh...
✓	2024-03-22 03:49:04	run_4856_1	4.1s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8732466313...	oz2mi70aejkh...
✓	2024-03-22 03:49:08	run_4856_2	4.5s	psemeta01	ipython3	49930d	sklearn	random forest	local	telco-churn	0.8669739750...	oz2mi70aejkh...

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.



```
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
```

psemeta01-session  Running
By User-psemeta01 psemeta01 - Session - 2 vCPU / 4 GiB Memory - 7 minutes ago

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

```
except:
    print("First time execution")

mlflow.set_experiment("expRetrain")
valuesParam=[9,11,15]
for i in range(len(valuesParam)):
    with mlflow.start_run(run_name="run_"+run_time_suffix+"_"+str(i)) as run:

        #with mlflow.start_run() as run:
            # tracking run parameters
            mlflow.log_param("compute", "local")
            mlflow.log_param("dataset", "telco-churn")
            mlflow.log_param("dataset_version", "2.0")
            mlflow.log_param("algo", "random forest")

            # tracking any additional hyperparameters for reproducibility
            n_estimators = valuesParam[i]
            mlflow.log_param("n_estimators", n_estimators)

            # train the model
            rf = RandomForestRegressor(n_estimators=n_estimators)
            rf.fit(X_train, y_train)
            y_pred = rf.predict(X_test)

            # automatically save the model artifact to the S3 bucket for later deployment
            mlflow.sklearn.log_model(rf, "rf-baseline-model")

            # log model performance using any metric
            precision=average_precision_score(y_test, y_pred)
            #max = max(maxed_precision_test_u_max)
```

After a few seconds, you will see the following message “Deploying Model...” 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: mob3r8kbuw2c56kg97ucyzge55t29b

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: m1qn68biwont284ezovfcsrt6p6g8b

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 **ModelOpsChurn**.

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 displays a table with two rows:

Model Deployment	Source	Status	Replicas	CPU	Memory	Last Deployed	Actions
ModelOpsChurn_psmeta01	_bestL...	Deployed	1 / 1	1	2.00 GiB	Mar 22, 2024, 04:28 PM	Stop
ModelViz_psmeta01	_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_psmeta01' Overview tab. The left sidebar is the same as in the previous screenshot, with 'Model Deployments' highlighted. The main content area is divided into several sections:

- Description:** visualization a given model prediction
- Sample Code:** A code editor showing a curl command for a POST request to a model service endpoint.
- Sample Response:** A JSON object: `{}`
- Model Details:** A table with the following information:
 - Source: cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/4b5f5045-a746-458f-ab03-20505ace5ee9
 - Model Id: 5
 - Model CRN: cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/4b5f5045-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/4b5f5045-a746-458f-ab03-20505ace5ee9
 - Build Id: 5
 - Build CRN: cm.cdp.mlus-west-1-d1a4553c-a799-432d-8e54-372cc2ab95f2:workspace-e1bc006e-6269-4eab-aa8a-5974c1264c8d/4b5f5045-a746-458f-ab03-20505ace5ee9
 - Deployed By: psmeta01
 - Comment: Initial revision.
 - Runtime Image: Python 3.7 (Standard)
 - File: _model_viz.py
 - Function: predict
- Model Resources:**
 - Replicas: 1
 - Total CPU: 1 vCPUs
 - Total Memory: 2.00 GiB

At the bottom, it shows 'Workspace: ssa-cml-workspace' and 'Cloud Provider: AWS (AWS)'.

To test it and make a request to the model, scroll down, and click on the button **Test**, which will take the value in JSON format that is in the field **Input** and will make the request call to the model. What you see in the field **Result** is the response from the model in JSON format. If you wish, you can change some of the parameters of the **Input** field (for example, change some values from *Not* to *Yes*), and call the model again, and observe the value of the attribute *probability* of the response to see if there were any changes.



At this point note the value of the **url** and **accessKey** somewhere as you will use it to call the churn related values given by model in the Data viz application. Please take a note of these values which will be used in the next lab.

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

