

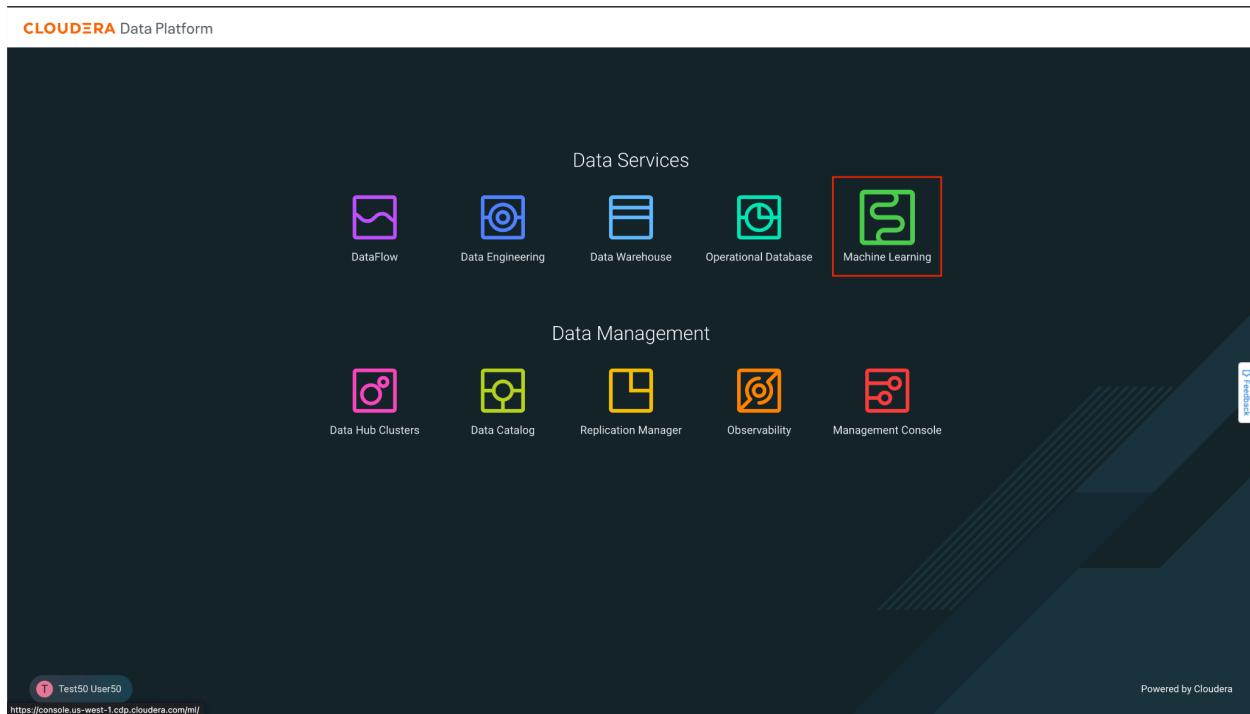
# Data Lifecycle in CDP Public Cloud

## Machine Learning Lab

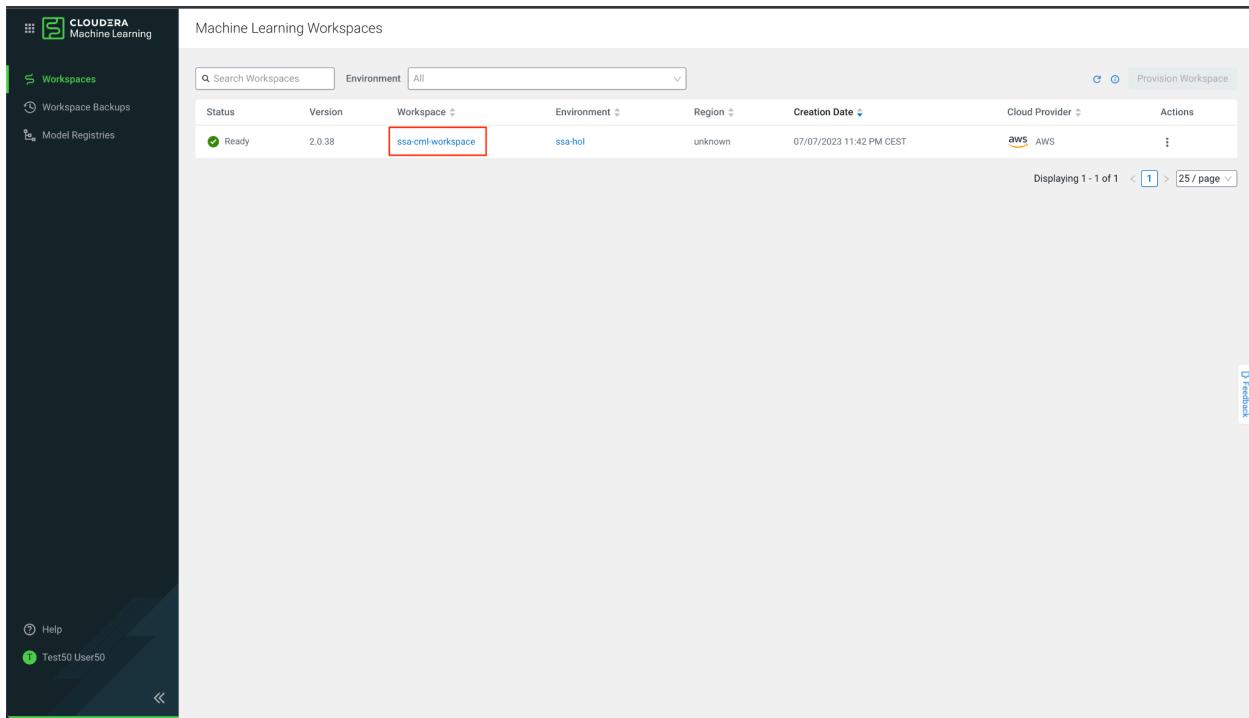
Goals:

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

1. Click on Machine Learning from CDP PC Home:



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



The screenshot shows the 'Machine Learning Workspaces' page in the Cloudera Machine Learning interface. On the left, there's a sidebar with options like 'Workspaces' (which is selected and highlighted in green), 'Workspace Backups', and 'Model Registries'. The main area has a header 'Machine Learning Workspaces' with a search bar and a 'Provision Workspace' button. Below the header is a table with the following data:

Status	Version	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	2.0.38	ssa-cml-workspace	ssa-hol	unknown	07/07/2023 11:42 PM CEST	AWS	[More]

Below the table, it says 'Displaying 1 - 1 of 1'. A red box highlights the 'ssa-cml-workspace' column. On the right side of the page, there are 'Feedback' and 'Help' links.

3. Once in the Workspace, click on **Projects** on the left side. You should see the following interface. Here are the projects you have created. It is time to create a new project. Click on the blue button **New Project**.

