

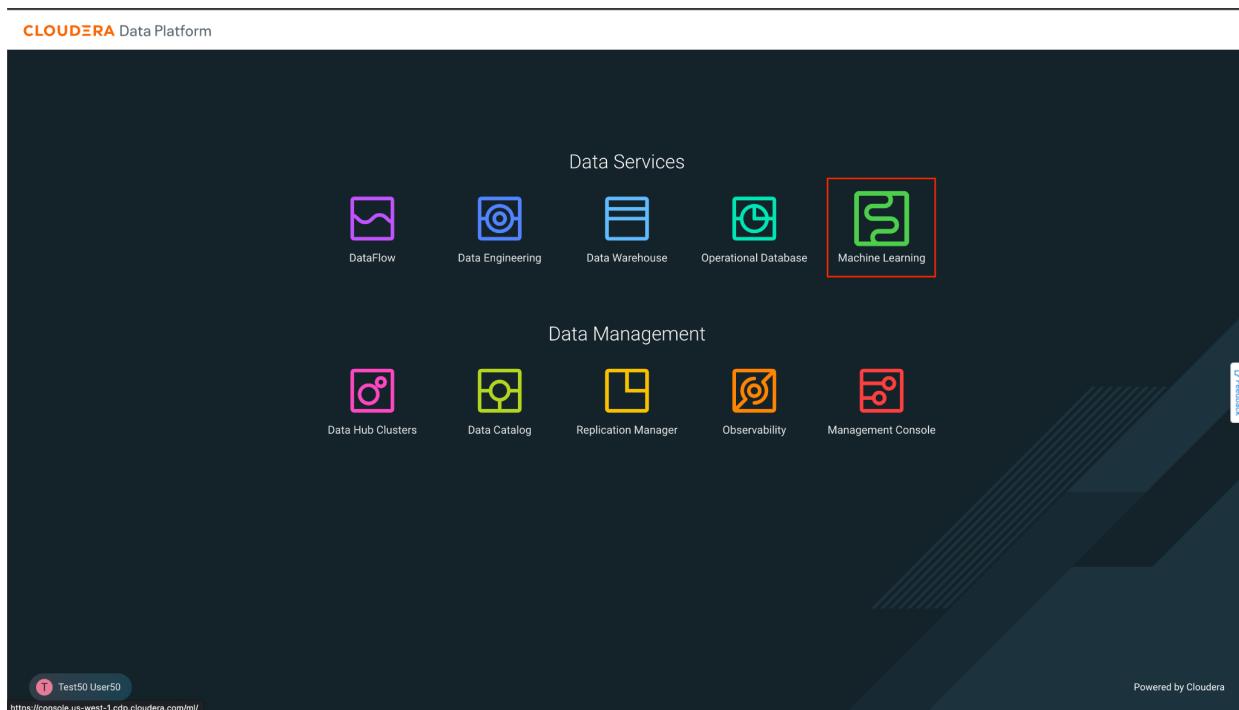
Data Lifecycle en CDP Public Cloud

Laboratorio Machine Learning

Objetivos:

- Entrenar un modelo para pronosticar si un cliente dejará de consumir productos
- Desplegar/exponer el modelo como REST API

1. Hacer clic en Data Warehouse desde el Home de CDP PC:



2. Pantalla de inicio de Machine Learning. Hacer clic en el único Workspace que aparece.

The screenshot shows the 'Machine Learning Workspaces' page. On the left, there's a sidebar with icons for Workspaces, Workspace Backups, Model Registries, Help, and User information (Test50 User50). The main area has a header 'Machine Learning Workspaces' with search and filter options. A table lists workspaces, with one row highlighted: 'Ready' status, version '2.0.38', workspace 'ssa-cml-workspace' (which is outlined in red), environment 'ssa-hol', region 'unknown', creation date '07/07/2023 11:42 PM CEST', cloud provider 'aws' (AWS), and actions. At the bottom, it says 'Displaying 1 - 1 of 1'.

3. Una vez en el Workspace, deberá ver la siguiente pantalla. Aquí se muestran los proyectos que has creado. Es momento de crear un nuevo proyecto. Hacer clic en el botón color azul **New Project**.

The screenshot shows the 'Projects' page. The sidebar includes 'Projects' (selected), Sessions, Experiments, Models, Jobs, Applications, User Settings, AMPS, Runtime Catalog, Site Administration, and Learning Hub. The main area has a header 'Projects' with a 'View Resource Usage Details' link and a 'Project quick find' bar. Below is a search bar, scope dropdown ('My Projects'), and creator dropdown ('All'). A central message says 'You currently don't have any projects' with a small cloud icon. A descriptive text explains that projects hold code, configuration, and libraries needed for reproducible runs. At the bottom right is a blue 'New Project' button (outlined in red). The footer shows 'Workspace: ssa-cml-workspace' and 'Cloud Provider: aws (AWS)'.

4. Introducir la siguiente información para crear un nuevo proyecto:

Project Name: Telco Churn

Project Visibility: Private

Initial Setup, selecciona Git

En el campo de texto debajo de HTTPS, ingresar la url del repo git:

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

Mantener el resto de las configuraciones igual. Hacer clic en el botón **Create Project**

