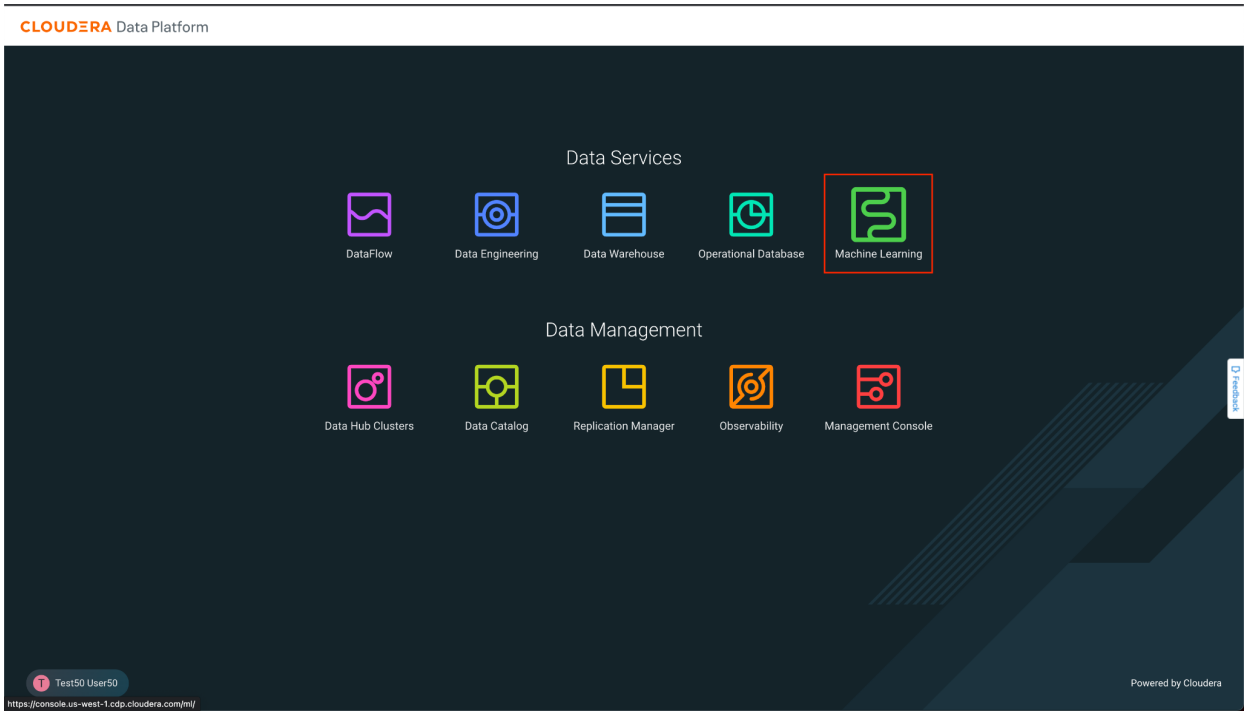
Data Lifecycle en CDP Public Cloud

Laboratorio Machine Learning

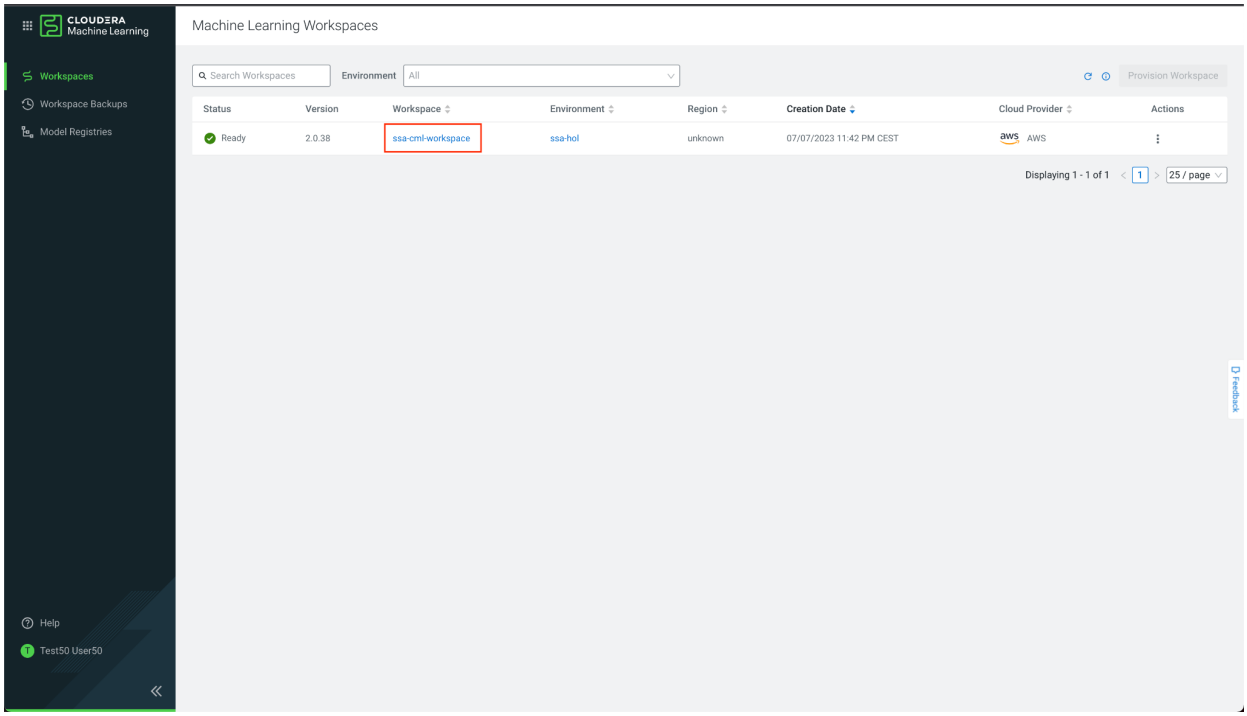
Objetivos:

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

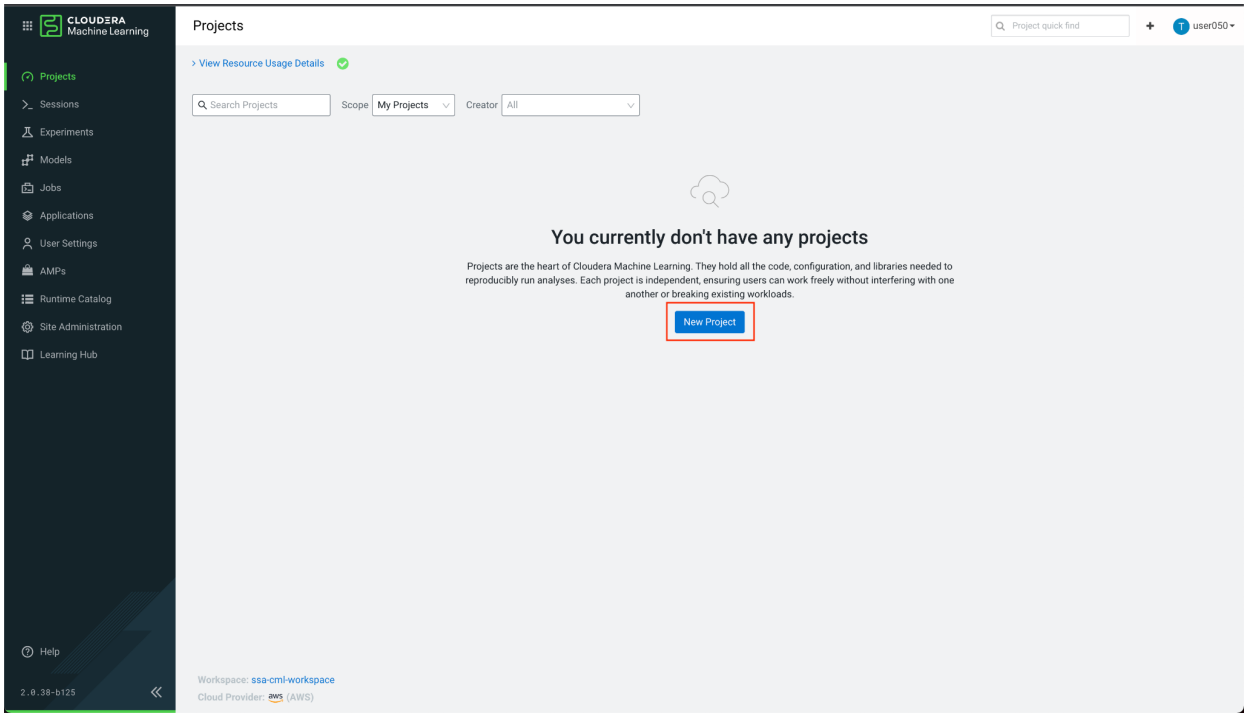
1. Hacer clic en Data Warehouse desde el Home de CDP PC:



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



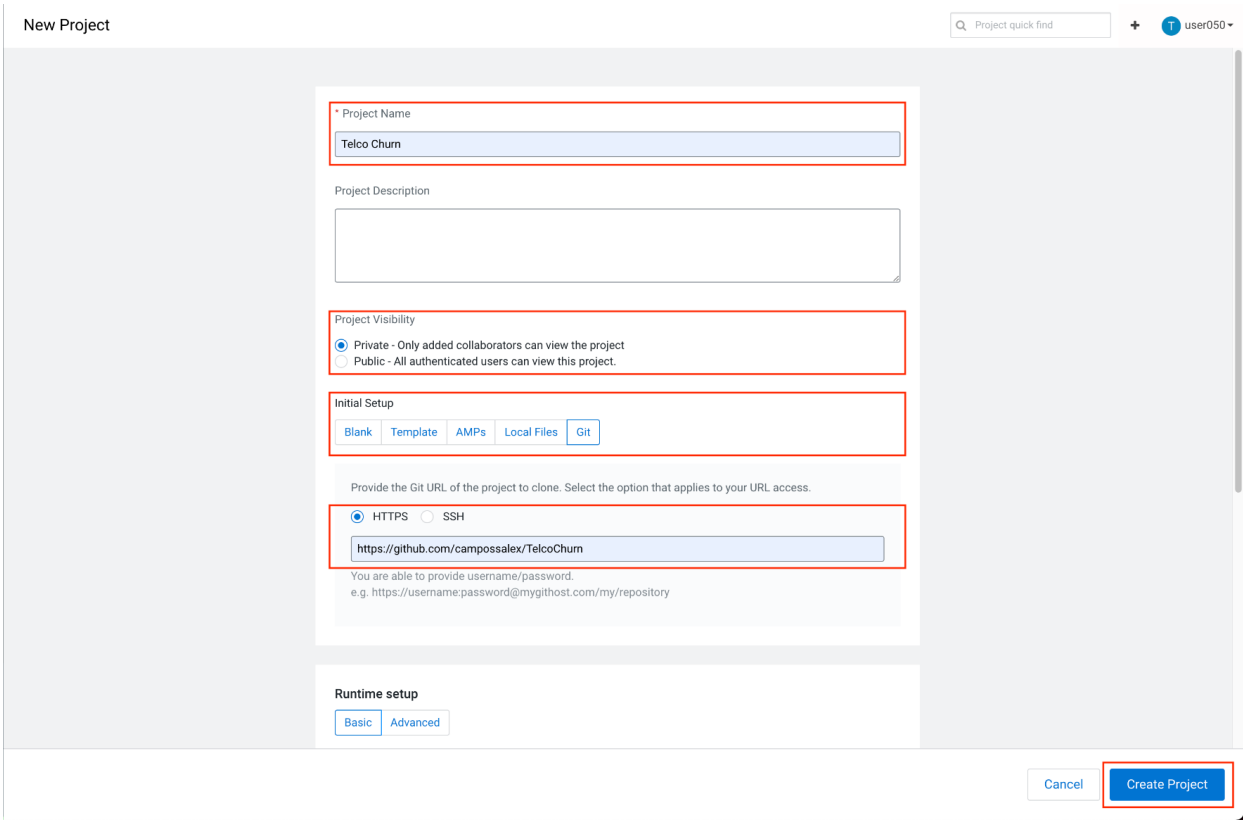
3. Una vez en el Workspace, deberá ver la siguiente pantalla. Aquí se muestran los proyectos que has creado. Es momento de crear un nuevo proyecto. Hacer clic en el botón color azul **New Project**.



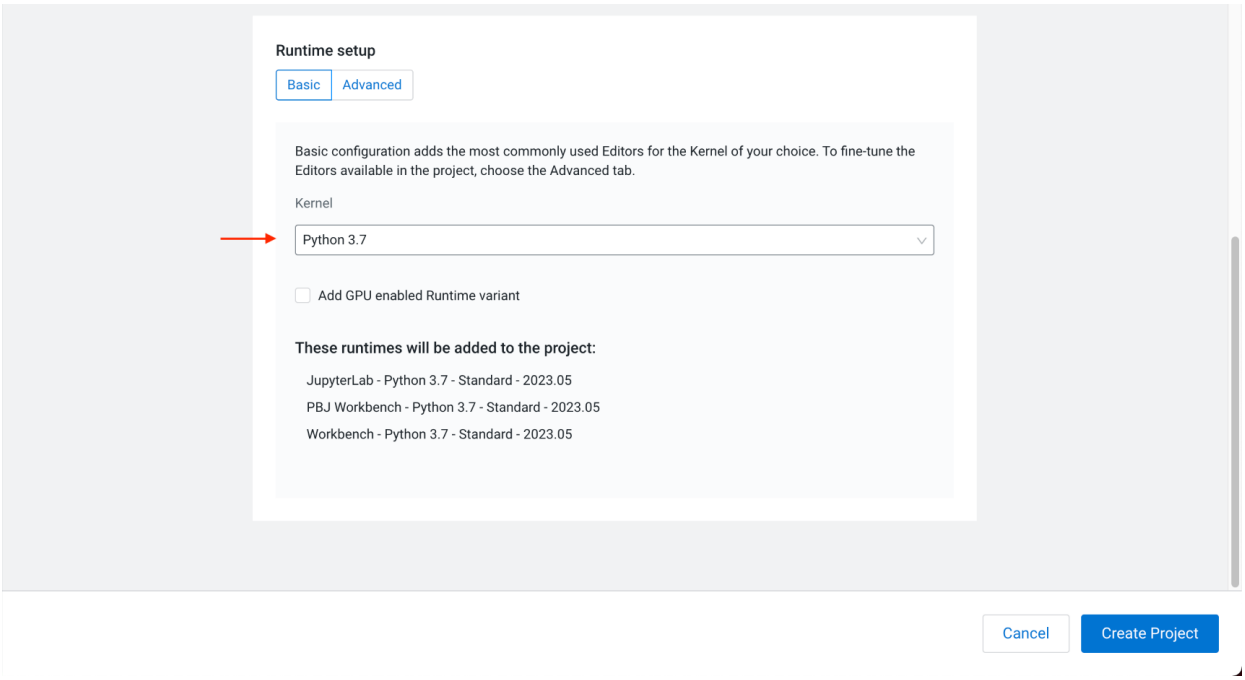
4. Introducir la siguiente información para crear un nuevo proyecto:

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**



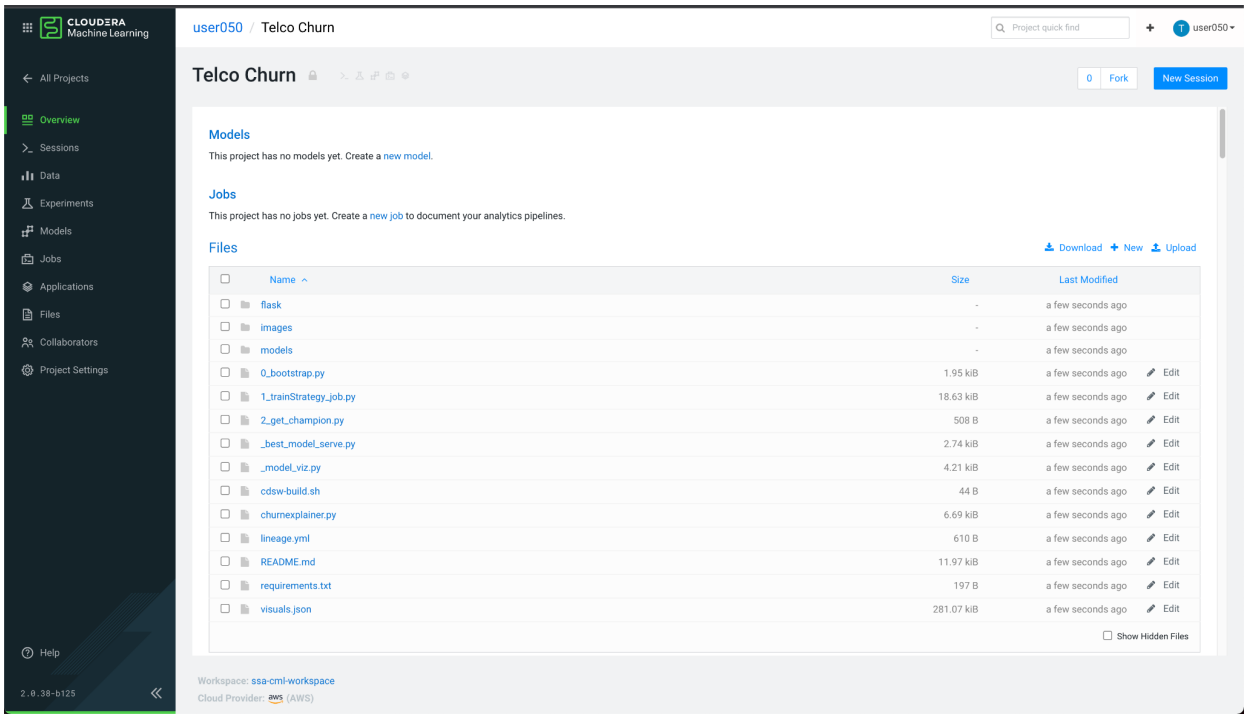
Asegúrese de seleccionar **Python 3.7** en la sección de Kernel. Hacer clic en el botón **Create Project**



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.