The screenshot shows the Cloudera Machine Learning workspace interface. On the left, a dark sidebar menu lists various options: Projects (selected), Sessions, Experiments, Models, Jobs, Applications, User Settings, AMPS, Runtime Catalog, Site Administration, and Learning Hub. At the bottom of the sidebar are Help and a version number (2.0.38-b125). On the right, the main area is titled "Projects". It features a search bar with "Search Projects" and "Scope: My Projects", and a dropdown for "Creator: All". A "View Resource Usage Details" link is also present. Below the search bar, there is a "New Project" button highlighted with a red box. The central message says "You currently don't have any projects" with a small cloud icon above it. A descriptive text explains that projects hold code, configuration, and libraries needed for reproducible runs, and that each project is independent. At the bottom, it shows the workspace name "ssa-cml-workspace" and the cloud provider "aws (AWS)".

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

**Project Name:** 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/campossalex/TelcoChurn>

The screenshot shows the 'New Project' creation interface. The 'Project Name' field contains 'Telco Churn'. The 'Project Visibility' section shows 'Private' selected. The 'Initial Setup' section shows 'Git' selected. The 'Git URL' field contains 'https://github.com/campossalex/TelcoChurn'.

New Project

Project Name

Telco Churn

Project Description

Project Visibility

Private - Only added collaborators can view the project  
 Public - All authenticated users can view this project.

Initial Setup

Blank Template AMPs Local Files Git

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

HTTPS  SSH

https://github.com/campossalex/TelcoChurn

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

Toggle **Advanced Options** on.

Make sure to select:

**Editor: Workbench**

**Kernel: Python 3.7**

**Edition: Standard**

Click on **Add Runtime**

Click on **Create Project**

The screenshot shows the 'Create Project' dialog with the 'Runtimes' tab selected. It displays a table of current runtimes and a form to add a new one. A warning message at the bottom indicates a deprecated kernel selection.

**Runtimes**

Projects are configured with the latest Python and R ML Runtimes. You can change this configuration under the Advanced Options.

Editor	Kernel	Edition	Version	Action
JupyterLab	Python 3.10	Nvidia GPU	2023.12	<a href="#">Remove</a>
JupyterLab	Python 3.10	Standard	2023.12	<a href="#">Remove</a>
PBJ Workbench	Python 3.10	Nvidia GPU	2023.12	<a href="#">Remove</a>
PBJ Workbench	Python 3.10	Standard	2023.12	<a href="#">Remove</a>
PBJ Workbench	R 4.3	Standard	2023.12	<a href="#">Remove</a>

Advanced Options

Editor [ⓘ](#) Kernel [ⓘ](#) Edition [ⓘ](#) Version

[Workbench](#) [Python 3.7](#) [Standard](#) [2023.12](#)

**⚠ Selected Python kernel is deprecated!**  
Please consider using kernel with higher Python version.

[Add Runtime](#)

[Cancel](#) [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 workloads

**Files**, assets that are part of the project, such as files, scripts and code

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. On the left is a sidebar with navigation links: All Projects, Overview (which is selected), Sessions, Data, Experiments, Models (selected), Jobs, Applications, Files, Collaborators, and Project Settings. At the bottom of the sidebar are Help and a version number (2.0.38-b125). The main area is titled "user050 / Telco Churn". It has tabs for "Telco Churn" and other project details. A "Project quick find" search bar is at the top right. Below it are buttons for "0 Fork" and "New Session". The "Overview" section contains three sections: "Models" (with a note about no models yet), "Jobs" (with a note about no jobs yet), and "Files". The "Files" section lists the following files:

Name	Size	Last Modified	Action
flask	-	a few seconds ago	Edit
images	-	a few seconds ago	Edit
models	-	a few seconds ago	Edit
0.bootstrap.py	1.95 kB	a few seconds ago	Edit
1.trainStrategy.job.py	18.63 kB	a few seconds ago	Edit
2.get_champion.py	508 B	a few seconds ago	Edit
_best_model.serve.py	2.74 kB	a few seconds ago	Edit
_model_viz.py	4.21 kB	a few seconds ago	Edit
cdsw-build.sh	44 B	a few seconds ago	Edit
churnexplainer.py	6.69 kB	a few seconds ago	Edit
lineage.yml	610 B	a few seconds ago	Edit
README.md	11.97 kB	a few seconds ago	Edit
requirements.txt	197 B	a few seconds ago	Edit
visuals.json	281.07 kB	a few seconds ago	Edit

At the bottom of the "Files" section are buttons for "Download", "New", and "Upload". Below the table is a link "Show Hidden Files". At the very bottom of the interface, it says "Workspace: ssa-cml-workspace" and "Cloud Provider: AWS (AWS)".

6. When starting a new session, make sure:

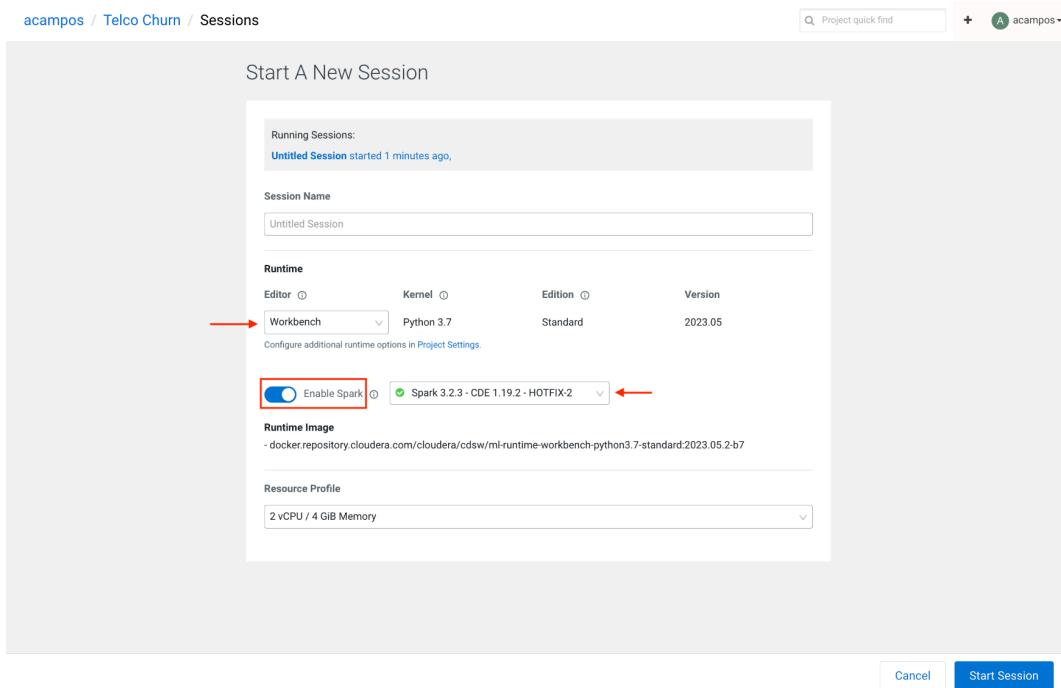
Select **Workbench** in the Editor selector.

You can safely ignore the message “Selected Python kernel is deprecated!”