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

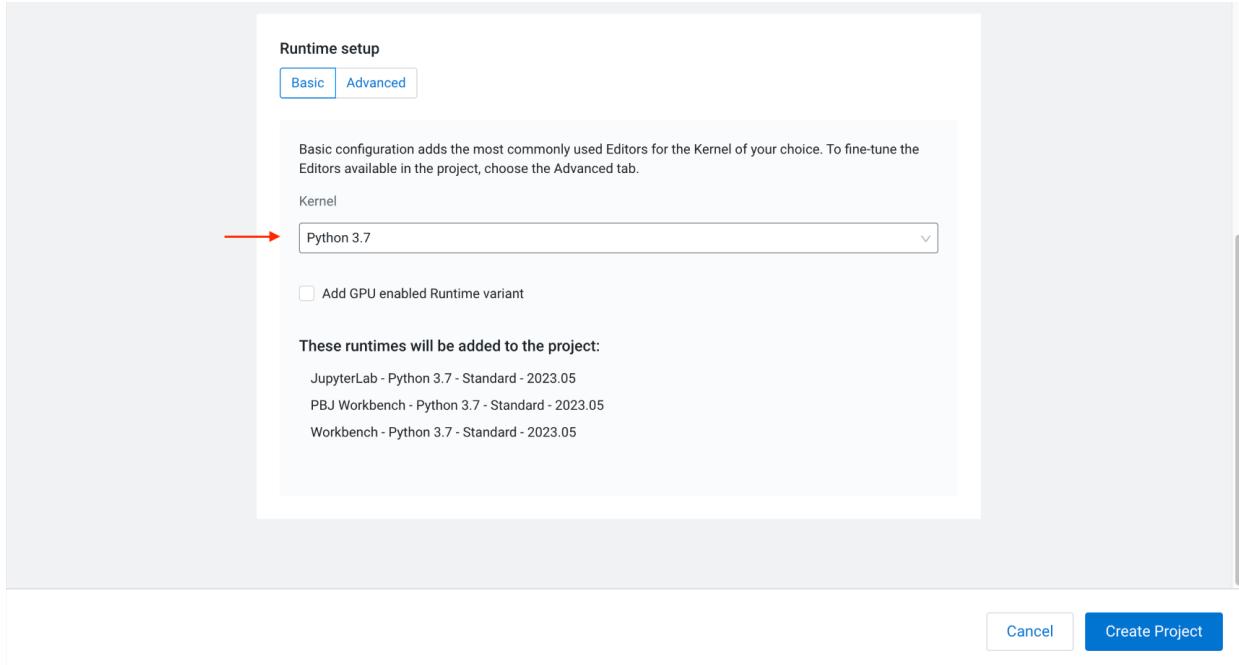
Provide the Git URL of the project to clone. Select the option that applies to your URL access.
 HTTPS SSH
https://github.com/campossalex/TelcoChurn

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

Runtime setup
Basic Advanced

Cancel Create Project

Asegúrese de seleccionar **Python 3.7** en la sección de Kernel. Hacer clic en el botón **Create Project**



5. Una vez creado el proyecto, deberás ver la siguiente pantalla:

Models, desplegar y administrar modelos como API REST para servir predicciones.

Jobs, automatizar y orquestar la ejecución de las cargas de trabajo analíticas batch

Files, activos que hacen parte del proyecto, como ficheros, scripts y código

Este proyecto consiste en la ejecución de tres scripts. La forma de ejecución es por medio de una sesión, que es la asignación de cómputo aislado para cada usuario. Para esto, deberás hacer clic en el botón de color azul **New Session**, ubicado en la parte superior derecha.

The screenshot shows the Cloudera Machine Learning Workbench interface. On the left, there's a sidebar with navigation links: All Projects, Overview, Sessions, Data, Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main area is titled 'Telco Churn'. It has sections for 'Models' (with a note about no models yet), 'Jobs' (with a note about no jobs yet), and 'Files'. The 'Files' section contains a table listing various Python files and their details:

Name	Size	Last Modified
flask	-	a few seconds ago
images	-	a few seconds ago
models	-	a few seconds ago
0_bootstrap.py	1.95 kB	a few seconds ago
1_trainStrategy_job.py	18.63 kB	a few seconds ago
2_get_champion.py	508 B	a few seconds ago
_best_model_serve.py	2.74 kB	a few seconds ago
_model_viz.py	4.21 kB	a few seconds ago
cdsw-build.sh	44 B	a few seconds ago
chumexplainer.py	6.69 kB	a few seconds ago
lineage.yml	610 B	a few seconds ago
README.md	11.97 kB	a few seconds ago
requirements.txt	197 B	a few seconds ago
visuals.json	281.07 kB	a few seconds ago

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

6. Asegurarse de habilitar Spark, marcando el check correspondiente, y hacer clic en el botón **Start Session**.

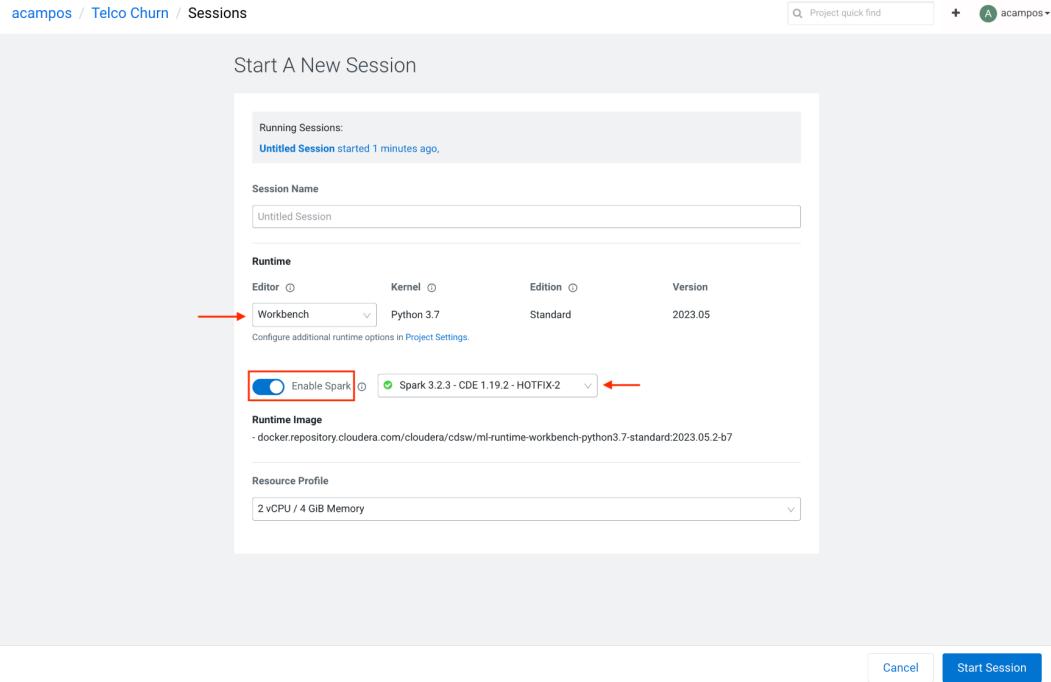
6. Cuando iniciar la sesión, asegúrese de::

Seleccionar **Workbench** en la opción Editor.