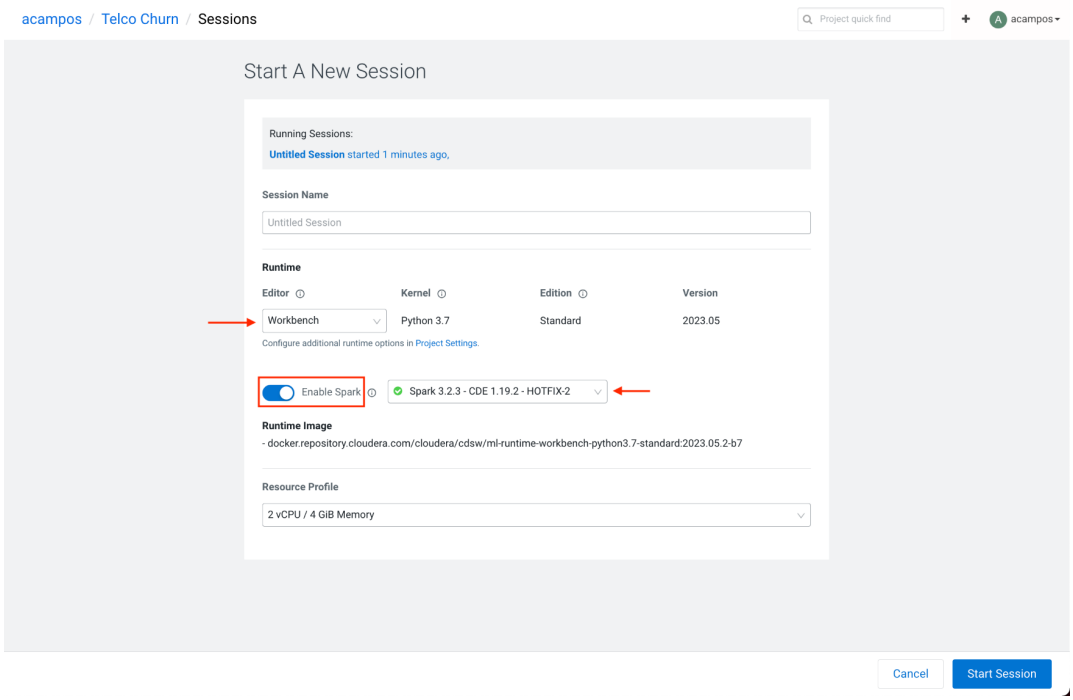


6. Asegurarse de habilitar Spark, marcando el check correspondiente, y hacer clic en el botón **Start Session**.

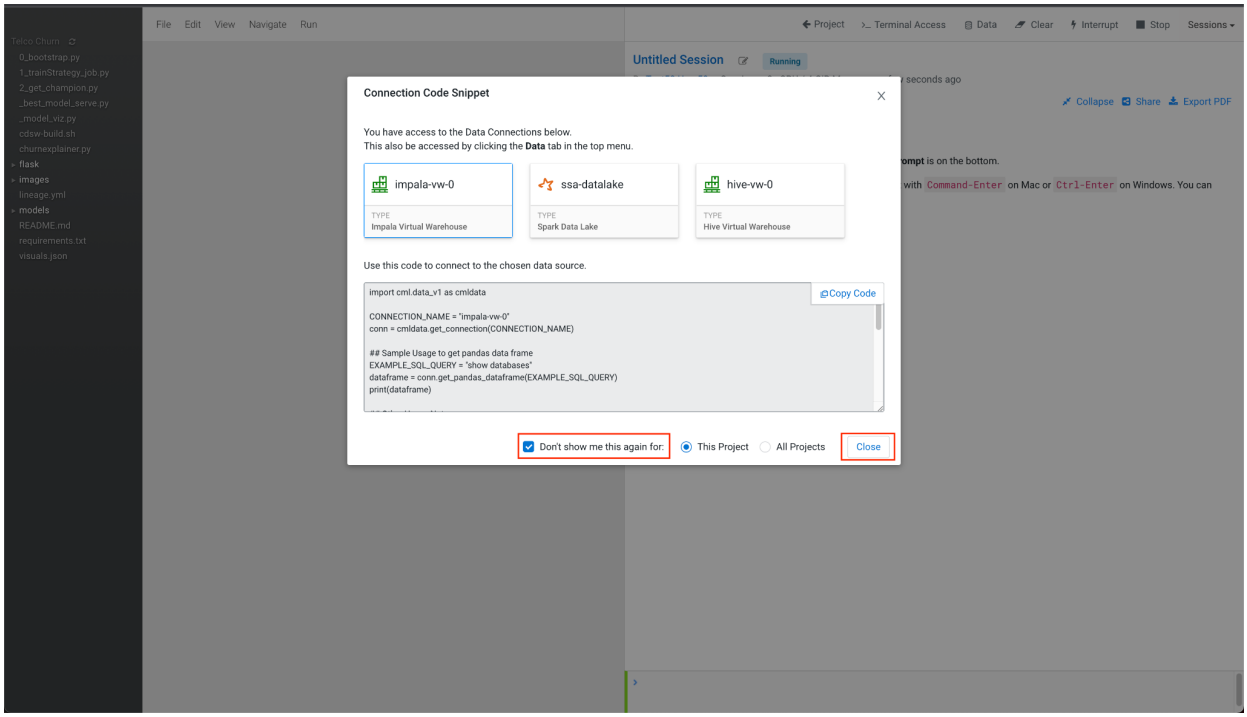
6. Cuando iniciar la sesión, asegúrese de::

Seleccionar **Workbench** en la opción Editor.
Habilitar **Spark**, marcando la opción correspondiente.
Seleccionar **Spark 3.2.x** en la opción de versión de Spark.