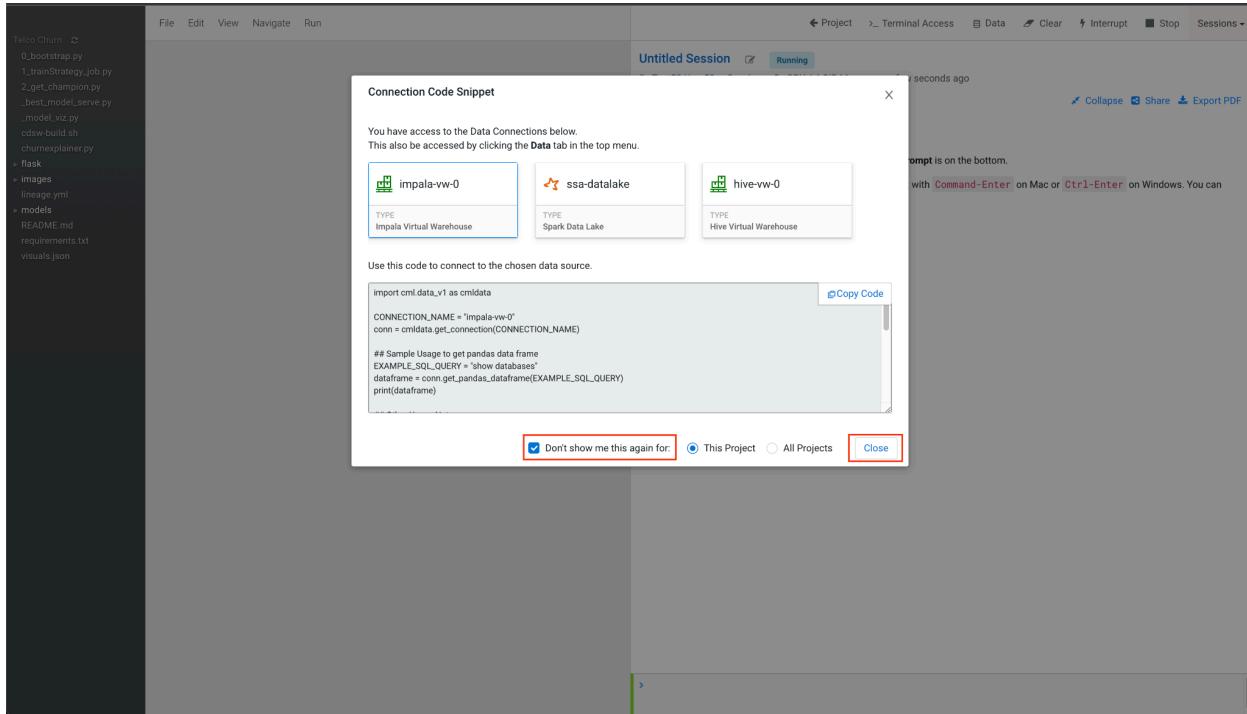
Enable **Spark**, by toggling it on.

Select **Spark 3.2.x**, in the Spark version selector.

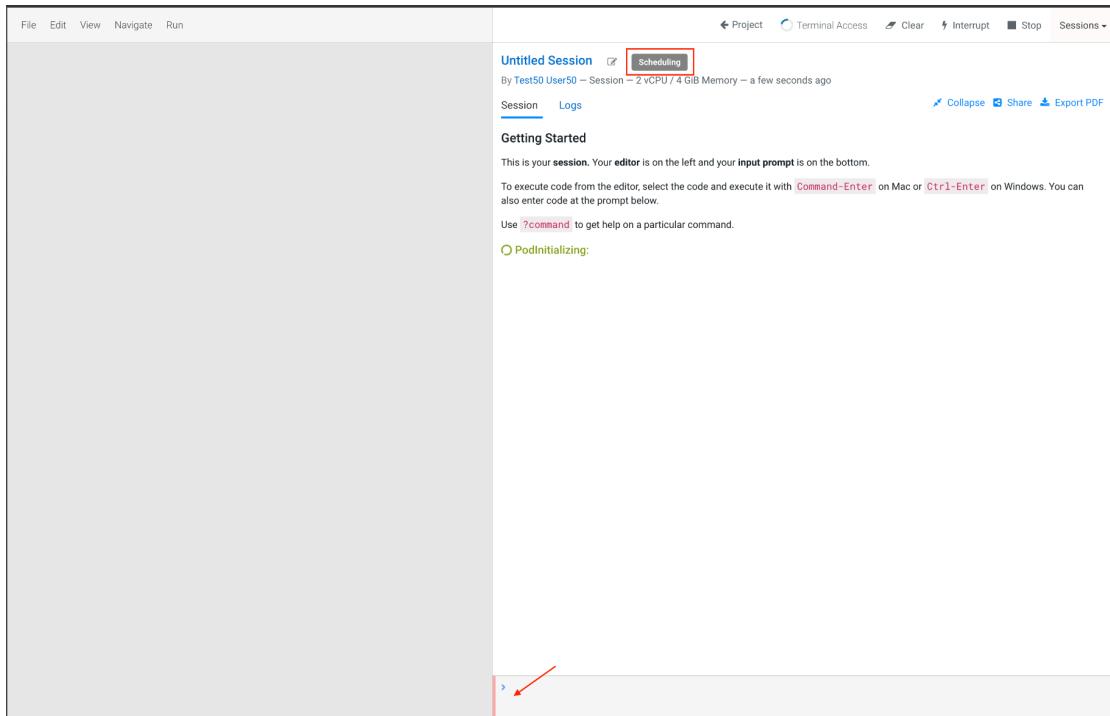
Click on the button **Start Session**



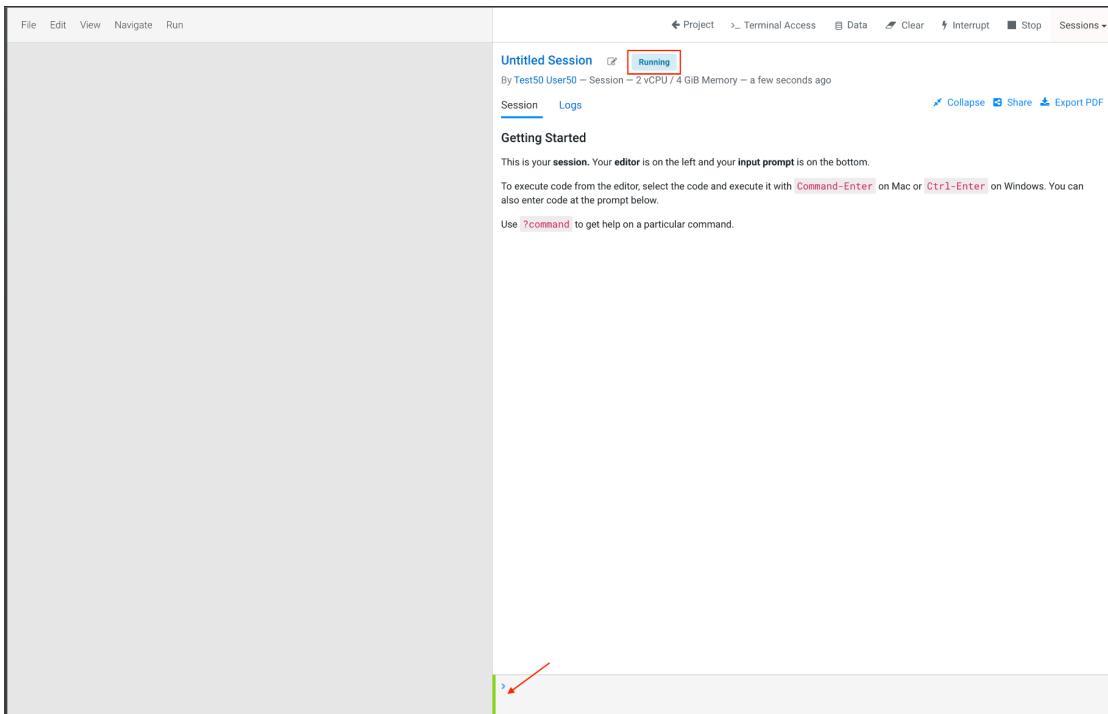
7. When you start a session for the first time, it will ask if you want to use a data connection. This project does not need this type of connection. mark the check of **Don't show me this again**, and then click the button **Close**, so this window will not appear anymore.