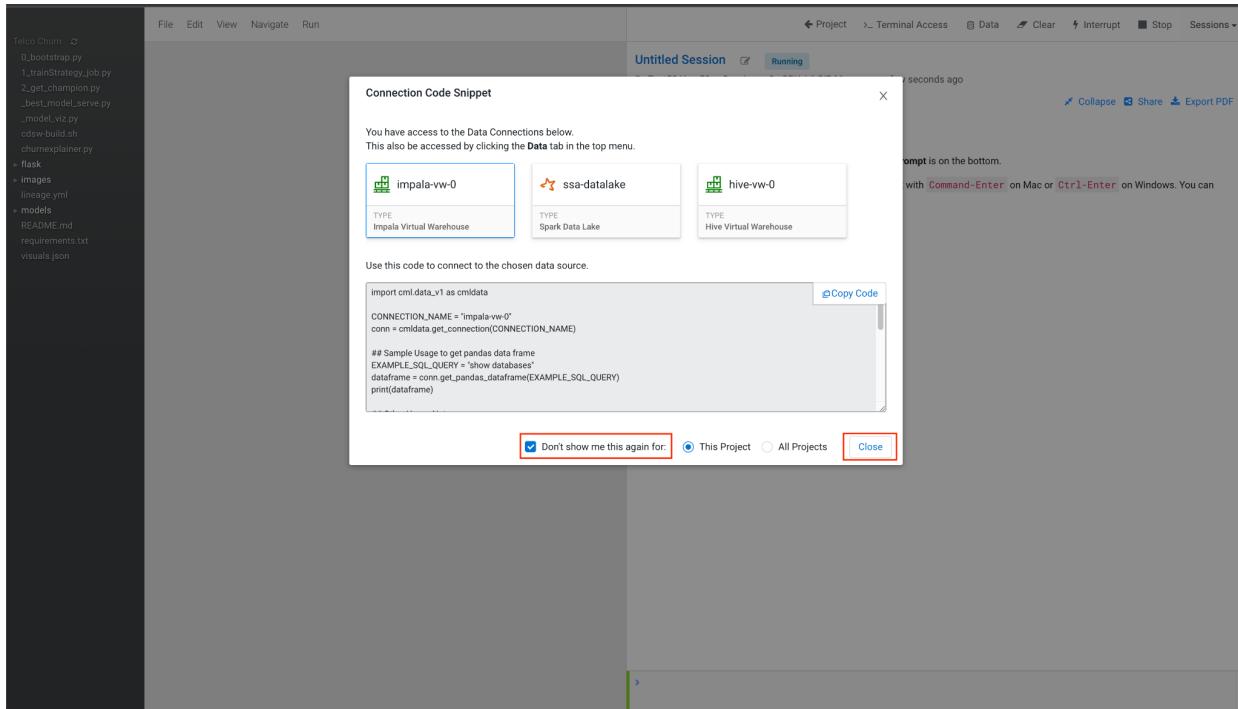
Habilitar **Spark**, marcando la opción correspondiente.

Seleccionar **Spark 3.2.x** en la opción de versión de Spark.