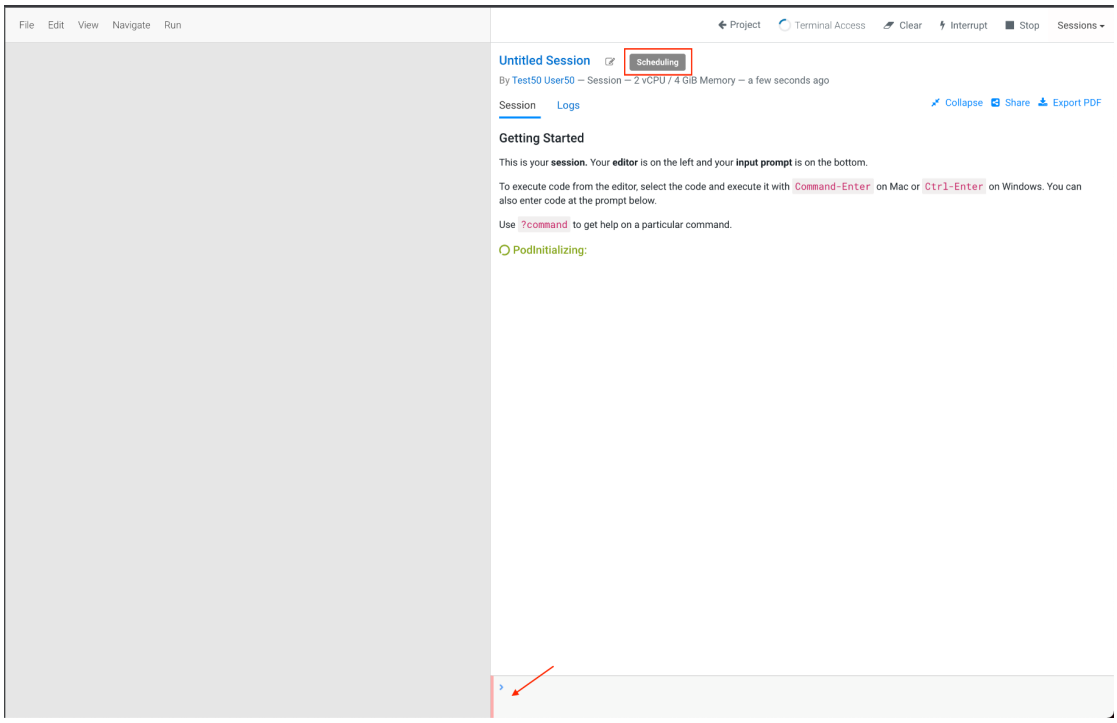
Hacer click en **Start Session**



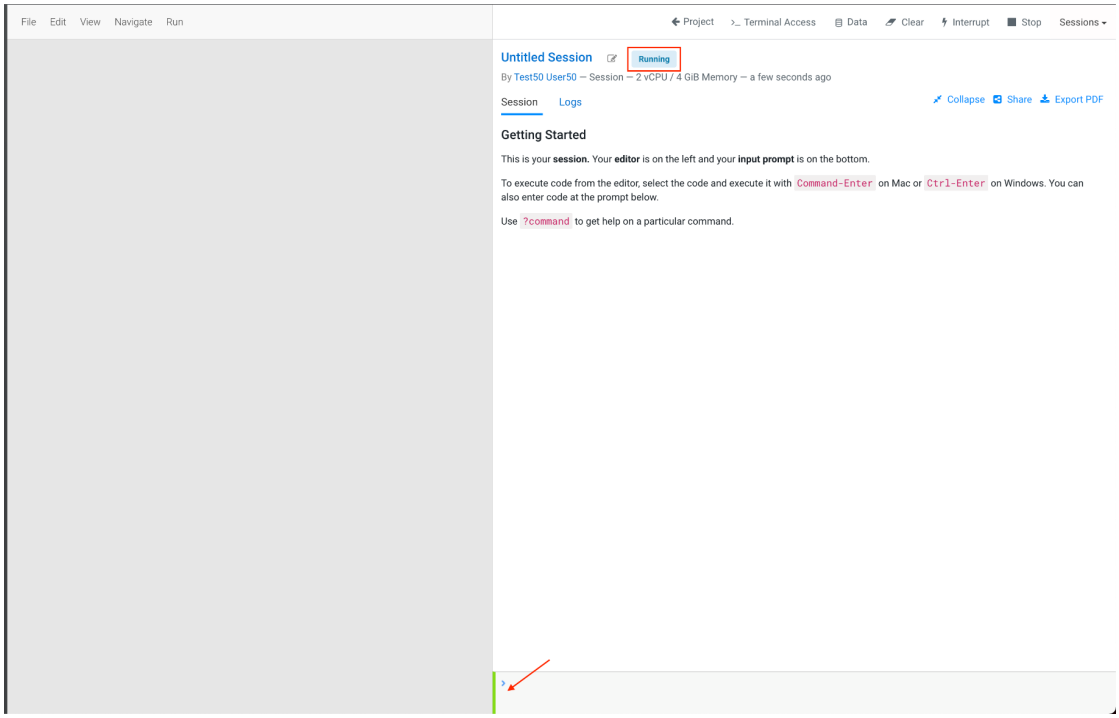
7. Al iniciar la sesión por primera vez, preguntará si deseas utilizar alguna conexión de datos. Este proyecto no necesita de este tipo de conexión. Marcar el check de **Don't show me this again**, y después hacer clic en el botón **Close**.



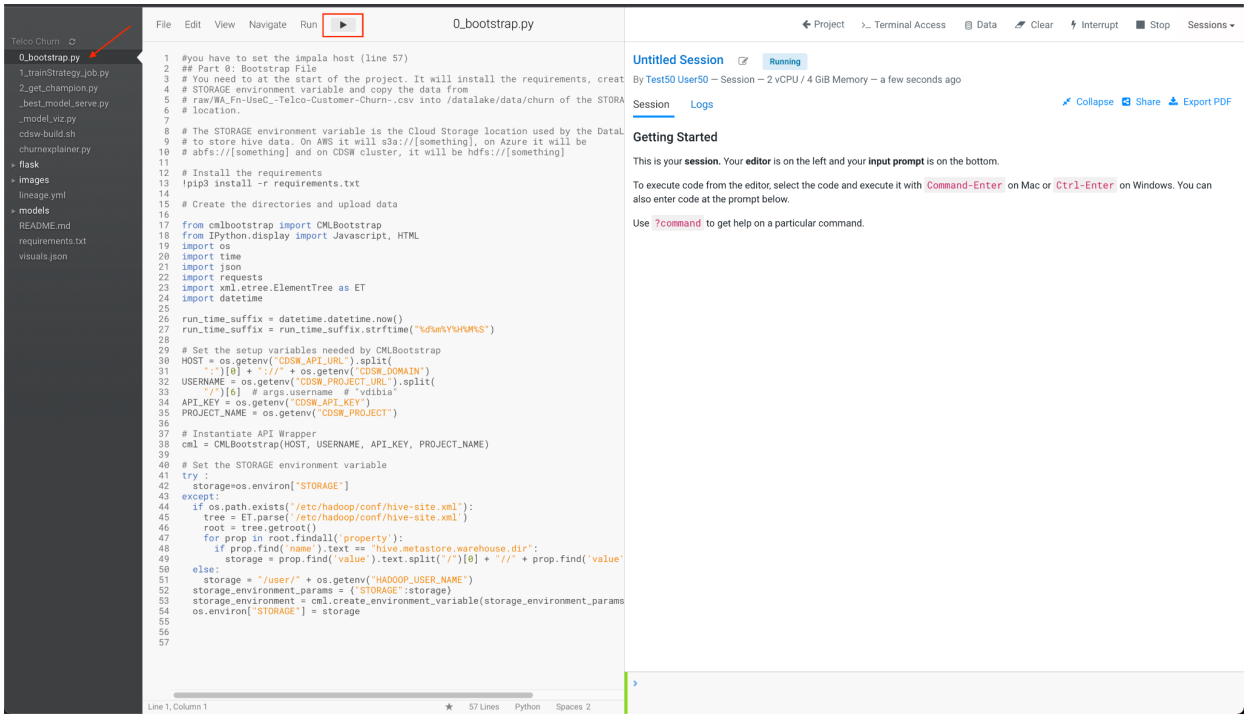
8. El editor/notebook ubicado en el lado derecho de la ventana, estará en estado **Scheduling**, y la barra de comando inferior parpadeando en color rojo. Esto significa que CML está alocando cómputo para su sesión.



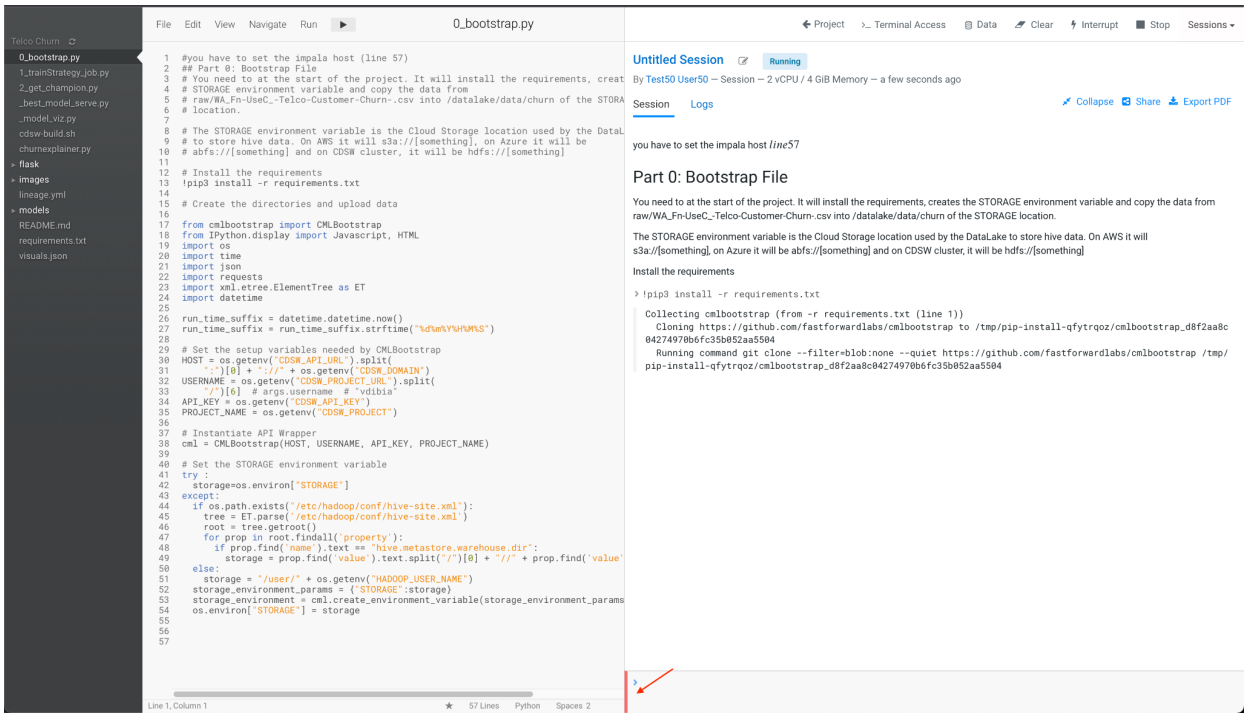
Pasado algunos segundos, el estado cambia a **Running**, y la barra de comando a color verde. Esto significa que la sesión está lista para ejecutar código.