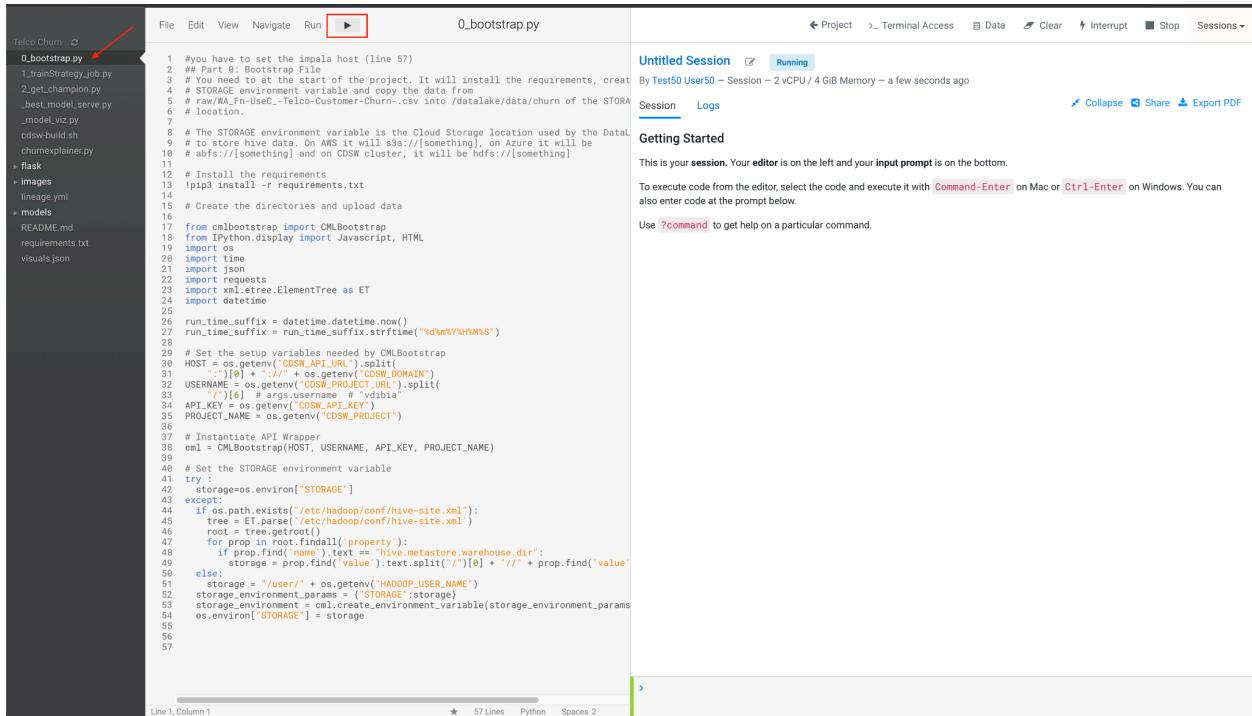
8. The editor/notebook located on the right side of the window will be in **Scheduling** status, and the bottom command bar flashing red. This means that CML is allocating resources for your session.



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 DataBrick interface with the following details:

- Left Sidebar:** Shows the project structure for "Telco Churn" with files like `0_bootstrap.py`, `1_trainStrategyJob.py`, `2_getChampion.py`, `_bestModelServer.py`, `_modelViz.py`, `cdfw-build.sh`, `churnExplainer.py`, `flask`, `images`, `lineage.yml`, `models`, `README.md`, `requirements.txt`, and `visuals.json`.
- Editor Area:** The file `0_bootstrap.py` is selected and displayed in the main editor window. The code is as follows:

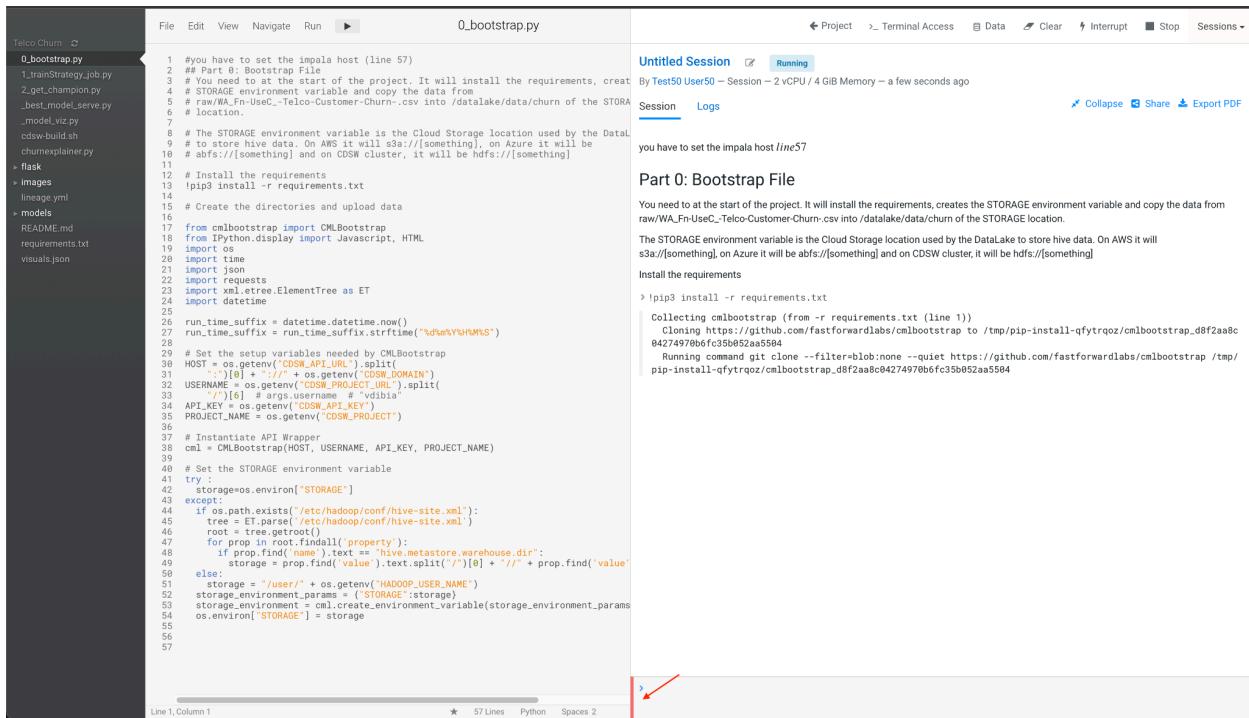
```

1 #you have to set the impala host (line 57)
2 ## Part B: Bootstrap File
3 # This is at the start of the project. It will install the requirements, create
4 # STORAGE environment variable and copy the data from
5 # raw/Fn-UserC-Telco-Customer-Churn--.csv into /datalake/data/churn of the STORA
6 # location
7
8 # The STORAGE environment variable is the Cloud Storage location used by the Data
9 # to store hive data. On AWS it will be s3://[something], on Azure it will be
10 # abfs://[something] and on CDSW 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 cmlbootstrap 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%m%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSN_API_URL").split(
31     ":" )[0] + ":" + os.getenv("CDSN_DOMAIN")
32 USERNAME = os.getenv("CDSN_PROJECT_URL").split(
33     "/" )[1].args.username + "-vizual"
34 API_KEY = os.getenv("CDSN_API_KEY")
35 PROJECT_NAME = os.getenv("CDSN_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 not 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 == "metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('"')[0] + "//" + prop.find('value').text
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