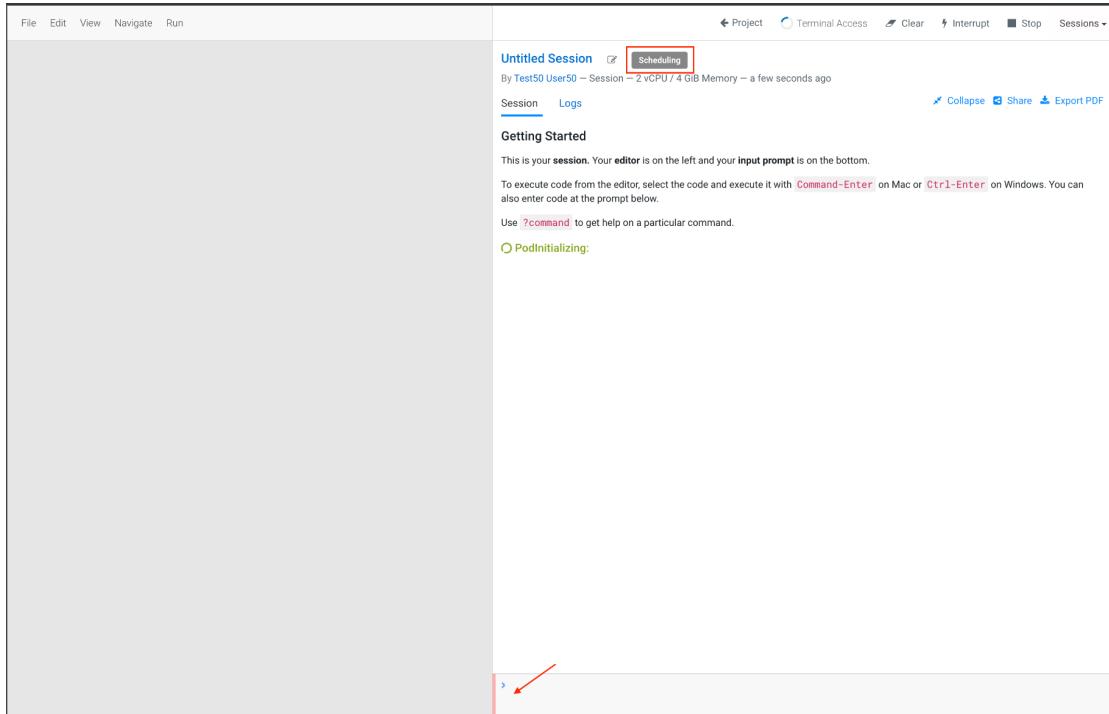
Hacer click en **Start Session**



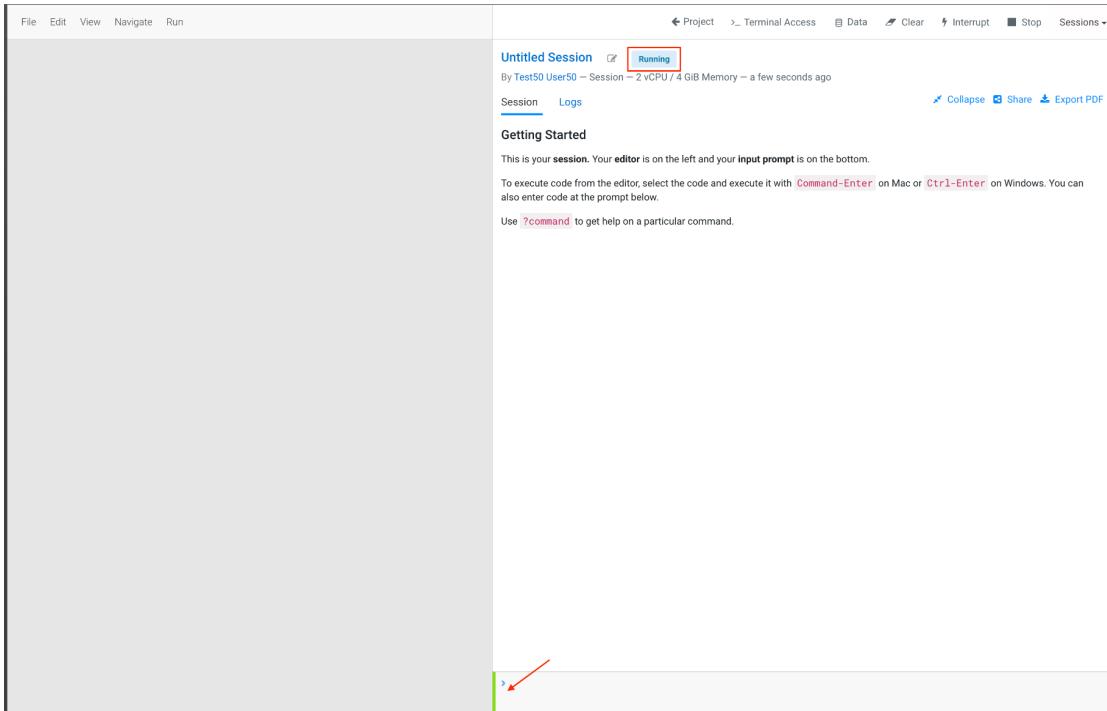
7. Al iniciar la sesión por primera vez, preguntará si deseas utilizar alguna conexión de datos. Este proyecto no necesita de este tipo de conexión. Marcar el check de **Don't show me this again**, y después hacer clic en el botón **Close**.



8. El editor/notebook ubicado en el lado derecho de la ventana, estará en estado **Scheduling**, y la barra de comando inferior parpadeando en color rojo. Esto significa que CML está alocando cómputo para su sesión.



Pasado algunos segundos, el estado cambia a **Running**, y la barra de comando a color verde. Esto significa que la sesión está lista para ejecutar código.



9. El primer script/código para ejecutar es **0_bootstrap.py**. Este código en Python configura las librerías requeridas para el proyecto e integración con el Lakehouse. Selecciona (solo un clic) el fichero en la barra ubicada en el lado izquierdo de la interfaz, esto hará que el código aparezca en el editor. Una vez seleccionado el fichero, hacer clic en el botón para ejecutar el código.

```

File Edit View Navigate Run  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 # /opt/ibm/BigInsight-UserC-Telco-Customer-Churn-.csv into /data湖ake/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("CDSW_API_URL").split(
31     "[")[-1] + ":" + os.getenv("CDSW_PORT")
32 USERNAME = os.getenv("CDSW_USERNAME")
33 USERPASS = os.getenv("CDSW_PASSWORD")
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_PROJECT")
36
37 # Instantiates 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
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

```

Al empezar la ejecución, verás salida del código en el lado derecho de la interfaz, y la barra de comando inferior cambia a color rojo, indicando que está ocupado.

The screenshot shows a Jupyter Notebook interface with the following details:

- File Tree:** Shows a directory structure for a project named "Telco Churn".
- Code Cell:** Contains Python code for a bootstrap file, specifically "0_bootstrap.py". The code sets up environment variables, installs requirements, and configures a CMLBootstrap instance.
- Output Panel:** Displays the output of the code execution, including logs from a session named "Untitled Session". The logs show the command being run: "pip3 install -r requirements.txt".
- Status Bar:** At the bottom, it says "Running" and has a red background, indicating the cell is currently executing.
- Bottom Buttons:** Includes "Project", "Terminal Access", "Data", "Clear", "Interrupt", "Stop", and "Sessions".

La barra de comando en color verde indica que la ejecución del código ha sido finalizada. Este código se demora entre 3 a 4 minutos en ejecutarse.

```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1_trainStrategy_job.py
2_get_champion.py
best_model_serve.py
model_viz.py
cdsw-build.sh
churnexplainer.py
> flask
> images
lineage.yml
> models
README.md
requirements.txt
visuals.json

Line 1, Column 1      * 57 Lines  Python  Spaces 2

```

```

Untitled Session  Running
By Test50 User50 - Session - 2 vCPU / 4 GiB Memory - a few seconds ago
Session Logs
Create the directories and upload data
> from cmlbootstrap import CMLBootstrap
> 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%b%Y%H%M%S")
Set the setup variable needed by CMLBootstrap
> HOST = os.getenv('CDSW_API_URL').split(
...    (':')[0] + '/' + os.getenv('CDSW_DOMAIN'))
> USERNAME = os.getenv('CDSW_PROJECT_URL').split(
...    ('/')[1] + args.username + '_vdbiba')
> API_KEY = os.getenv('CDSW_API_KEY')
> PROJECT_NAME = os.getenv('CDSW_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
...             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

```

10. El segundo script/código para ejecutar es **1_trainStrategy_job.py**. Este código en Python creará el Experimento para ejecutar el modelo con tres hiper parámetros distintos e registra la precisión. Selecciona (solo un clic) el fichero en la barra ubicada en el lado izquierdo de la interfaz, esto hará que el código aparezca en el editor. Una vez seleccionado el fichero, hacer clic en el botón para ejecutar el código. Una vez terminada la ejecución (1 minuto aproximadamente), hacer clic en el botón **Project**, ubicado en la barra superior derecha de la sesión para regresar a la página inicial del proyecto.