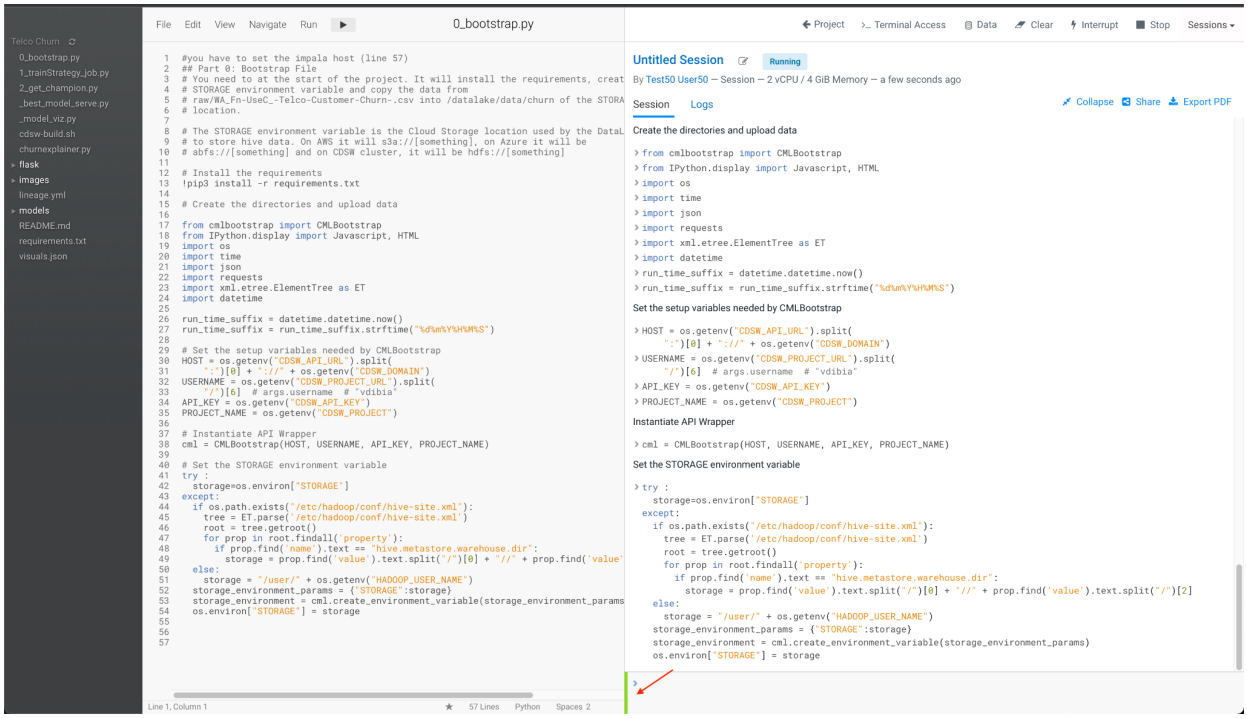
9. El primer script/código para ejecutar es **0_bootstrap.py**. Este código en Python configura las librerías requeridas para el proyecto e integración con el Lakehouse. Selecciona (solo un clic) el fichero en la barra ubicada en el lado izquierdo de la interfaz, esto hará que el código aparezca en el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



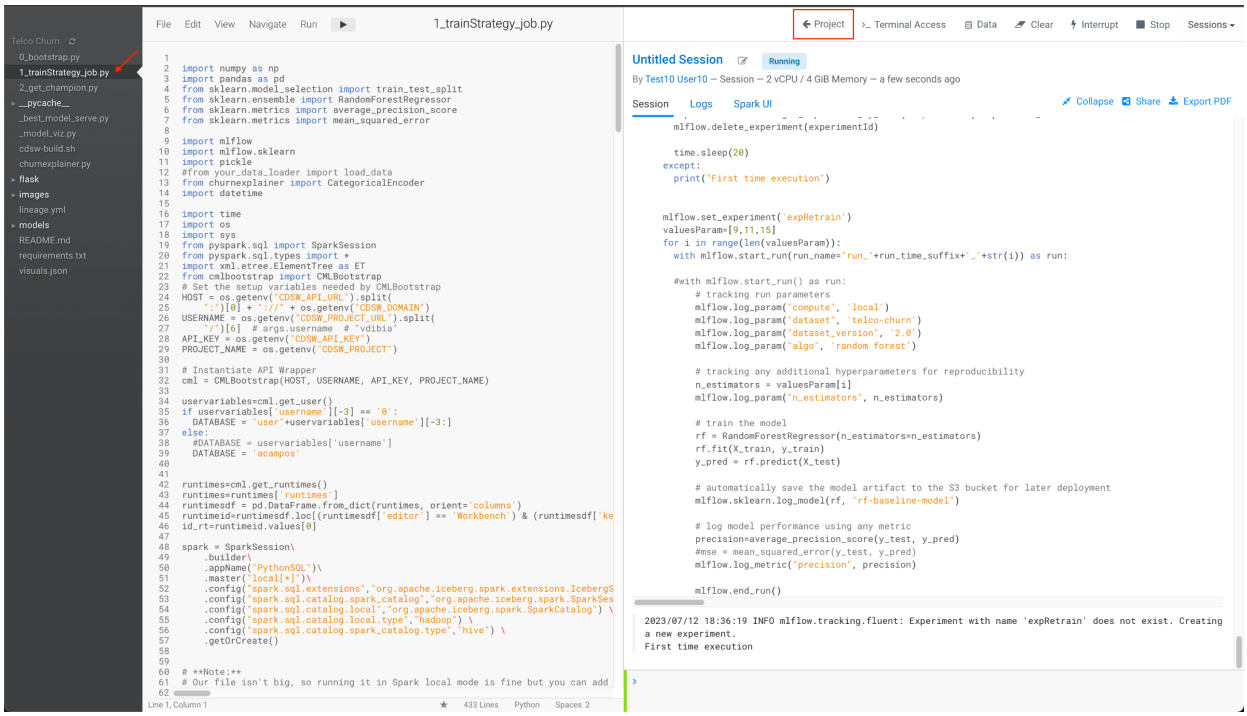
Al empezar la ejecución, verás salida del código en el lado derecho de la interfaz, y la barra de comando inferior cambia a color rojo, indicando que está ocupado.



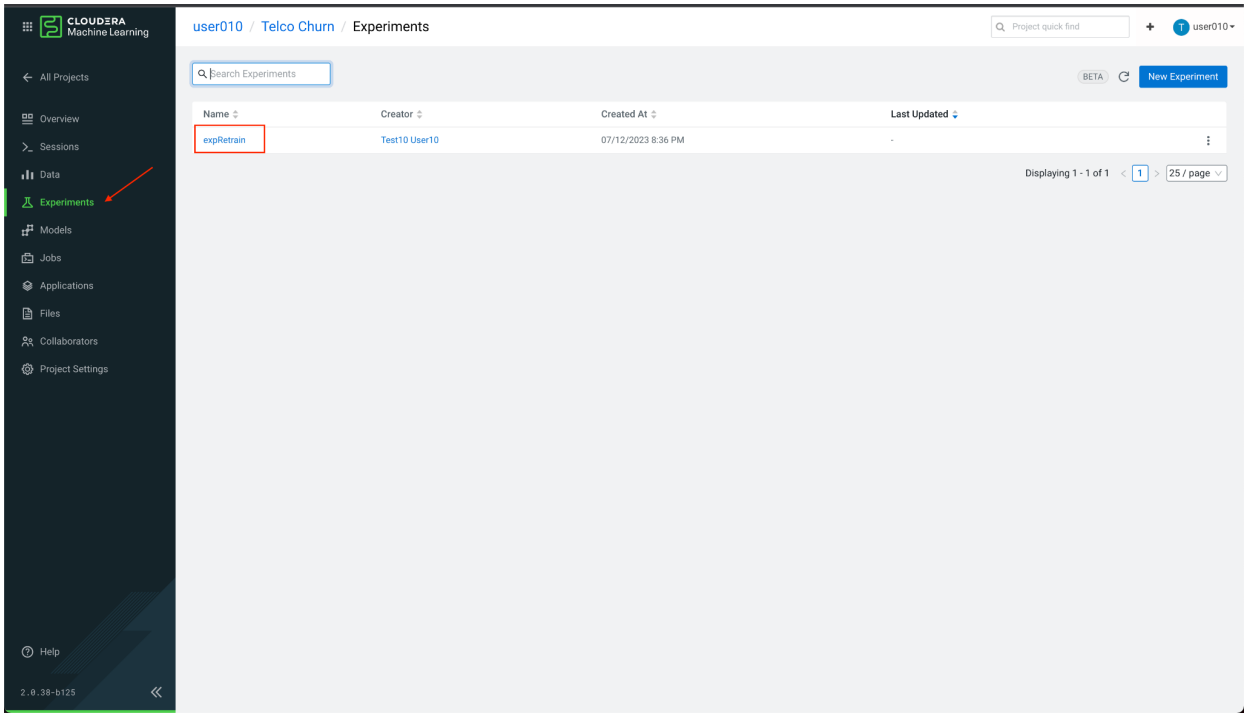
La barra de comando en color verde indica que la ejecución del código ha sido finalizada. Este código se demora entre 3 a 4 minutos en ejecutarse.