```

- Session Bar:** Shows "Untitled Session" is running. Other options include Project, Terminal Access, Data, Clear, Interrupt, Stop, and Sessions.
- Bottom Status:** Shows Line 1, Column 1, 57 Lines, Python, Spaces 2.

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.



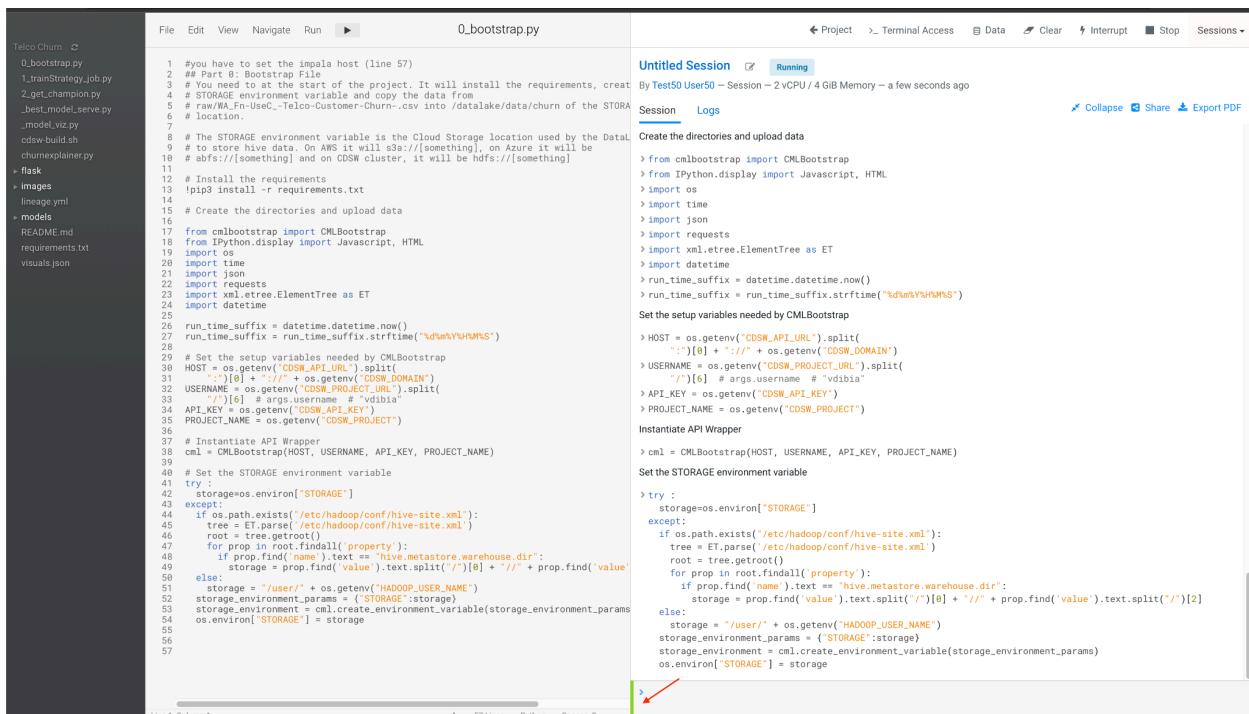
The screenshot shows the DataBrick interface. On the left is a code editor with a file named `0_bootstrap.py`. The code is a Python script for setting up a DataLake. On the right is a terminal session window titled "Untitled Session" with the status "Running". The bottom status bar is red, indicating the session is active. A red arrow points to this status bar.

```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
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, creat
4 # STORAGE environment variable and copy the data from
5 # raw/WA_Fn-UseC-Telco-Customer-Churn-.csv into /datalake/data/churn of the STOR
6 # location.
7
8 # The STORAGE environment variable is the Cloud Storage location used by the DataL
9 # to store hive data. On AWS it will s3://[something], on Azure it will be
10 # abfs://[something] and on CDSW 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 cmlbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import json
21 import requests
22 import xml.etree.ElementTree as ET
23 import datetime
24
25 run_time_suffix = datetime.datetime.now()
26 run_time_suffix = run_time_suffix.strftime("%d%b%Y%H%M%S")
27
28 # Set the setup variables needed by CMLBootstrap
29 HOST = os.getenv("CDSW_API_URL").split(
30     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
31 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
32     "/" )[6] # args.username # "vibia"
33 API_KEY = os.getenv("CDSW_API_KEY")
34 PROJECT_NAME = os.getenv("CDSW_PROJECT")
35
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('/')[0] + "//" + prop.find('value'
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