The screenshot shows the Cloudera Machine Learning interface. On the left, the file tree for the 'Telco Churn' project is visible, with a red arrow pointing to the file '1_trainStrategy.job.py'. The main area contains two panes: a code editor on the left and a terminal session on the right. The code editor shows the Python script '1_trainStrategy.job.py' with several imports and logic for training a Random Forest model using mlflow. The terminal session titled 'Untitled Session' shows the execution of the script, including setting up the experiment, training the model, and calculating precision metrics.

```

File Edit View Navigate Run Project Terminal Access Data Clear Interrupt Stop Sessions
Telco Churn
1_bootstrap.py
1_trainStrategy.job.py
2_get_champion.py
3_pycache_
4_bst_model.serve.py
5_model.viz.py
6_swift.sh
7_churnexplainer.py
8_flask
9_images
10_imagine.yml
11_models
12_README.md
13_requirements.txt
14_visuals.json
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 mlflow import MlflowClient
23 # Set the setup variables needed by CMLBootstrap
24 HOST = os.getenv("CDSW_API_URL").split(
25     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
26 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
27     "/" )[6] # args.username = "vdubla"
28 API_KEY = os.getenv("CDSW_API_KEY")
29 PROJECT_NAME = os.getenv("CDSW_PROJECT")
30
31 # Instantiates API Wrapper
32 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
33
34 uservariables=cml.get_user()
35 if uservariables['username'][~-3:] == '0':
36     DATABASE = "user"+uservariables['username'][~-3:]
37 else:
38     DATABASE = uservariables['username']
39     DATABASE = 'acampos'
40
41
42 runtimes=cml.get_runtimes()
43 runtimes=runtimes['runtimes']
44 runtimesdf = pd.DataFrame.from_dict(runtimes, orient='columns')
45 runtimesdf['runtimesdf'].loc[(runtimesdf['editor'] == 'Workbench') & (runtimesdf['ke
46 id_rf']==runtimesdf['values'][0])
47
48 spark = SparkSession(
49     .builder
50     .appName("PythonSQL")
51     .master("local[1]")
52     .config("spark.sql.extensions", "org.apache.iceberg.spark.extensions.IcebergS
53     config("spark.sql.catalog.spark_catalog", "org.apache.iceberg.spark.SparkCa
54     .config("spark.sql.catalog.local", "org.apache.iceberg.spark.SparkCatalog") \
55     .config("spark.sql.catalog.local.type", "hadoop") \
56     .config("spark.sql.catalog.spark_catalog.type", "hive") \
57     .getOrCreate()
58
59
60 # **Note:**
61 # Our file isn't big, so running it in Spark local mode is fine but you can add
62
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_."+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

```

11. Una vez de vuelta en la interfaz principal del proyecto, hacer clic en la opción **Experiments**, del menú izquierdo, y posteriormente en **expRetrain** en el listado de Experiments que aparece.

The screenshot shows the 'Experiments' page in the Cloudera Machine Learning interface. The sidebar on the left has a red arrow pointing to the 'Experiments' option. The main area displays a table of experiments. One experiment, 'expRetrain', is highlighted with a red box. The table columns include Name, Creator, Created At, and Last Updated. The 'Last Updated' column shows the most recent update time for each experiment.

Name	Creator	Created At	Last Updated
expRetrain	Test10 User10	07/12/2023 8:36 PM	-

12. En esta pantalla verás las tres ejecuciones del experimento. Fíjense en la última columna, donde aparece **precision**. Esto es la precisión que está entregando cada hiper parámetro.

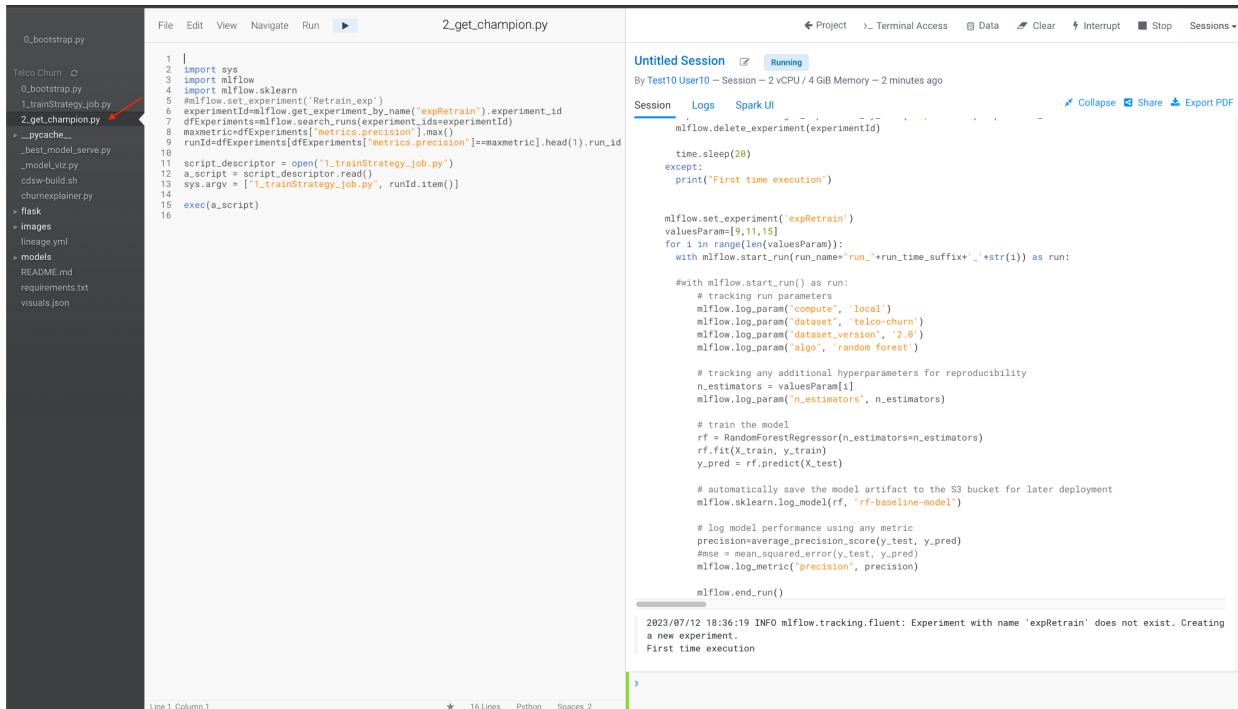
The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with navigation links: All Projects, Overview, Sessions, Data, Experiments (which is selected and highlighted in green), Models, Jobs, Applications, Files, Collaborators, and Project Settings. At the bottom of the sidebar, it says "2.0.38-b125". The main content area has a header "user010 / Telco Churn / Experiments / expRetrain". Below the header, there's a section titled "Experiment" with a "BETA" badge. It displays the Experiment Name (expRetrain), Experiment ID (pdj8-a6kp-bh2r-dehf), and Artifact Location (/home/cdsu/experiments/pdj8-a6kp-bh2r-dehf). There's also a "Notes" link. The next section is "Runs (3)", which contains a table with three rows of data. The table has columns for Status, Start Time, Run Name, Duration, User, Source, Version, Models, algo, compute, dataset, and precision. The runs are as follows:

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

13. Regresemos a la sesión para ejecutar el último código. Como las sesiones se ejecutan en contenedores de Kubernetes, es muy fácil regresar a donde estábamos. Hacer clic en la opción **Sessions** del menú izquierdo, y posteriormente en la única sesión que aparecerá en el listado. Si no agregaste un nombre a tu sesión al momento de iniciarla (paso 6), se debe llamar *Untitled Session*.

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with various project management options like Overview, Sessions (which is highlighted with a red arrow), Data, Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. Below the sidebar, it says 'Help' and '2.0.38-b125'. At the bottom, it indicates the workspace is 'ssa-cml-workspace' and the cloud provider is 'AWS (AWS)'. The main area is titled 'user010 / Telco Churn Sessions'. It has a search bar, a 'Creator' dropdown set to 'All', and a 'Show Running Only' toggle switch. A table lists sessions: one running session named 'Untitled Session' (Python 3.7 Workbench Standard) created by 'Test10 User10' at 07/12/2023 8:35 PM, which has been running for 1m 43s. There are buttons for 'Edit', 'Stop', and 'Delete'. The table includes columns for Status, Session, Kernel, Creator, Created At, and Duration. At the bottom right, it says 'Displaying 1 - 1 < 1 > 25 / page'.

14. El tercer y último script/código para ejecutar es **2_get_champion.py**. Este código en Python toma el hiper parámetro de la ejecución del Experiment con mejor precisión y despliega dos Modelos en REST API, uno para integrarse en Data Visualization y otro para uso unitario para llamadas. Selecciona (solo un clic) el fichero en la barra ubicada en el lado izquierdo de la interfaz, esto hará que el código aparezca en el editor. Una vez seleccionado el fichero, hacer clic en el botón para ejecutar el código.



```

File Edit View Navigate Run ▶ 2_get_champion.py
Untitled Session [Running]
By Test10User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes 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:
        # 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")

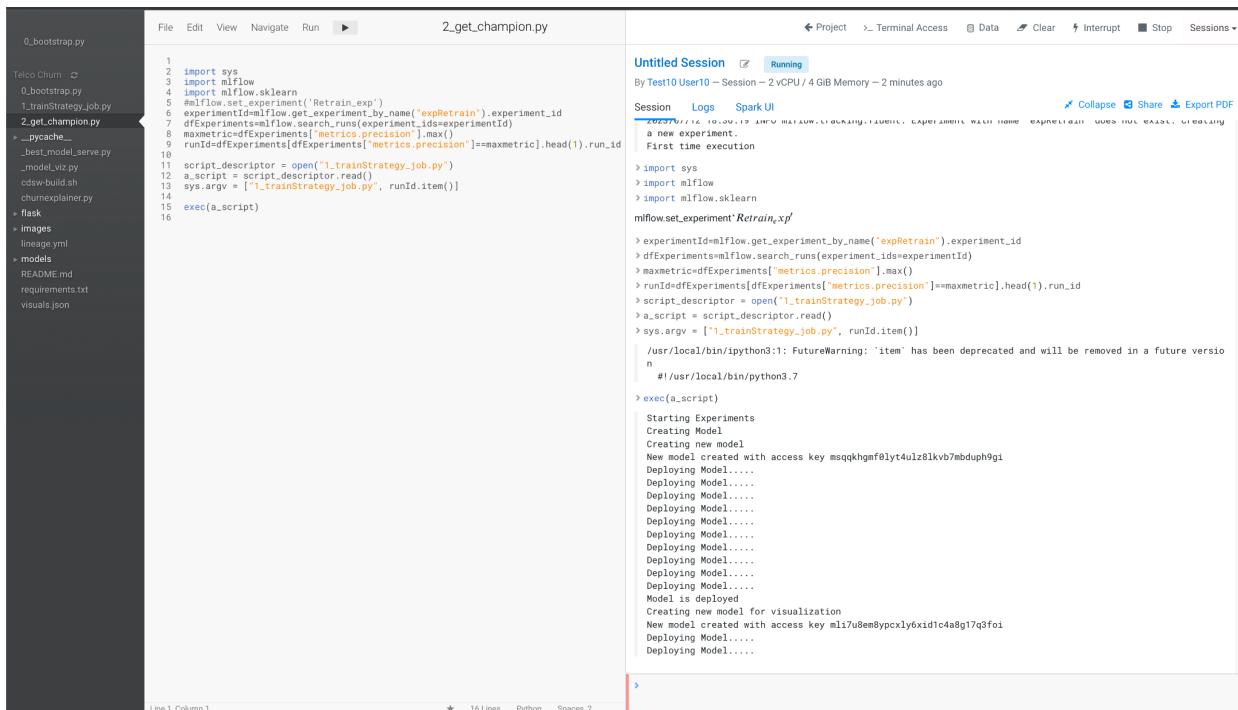
        # log model performance using any metric
        precision=average_precision_score(y_test, y_pred)
        #mse = mean_squared_error(y_test, y_pred)
        mlflow.log_metric("precision", precision)

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

```

Después de unos segundos, verás el siguiente mensaje “Deploying Model...” repetirse varias veces, y la barra de comandos inferior estará en color rojo.