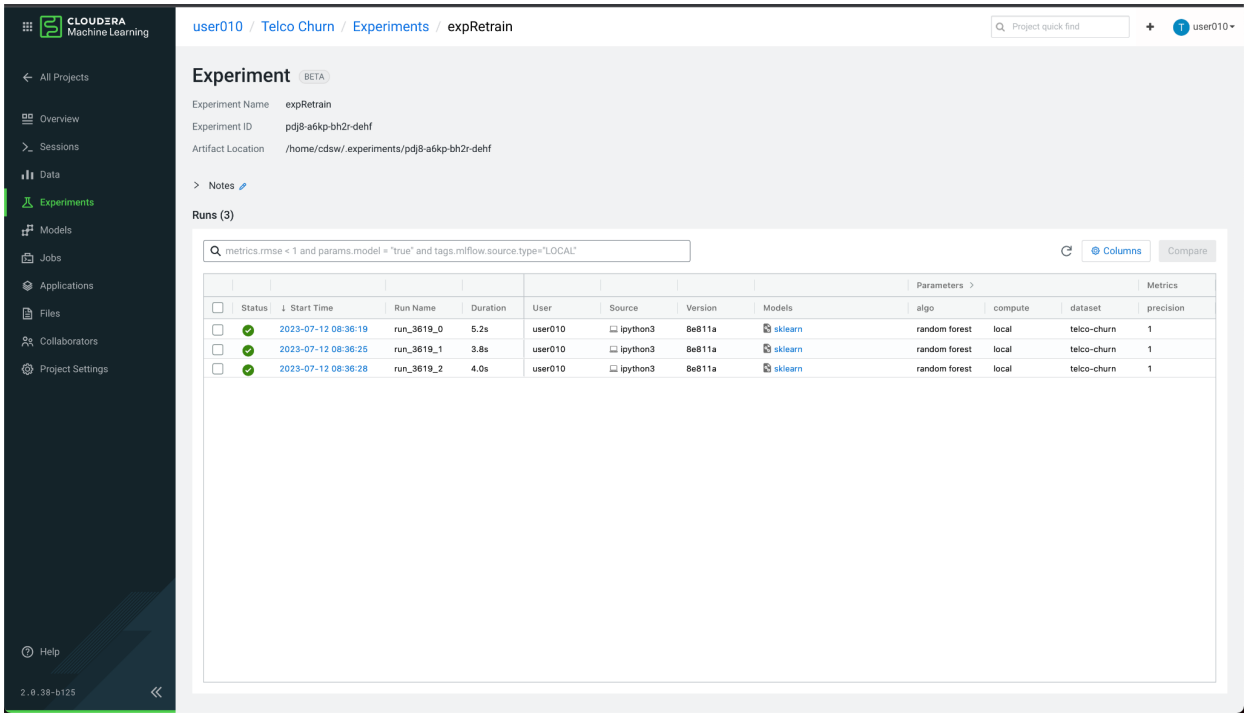
10. El segundo script/código para ejecutar es **1_trainStrategy_job.py**. Este código en Python creará el Experimento para ejecutar el modelo con tres hiper parámetros distintos e registra la precisión. Selecciona (solo un clic) el fichero en la barra ubicada en el lado izquierdo de la interfaz, esto 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.



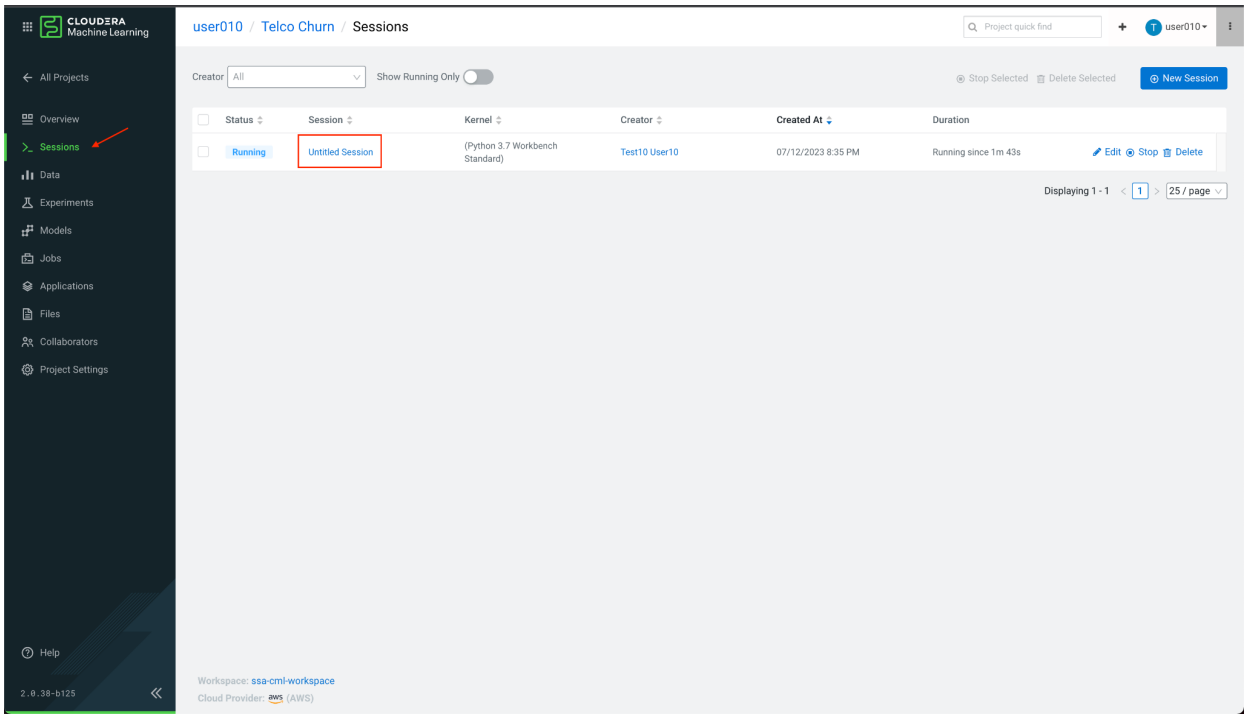
11. Una vez de vuelta en la interfaz principal del proyecto, hacer clic en la opción **Experiments**, del menú izquierdo, y posteriormente en **expRetrain** en el listado de Experiments que aparece.