```

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



The screenshot shows the DataBrick interface after the code has finished executing. The terminal session window now shows "Session 1" and "Logs". The bottom status bar is green, indicating the session has completed. A red arrow points to this status bar.

```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
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, creat
4 # STORAGE environment variable and copy the data from
5 # raw/WA_Fn-UseC-Telco-Customer-Churn-.csv into /datalake/data/churn of the STOR
6 # location.
7
8 # The STORAGE environment variable is the Cloud Storage location used by the DataL
9 # to store hive data. On AWS it will s3://[something], on Azure it will be
10 # abfs://[something] and on CDSW 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 cmlbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import json
21 import requests
22 import xml.etree.ElementTree as ET
23 import datetime
24
25 run_time_suffix = datetime.datetime.now()
26 run_time_suffix = run_time_suffix.strftime("%d%b%Y%H%M%S")
27
28 # Set the setup variables needed by CMLBootstrap
29 HOST = os.getenv("CDSW_API_URL").split(
30     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
31 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
32     "/" )[6] # args.username # "vibia"
33 API_KEY = os.getenv("CDSW_API_KEY")
34 PROJECT_NAME = os.getenv("CDSW_PROJECT")
35
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('/')[0] + "//" + prop.find('value'
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 will record 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. Once the file is selected, click on the button  to run the code. Once the execution is finished (approximately 1 minute), click on the button **Project**, located in the upper right bar of the session to go back to the project home.

The screenshot shows a Jupyter Notebook interface with two panes. The left pane displays the code for `1_trainStrategy.job.py`, which includes imports for numpy, pandas, sklearn, and mlflow, along with logic for reading data from a CSV file, setting up a CategoricalEncoder, and training a RandomForestRegressor. The right pane shows the execution of this code in an "Untitled Session" notebook, with the status bar indicating "Running". The terminal access tab shows the command `mlflow.tracking.fluent: Experiment with name 'expRetrain' does not exist. Creating a new experiment.`. The session log shows the first time execution and the start of the run. The mlflow command `mlflow.end_run()` is shown at the bottom of the code in the left pane.

```
File Edit View Navigate Run ▶ 1_trainStrategy.job.py

Telco Churn
└── o_bootstrap.py
    ├── 1_trainStrategy.job.py
    ├── 2_get_champion.py
    ├── > __pycache__
    ├── _best_model.Serve.py
    ├── model_viz.py
    ├── cdsw-build.sh
    ├── churnexplainer.py
    > flask
    ├── images
    └── lineage.yml

README.md
requirements.txt
visuals.json

Line 1, Column 1  ★ 433 Lines  Python  Spaces 2
```

```
Untitled Session  Running
By Test10 User10 - Session = 2 vCPU / 4 GiB Memory – a few seconds ago
Session Logs Spark UI
mlflow.delete_experiment(experimentId)

time.sleep(20)
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)

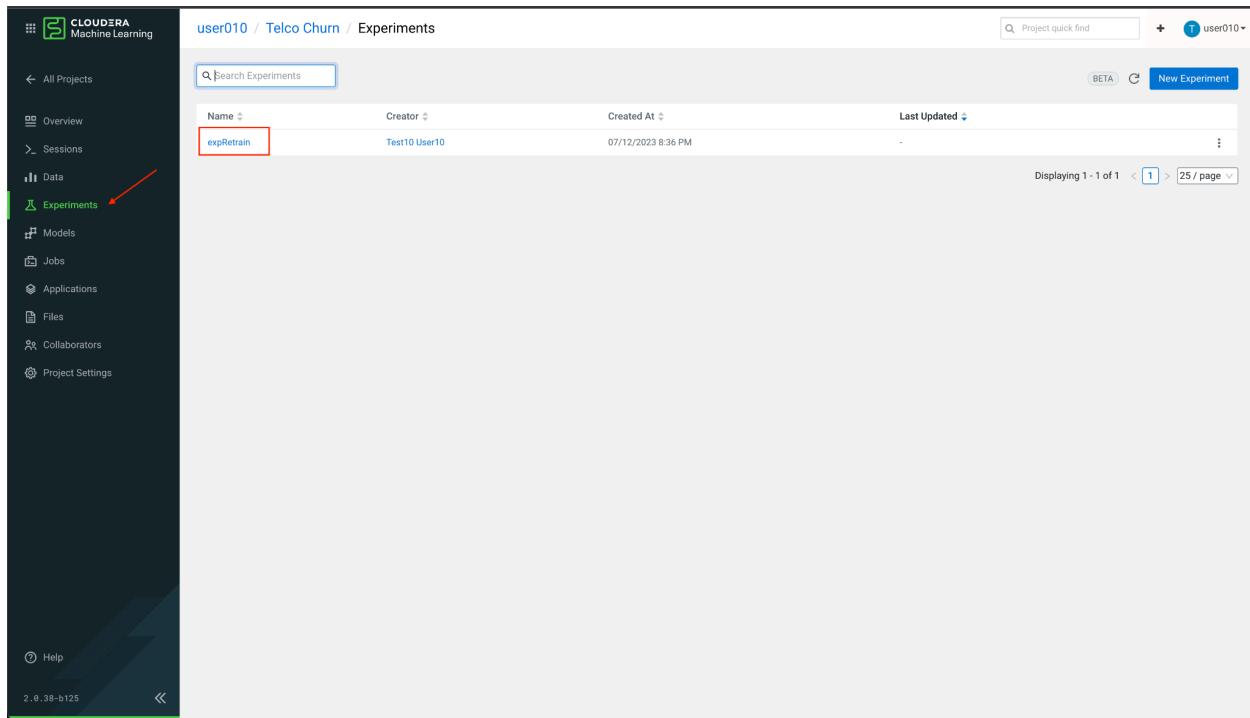
        # automatically save the model artifact to the S3 bucket for later deployment
        mlflow.sklearn.log_model(rf, 'rf-baseline-model')

        # log model performance using any metric
        precision=average_precision_score(y_test, y_pred)
        #mse = mean_squared_error(y_test, y_pred)
        mlflow.log.metric("precision", precision)

mlflow.end_run()