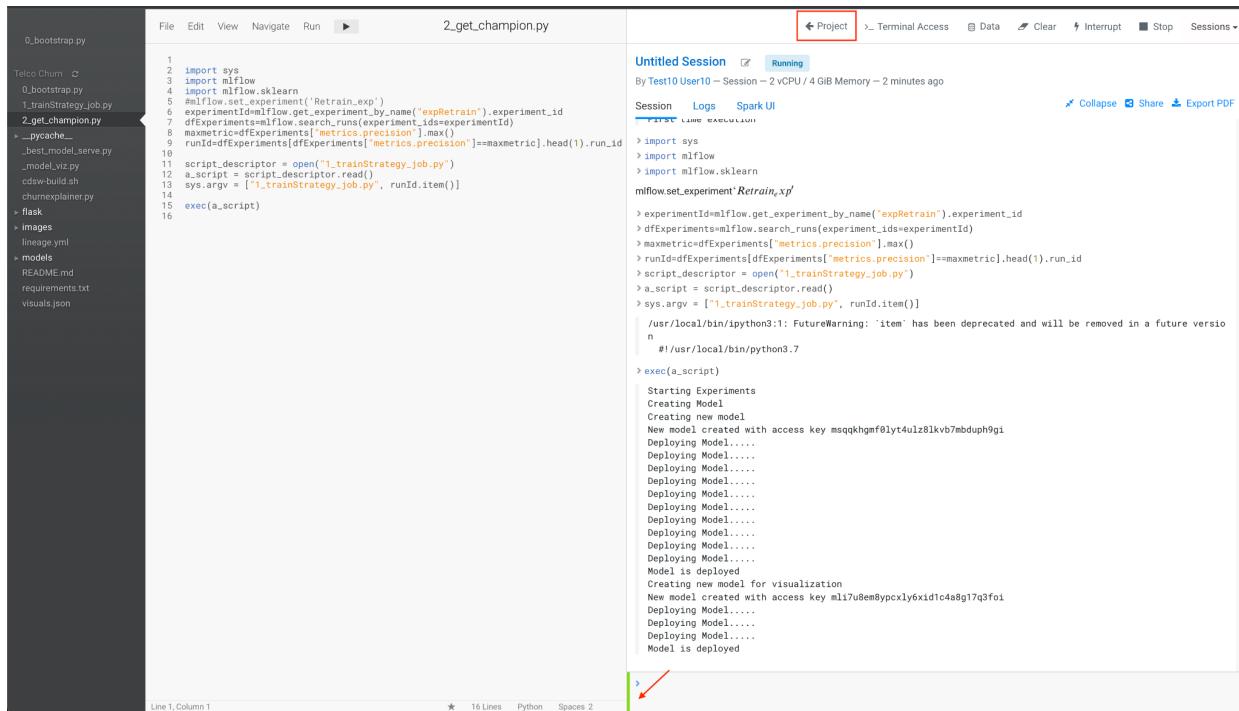
```

File Edit View Navigate Run ▶ 2_get_champion.py
Untitled Session [Running]
By Test10User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago
Session Logs Spark UI
2023/07/12 18:50:17 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()]
> exec(a_script)
Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lyt4ulz8lkvb7mbduph9gi
Deploying Model.....
Model is deployed
Creating new model for visualization
New model created with access key mli7u0em8ypcxl6xidlc4s8g17q3foi
Deploying Model.....
Deploying Model.....

```

Pasados unos 2 minutos, el último mensaje deberá ser “Model is deployed”, y la barra estará en color verde. Significa que el despliegue de los dos Models está finalizado. Hacer clic en el

botón **Project**, ubicado en la barra superior derecha de la sesión para volver a la página inicial del proyecto.

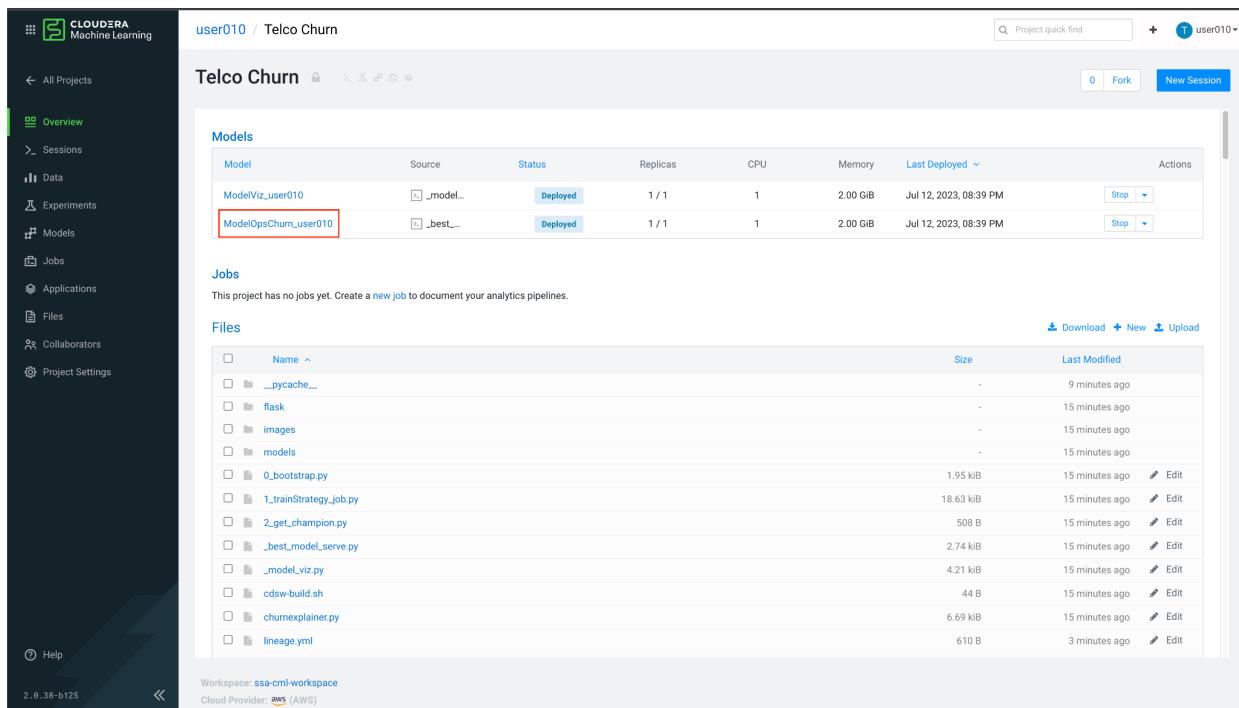


```

File Edit View Navigate Run ▶ Project Untitled Session Running
By Test010User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago
Session Logs Spark UI
FASTER, LARGER, EASIER TOOLS
> 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)
> dfExperiments=dfExperiments[['metrics.precision']].max()
> runId=dfExperiments[dfExperiments['metrics.precision']==maxmetric].head(1).run_id
> 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()]
> exec(a_script)
Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lyt4ulz8lkvb7mbdph9gi
Deploying Model.....
Model is deployed
Creating new model for visualization
New model created with access key ml17u8em8ypcxlbyxid1c48g17q3fo1
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Model is deployed

```

15. Una vez en la página inicial del proyecto, verás los Models desplegados, que son dos. Hacer clic en el que empieza con el nombre **ModelOpsChurn**.