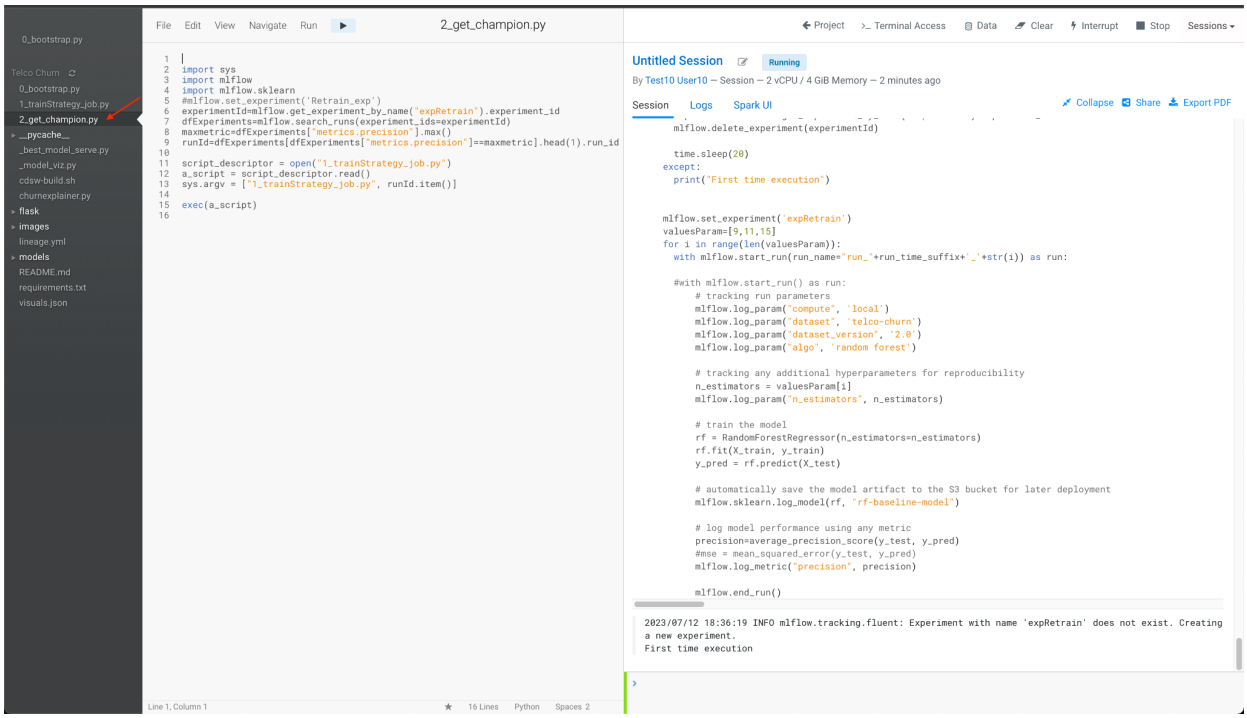
12. En esta pantalla verás las tres ejecuciones del experimento. Fíjense en la última columna, donde aparece **precision**. Esto es la precisión que está entregando cada hiper parámetro.



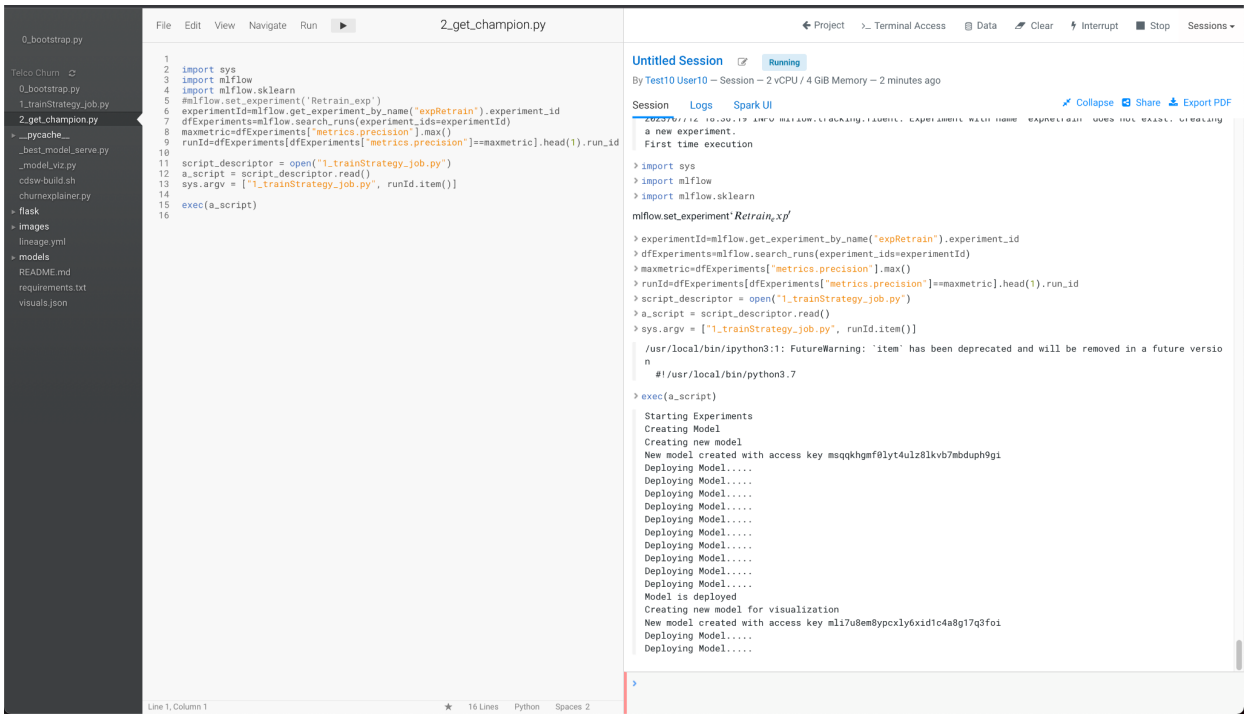
13. Regresemos a la sesión para ejecutar el último código. Como las sesiones se ejecutan en contenedores de Kubernetes, es muy fácil regresar a donde estábamos. Hacer clic en la opción **Sessions** del menú izquierdo, y posteriormente en la única sesión que aparecerá en el listado. Si no agregaste un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



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.

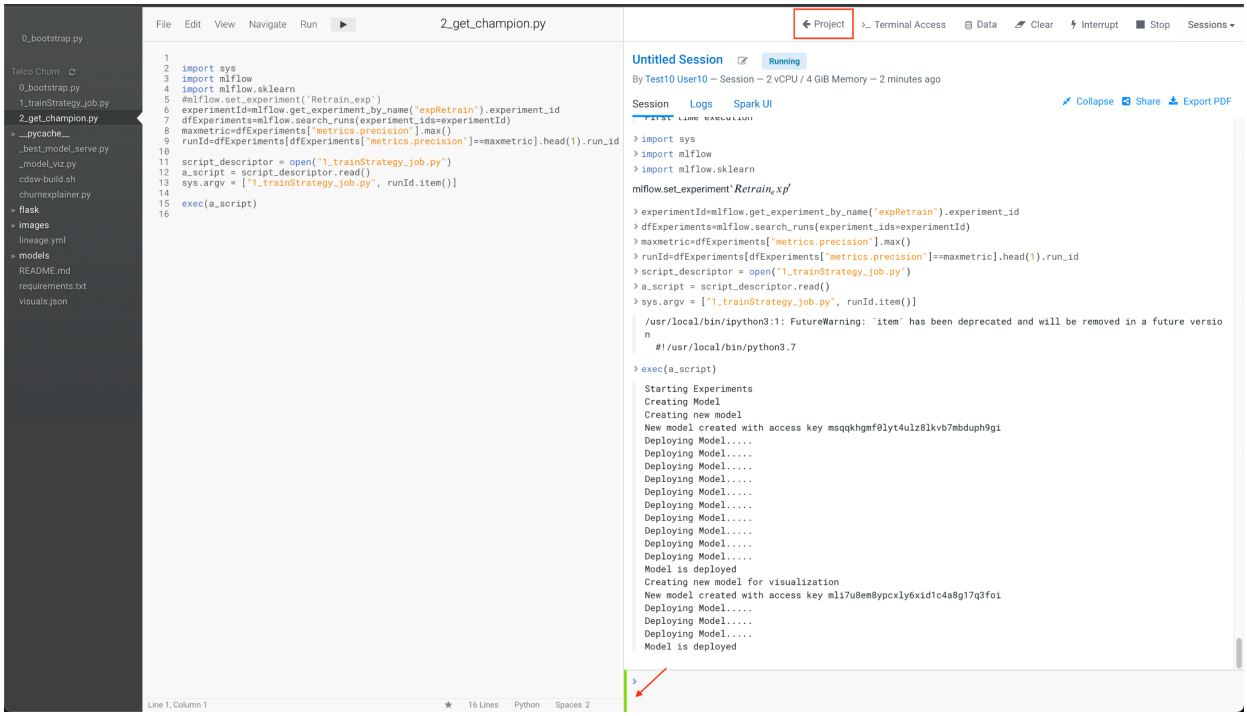


Después de unos segundos, verás el siguiente mensaje “Deploying Model…” repetirse varias veces, y la barra de comandos inferior estará en color rojo.