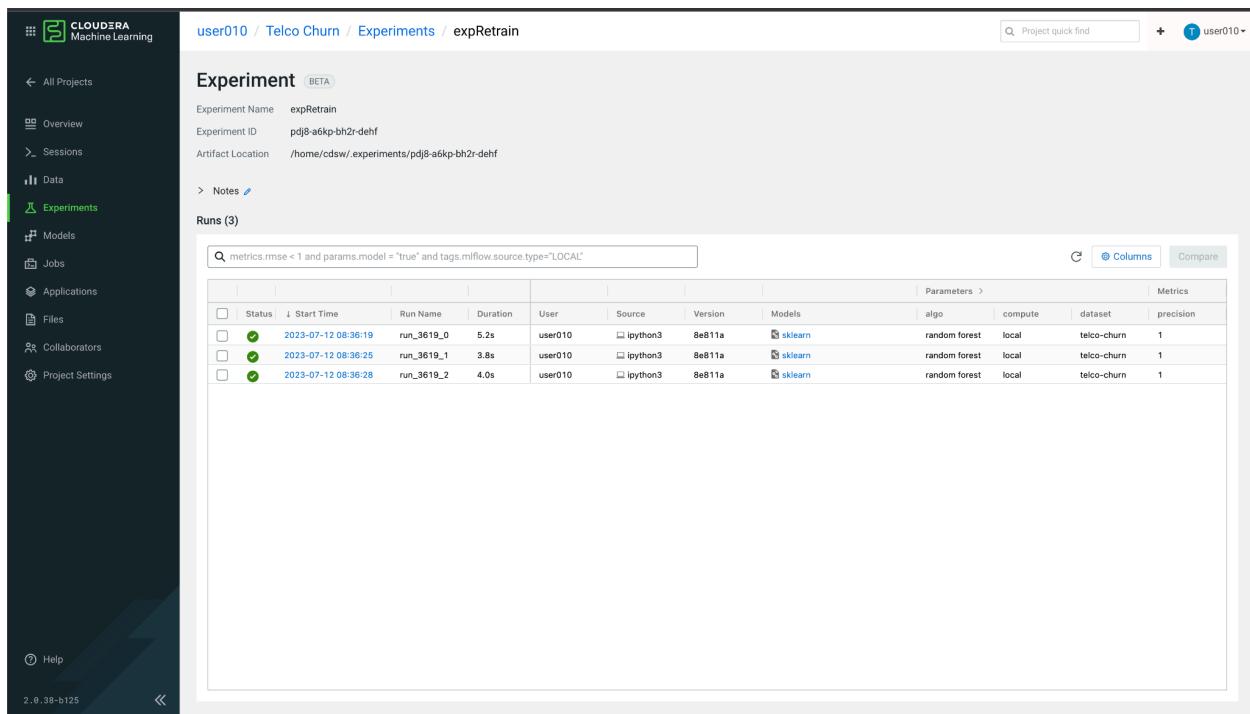
2023/07/12 18:36:19 INFO mlflow.tracking.fluent: Experiment with name 'expRetrain' does not exist. Creating a new experiment.
First time execution
```

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.



The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with various project management options like Overview, Sessions, Data, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The 'Experiments' option is highlighted with a green arrow. In the main content area, the path 'user010 / Telco Churn / Experiments' is shown. A search bar at the top says 'Search Experiments'. Below it is a table with columns: Name, Creator, Created At, and Last Updated. One row in the table is highlighted with a red box, showing the experiment name 'expRetrain', creator 'Test10 User10', and creation date '07/12/2023 8:36 PM'. At the bottom right of the table, it says 'Displaying 1 - 1 of 1' and has a '25 / page' dropdown.

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



The screenshot shows the details of the 'expRetrain' experiment. The left sidebar is identical to the previous screenshot. The main content area shows the experiment details: Experiment Name 'expRetrain', Experiment ID 'pdj8-a6kp-bh2r-dehf', and Artifact Location '/home/cdsweb/experiments/pdj8-a6kp-bh2r-dehf'. Below this, there's a 'Notes' section and a 'Runs (3)' section. The 'Runs (3)' section includes a search bar with the query 'metrics.rmse < 1 and params.model = "true" and tags.miflow.source.type="LOCAL"'. It displays three runs in a table:

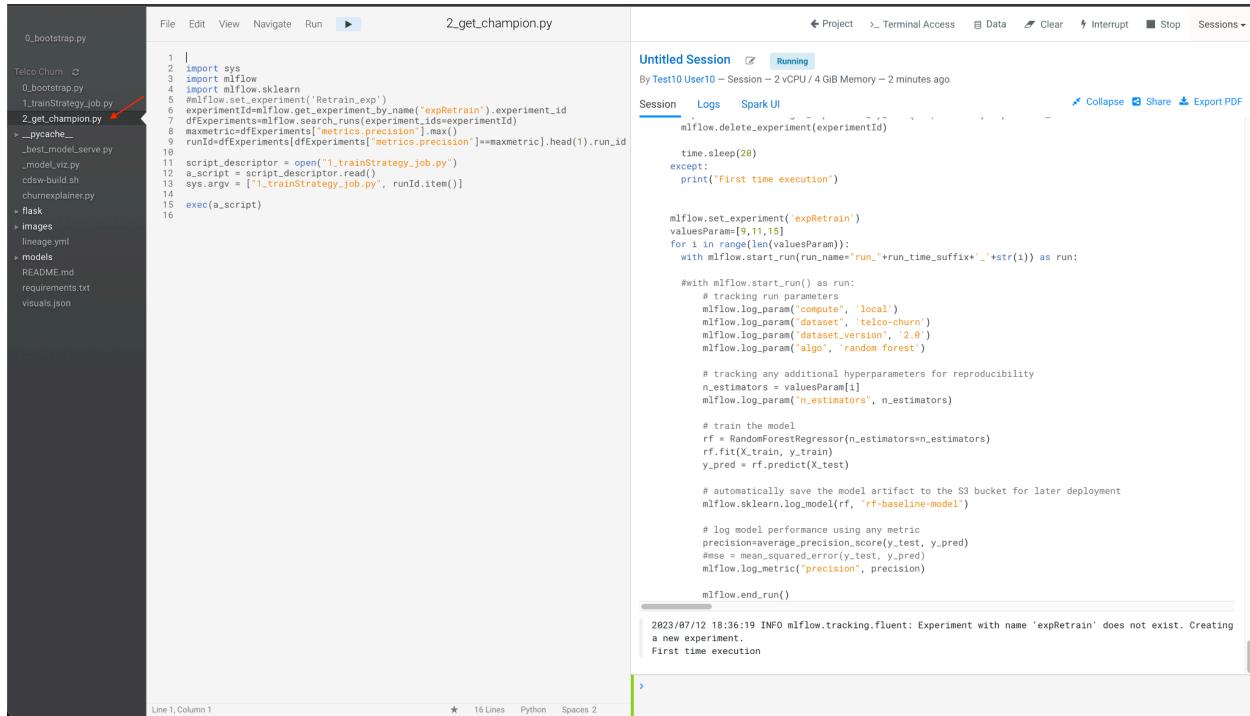
Status	Start Time	Run Name	Duration	User	Source	Version	Models	Parameters	Metrics
Success	2023-07-12 08:36:19	run_3619_0	5.2s	user010	ipython3	8e811a	sklearn	random forest local	telco-churn 1
Success	2023-07-12 08:36:25	run_3619_1	3.8s	user010	ipython3	8e811a	sklearn	random forest local	telco-churn 1
Success	2023-07-12 08:36:28	run_3619_2	4.0s	user010	ipython3	8e811a	sklearn	random forest local	telco-churn 1

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 then click on the only session that will appear on the list. If you didn't name your session when you started it (step 6), it should be called *Untitled Session*.

The screenshot shows the Cloudera Machine Learning interface. On the left, there is a sidebar with various project management and data science tools: All Projects, Overview, Sessions (highlighted with a red arrow), Data, Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. Below the sidebar, there is a Help section and a footer indicating the version 2.0.38-b125. The main content area is titled "user010 / Telco Churn / Sessions". It displays a table of sessions with the following columns: Status, Session, Kernel, Creator, Created At, and Duration. A single row is visible, showing a "Running" status for the "Untitled Session" (highlighted with a red box), created by "Test10 User10" on "07/12/2023 8:35 PM", which has been running for "Running since 1m 43s". There are buttons for Edit, Stop, and Delete at the bottom of the table row. The top right of the screen shows a user icon for "user010" and other navigation options. The bottom right of the main content area shows pagination information: "Displaying 1 - 1 < 1 > 25 / page".

Status	Session	Kernel	Creator	Created At	Duration
Running	Untitled Session	(Python 3.7 Workbench Standard)	Test10 User10	07/12/2023 8:35 PM	Running since 1m 43s

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 best 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.