Model	Source	Status	Replicas	CPU	Memory	Last Deployed	Actions
ModelViz_user010	model...	Deployed	1 / 1	1	2.00 GiB	Jul 12, 2023, 08:39 PM	<button>Stop</button>
ModelOpsChurn_user010	best...	Deployed	1 / 1	1	2.00 GiB	Jul 12, 2023, 08:39 PM	<button>Stop</button>

Jobs
This project has no jobs yet. Create a [new job](#) to document your analytics pipelines.

Files

Name	Size	Last Modified
__pycache__	-	9 minutes ago
flask	-	15 minutes ago
images	-	15 minutes ago
models	-	15 minutes ago
0_bootstrap.py	1.95 kB	15 minutes ago
1_trainStrategy_job.py	18.63 kB	15 minutes ago
2_get_champion.py	508 B	15 minutes ago
best_model_serve.py	2.74 kB	15 minutes ago
_model_viz.py	4.21 kB	15 minutes ago
cdsw-build.sh	44 B	15 minutes ago
churnexplainer.py	6.69 kB	15 minutes ago
lineage.yml	610 B	3 minutes ago

Workspace: ssa-cmli-workspace
Cloud Provider: aws (AWS)

16. Aquí verás la información y configuración del Model en la pestaña Overview.

The screenshot shows the Cloudera Machine Learning interface with the following details:

- Project:** user010 / Telco Churn / Models / ModelOpsChurn_user010
- Overview Tab:** Active
- Model Details:**
 - Source: Code
 - Model Id: 19
 - Model CRN: cm:cdpm:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e4ab728-bcf1-4867-8a54-f8309c99355/1a9eb299-1c63-45f2-b60d-9a705aefba5#
 - Deployment Id: 14
 - Deployment CRN: cm:cdpm:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e4ab728-bcf1-4867-8a54-f8309c99355/32aa37d1af8b4225-a86d-4ca5c60f3109
 - Build Id: 14
 - Build CRN: cm:cdpm:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e4ab728-bcf1-4867-8a54-f8309c99355/197c0ed8b00e-4354-b63a-746af64e75e8
 - Deployed By: user010
 - 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
- Description:** Explain a given model prediction
- Sample Code:**
 - Shell: curl -H "Content-Type: application/json" -X POST https://modelservice.ml-1e5bb8eb-444.ssa-hol.yut-vbzg.cloudera.site/model -d '{"accessKey": "msqkhhmfolyt4u1z8lkb7mbduhp9gi", "request": {"streamingtv": "No", "monthlycharges": 70.35, "phoneservice": "No", "paperlesssubling": "No", "partner": "No", "onlinebackup": "No", "gender": "female", "contract": "Month-to-month", "totalcharges": 1397.475, "streamingmovies": "No", "deviceprotection": "No", "paymentmethod": "Bank transfer (automatic)", "tenure": 29, "dependents": "No", "onlinesecurity": "No", "multiplelines": "No", "internetservice": "DSL", "seniorcitizen": "No", "techsupport": "No"} }' or curl -H "Content-Type: application/json" -X POST https://modelservice.ml-1e5bb8eb-444.ssa-hol.yut-vbzg.cloudera.site/model?accessKey=msqkhhmfolyt4u1z8lkb7mbduhp9gi -d '{"request": {"streamingtv": "No", "monthlycharges": 70.35, "phoneservice": "No", "paperlesssubling": "No", "partner": "No", "onlinebackup": "No", "gender": "female", "contract": "Month-to-month", "totalcharges": 1397.475, "streamingmovies": "No", "deviceprotection": "No", "paymentmethod": "Bank transfer (automatic)", "tenure": 29, "dependents": "No", "onlinesecurity": "No", "multiplelines": "No", "internetservice": "DSL", "seniorcitizen": "No", "techsupport": "No"} }'
 - Python: {}
 - R: {}
- Test Model:**
 - Input: A JSON object representing user input parameters. One parameter is highlighted in blue: "onlinesecurity": "No".
 - Result: A JSON response from the model. The highlighted parameter "onlinesecurity" has a "probability" attribute of 0.9999999999999998.
- Environment:**
 - Workspace: ssa-cml-workspace
 - Cloud Provider: AWS (AWS)

Para probarlo y hacer una petición al modelo, desplazar la página hacia arriba, y hacer clic en botón **Test**, que tomará el valor en formato JSON que está en el campo **Input** y realizará la llamada al modelo. Lo que ves en el campo **Result** es la respuesta del modelo en formato JSON. Si deseas, puedes cambiar algunos de los parámetros del campo **Input** (por ejemplo, cambiar No por Yes), y volver a realizar la llamada al modelo, y observe el valor del atributo *probability* de la respuesta, para ver si hubieron cambios.

CLOUDERA Machine Learning

All Projects Overview Sessions Data Experiments Models Jobs Applications Files Collaborators Project Settings Help

2.0.38-b125

user010 / Telco Churn / Models / ModelOpsChurn_user010 / Overview

Sample Response {}

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

Test Model

Input

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

Test Reset

Result

Status success

Response

```
{
  "model_deployment_crn": "crn:cdp:ml:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e48b728-bcff-4867-8a54-f83899c99355/32aa37d1-af8b-4225-a86d-4ca5",
  "prediction": {
    "probability": 0.5555555555555556
  },
  "uuid": "95a97cf3-36d3-459e-9372-b2b51334ca63"
}
```

Replica ID modelopschurn-user010-19-14-6c5d7947ff-52kzg

Workspace: ssa-cml-workspace
Cloud Provider: AWS (AWS)