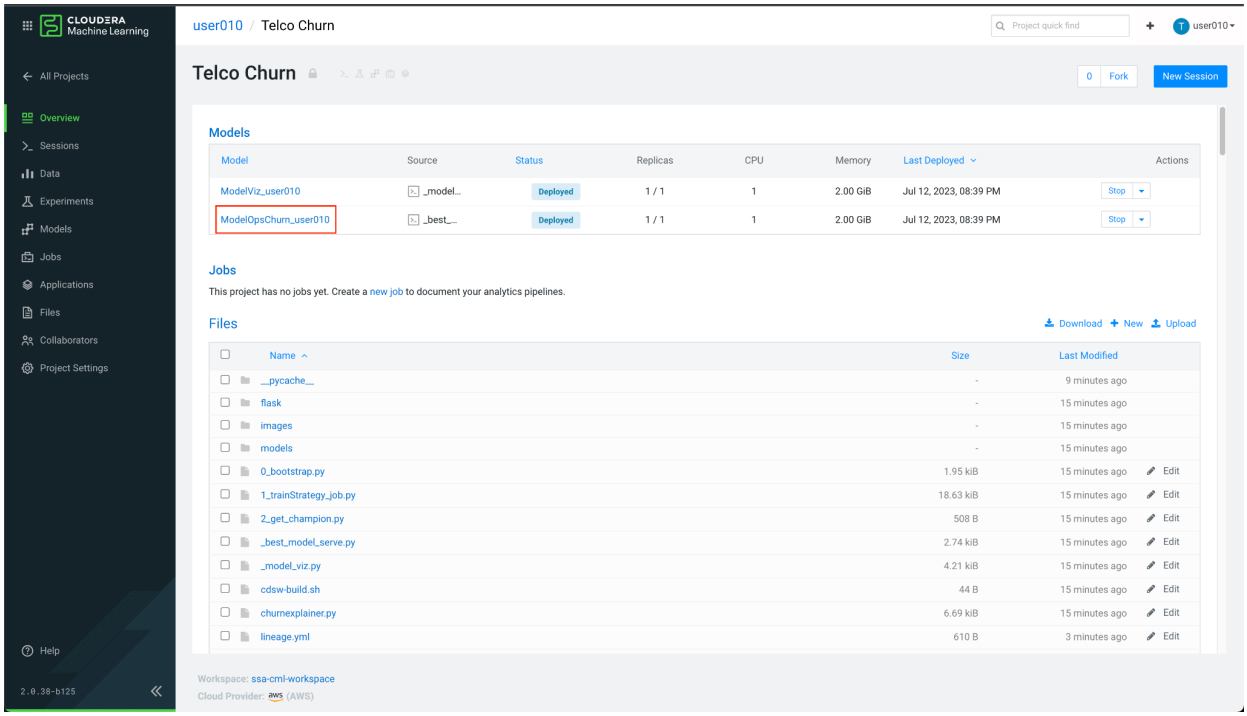


Pasados unos 2 minutos, el último mensaje deberá ser “Model is deployed”, y la barra estará en color verde. Significa que el despliegue de los dos Models está finalizado. Hacer clic en el

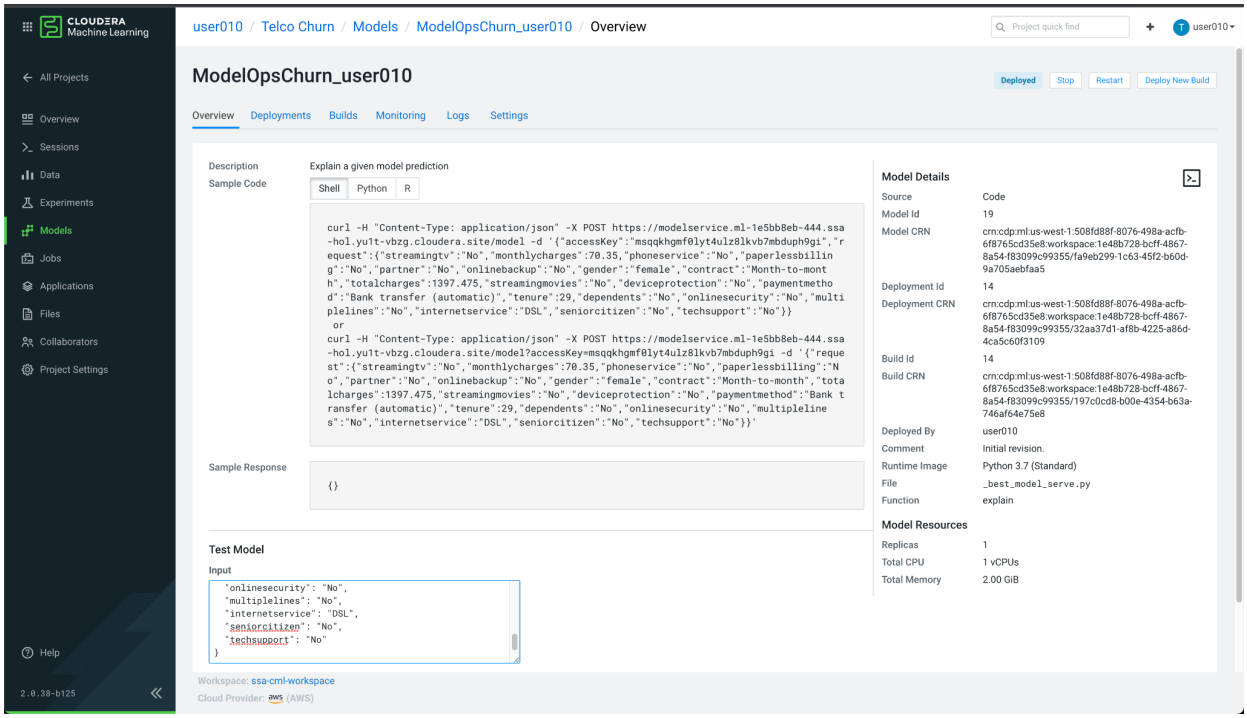
botón **Project**, ubicado en la barra superior derecha de la sesión para volver a la página inicial del proyecto.



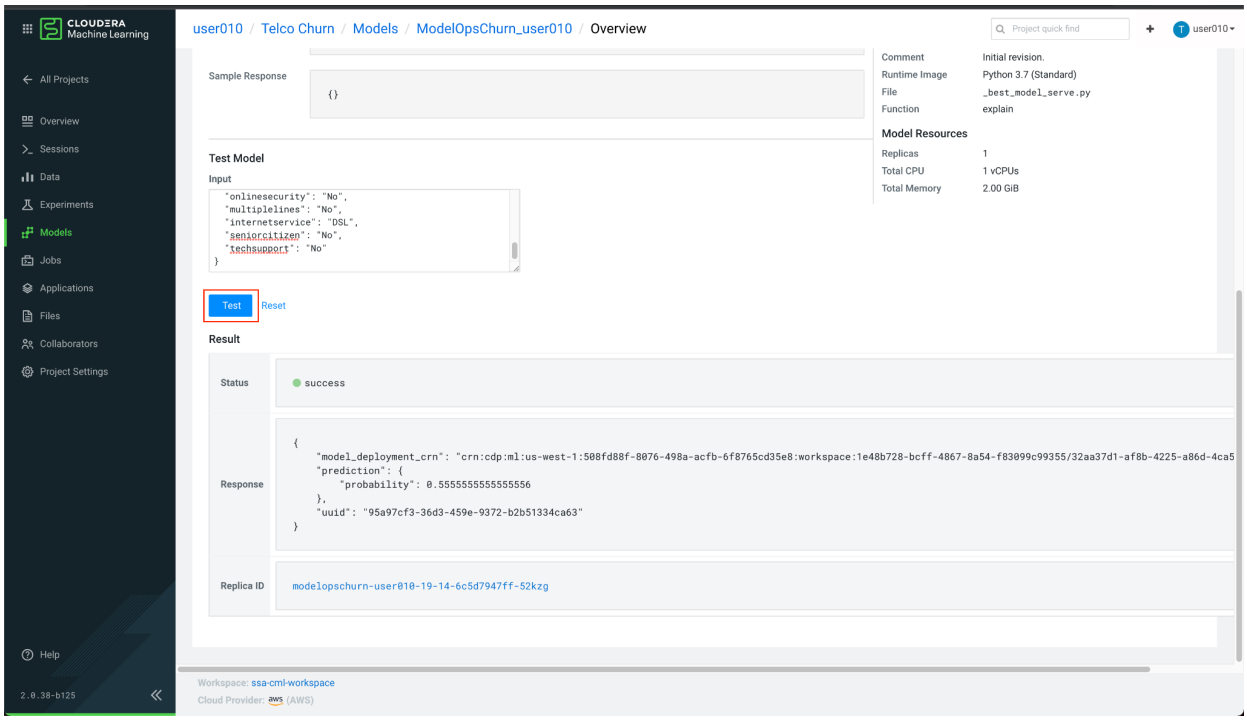
15. Una vez en la página inicial del proyecto, verás los Models desplegados, que son dos. Hacer clic en el que empieza con el nombre **ModelOpsChurn.**



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



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
Project quick find
user010
Sample Response
{}
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-f83099c99355/32aa37d1-af8b-4225-a86d-4ca5
"prediction": {
"probability": 0.5555555555555556
},
"uuid": "95a97cf3-36d3-459e-9372-b2b51334ca63"
}
Replica ID
modelopschurn-user010-19-14-6c5d7947ff-52kzg
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
Workspace: ssa-cml-workspace
Cloud Provider: aws (AWS)