```

File Edit View Navigate Run ▶ 2_get_champion.py

Untitled Session  Running
By Test10 User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago
Session Logs Spark UI
Collapse Share Export PDF

mlflow.set_experiment('Retrain_experiment')
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
time.sleep(20)
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:
        # 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)
        mse = mean_squared_error(y_test, y_pred)
        mlflow.log_metric("precision", precision)

    mlflow.end_run()

2023/07/12 18:36:19 INFO mlflow.tracking.fluent: Experiment with name 'expRetrain' does not exist. Creating a new experiment.
First time execution

```

Line 1, Column 1    ★ 16 Lines    Python    Spaces 2

After a few seconds, you will see the following message “Deploying Model...” repeated several times, and the bottom command bar will be red.

The screenshot shows a Jupyter Notebook environment. On the left, there is a file browser displaying a directory structure for a project named "Telco Churn". The files listed include `0_bootstrap.py`, `0_bootstrap.ipynb`, `1_trainStrategy.ipynb`, `2_get_champion.py`, `2_get_champion.ipynb`, `_best_model_serve.py`, `_model_viz.py`, `cdsw-build.sh`, `churnexplainer.py`, `flask`, `images`, `image.yaml`, `models`, `README.md`, `requirements.txt`, and `visuals.json`.

The main area contains two tabs: "2\_get\_champion.py" and "Untitled Session".

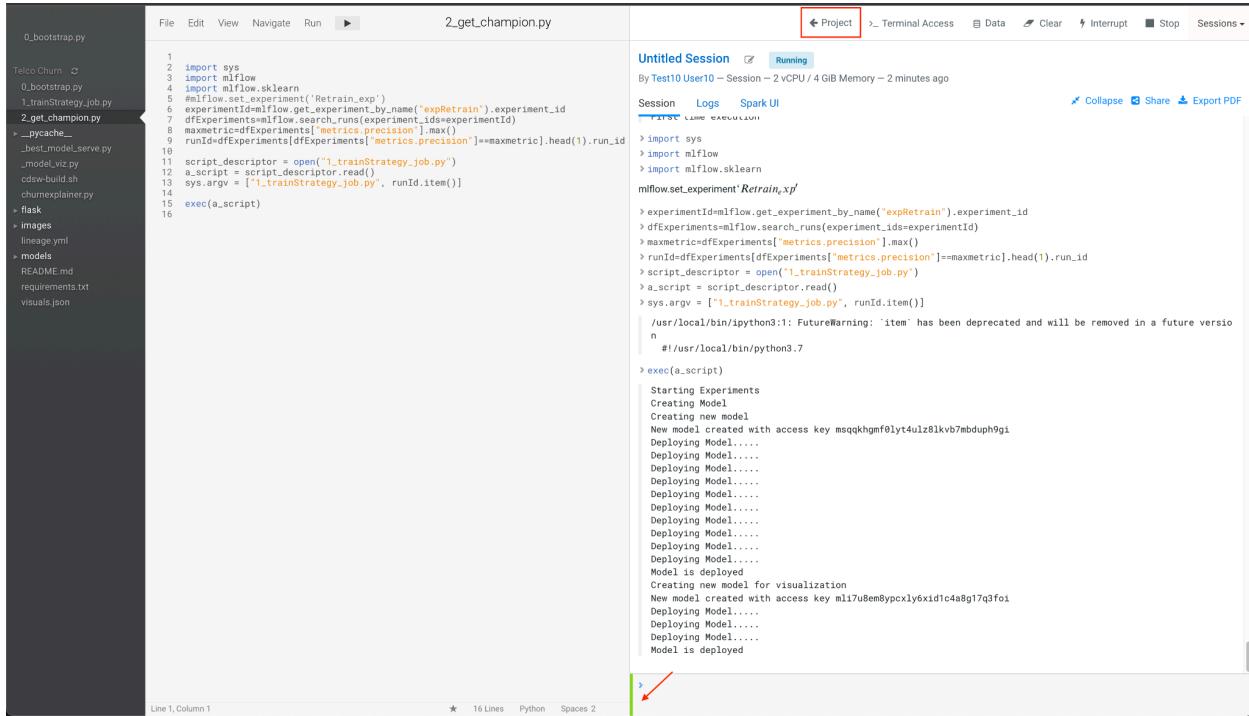
The "2\_get\_champion.py" tab shows the following Python code:

```
1 import sys
2 import mlflow
3 import sklearn
4 mlflow.set_experiment('Retrain.exp')
5 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
6 experimentId=mlflow.get_experiment_by_name("expRetrain").experiment_id
7 dfExperiments=dfExperiments[dfExperiments['metrics.precision'].max()]
8 runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
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
```

The "Untitled Session" tab shows the output of running the script. It includes a warning about the deprecation of the `item` method, followed by a series of messages indicating the deployment process:

```
Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lyt4ulz8lkvb7mbduph9g1
Deploying Model....
Model is deployed
Creating new model for visualization
New model created with access key mli7u8em8ypcxly6xid1c4a8g17q3fo1
Deploying Model....
Deploying Model....
```

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.



The screenshot shows a Jupyter Notebook environment. On the left, there is a file tree with the following structure:

```

0_bootstrap.py
TelcoChurn
  0_bootstrap.py
  1_trainStrategy.job.py
  2_get_champion.py
> _pycache_...
> _best_model.serve.py
> _model_viz.py
> cdsw-build.sh
> churn_explainer.py
> flask
> images
> lineage.yml
> models
> README.md
> requirements.txt
> visualis.json

```

In the center, there is a code editor window titled "2\_get\_champion.py" containing Python code. The code imports sys, miflow, and sklearn, and performs some data processing and model deployment logic.

To the right of the code editor is a session area titled "Untitled Session" which is "Running". The session log shows the execution of the script, including the creation of new models and their deployment. The log ends with the message "Model is deployed".

The top right of the session area has a "Project" button, which is highlighted with a red arrow in the screenshot.

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**.

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

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 *No* 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.

The screenshot shows the Cloudera Machine Learning interface for a project named 'user010 / Telco Churn / Models / ModelOpsChurn\_user010'. The left sidebar has a 'Models' section selected. The main area displays a 'Test Model' interface. In the 'Input' field, the following JSON is shown:

```
{
  "onlinesecurity": "No",
  "multiplelines": "No",
  "internetservice": "DSL",
  "seniorcitizen": "No",
  "techsupport": "No"
}
```

A red box highlights the 'Test' button. Below it, the 'Result' section shows a status of 'success' with a green dot. The 'Response' field contains the following JSON output:

```
{
  "model_deployment_crn": "crn:cdp:ml:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e48b728-bcff-4867-8a54-f83099c99355/32aa37d1-af8b-4225-a86d-4ca5",
  "prediction": {
    "probability": 0.5555555555555556
  },
  "uuid": "95a97cf3-36d3-459e-9372-b2b51334ca63"
}
```

The 'Replica ID' field shows 'modelopschurn-user010-19-14-6c5d7947ff-52kzg'. On the right side, there are details about the model resources:

- Comment: Initial revision.
- Runtime Image: Python 3.7 (Standard)
- File: \_best\_model\_serve.py
- Function: explain
- Model Resources:
  - Replicas: 1
  - Total CPU: 1 vCPUs
  - Total Memory: 2.00 GiB

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

**End of Lab**