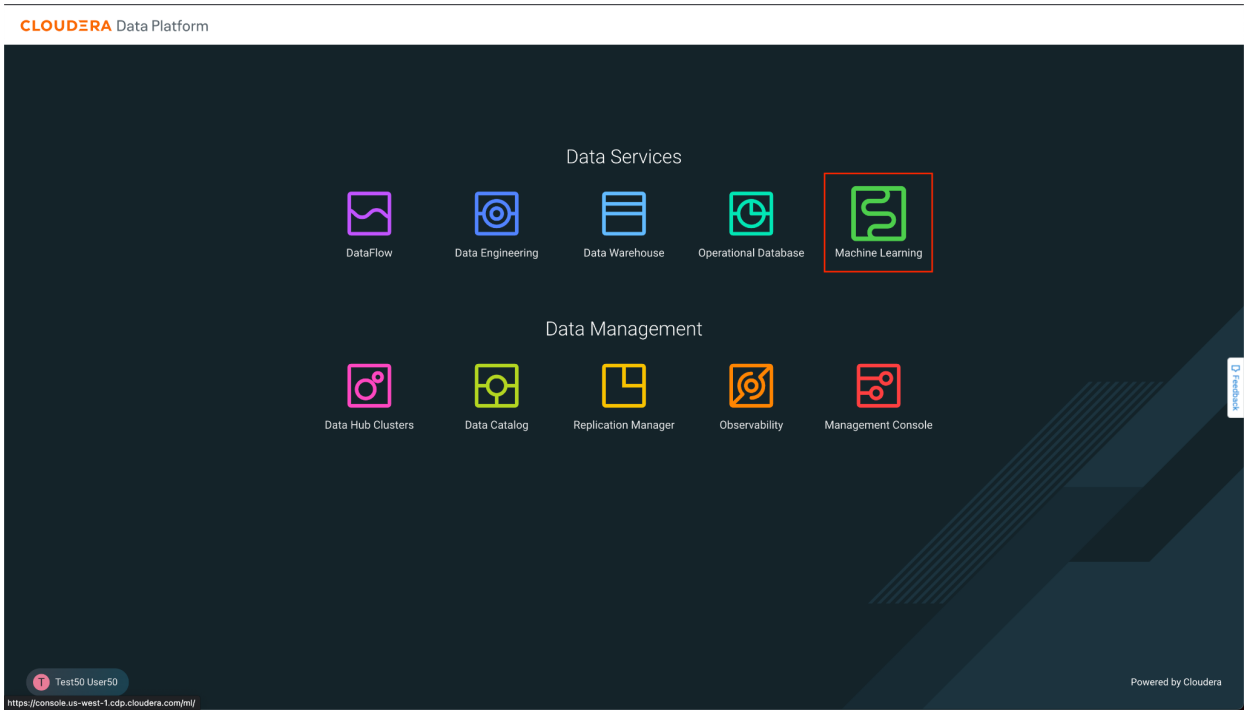
Ciclo de vida dos dados em CDP Public Cloud

Laboratório Machine Learning

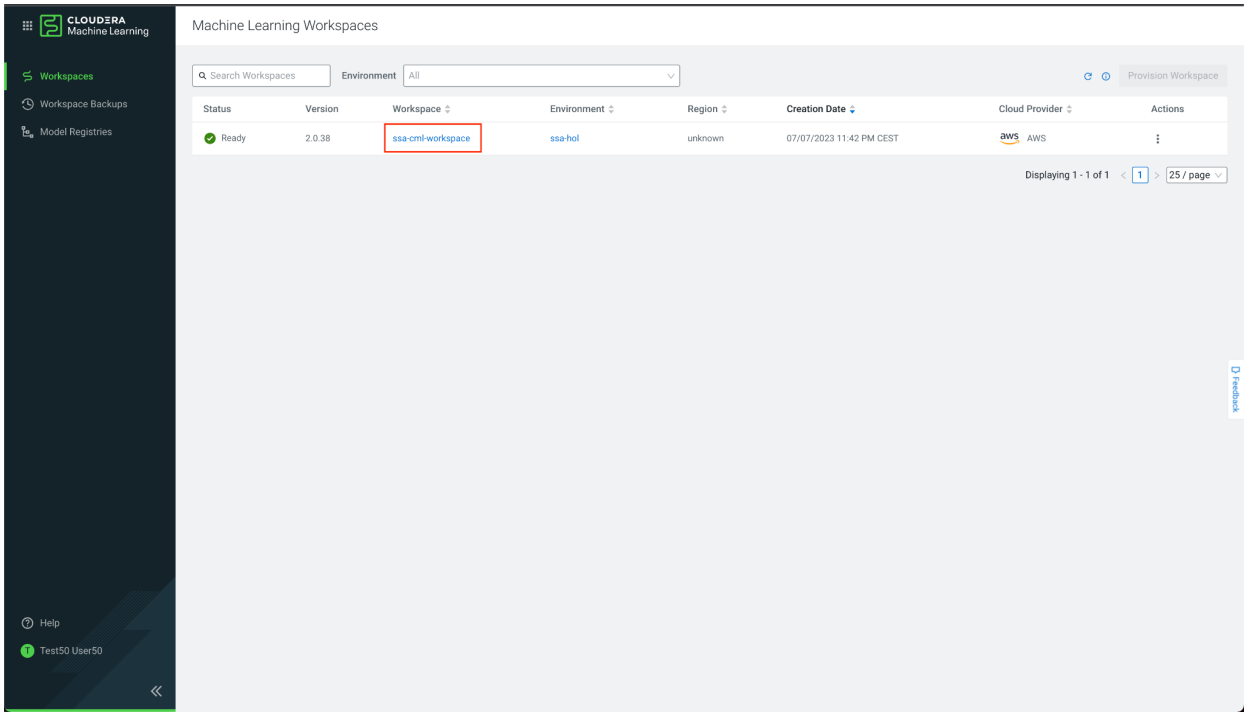
Metas:

- Treine um modelo para prever se um cliente vai parar de consumir produtos
- Implantar/expor modelo como API REST

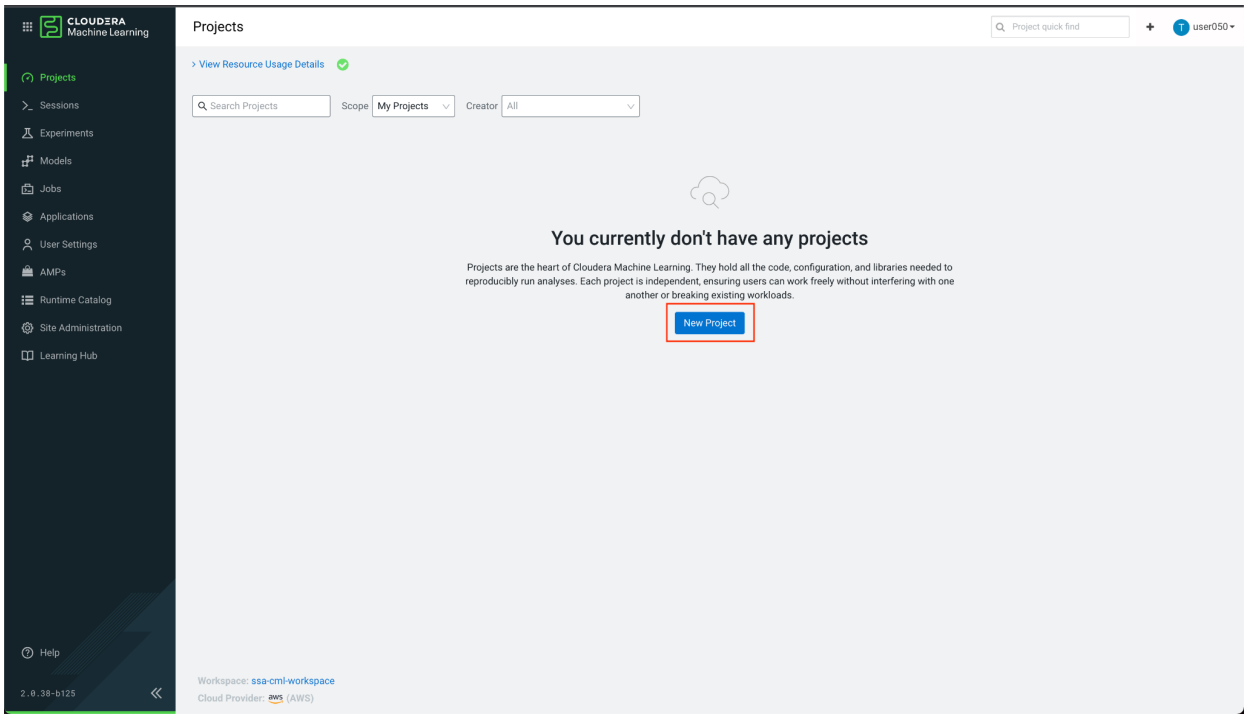
1. Clique em Machine Learning na tela inicial do CDP PC:



2. Tela inicial do Machine Learning. Clique no único Workspace listado.



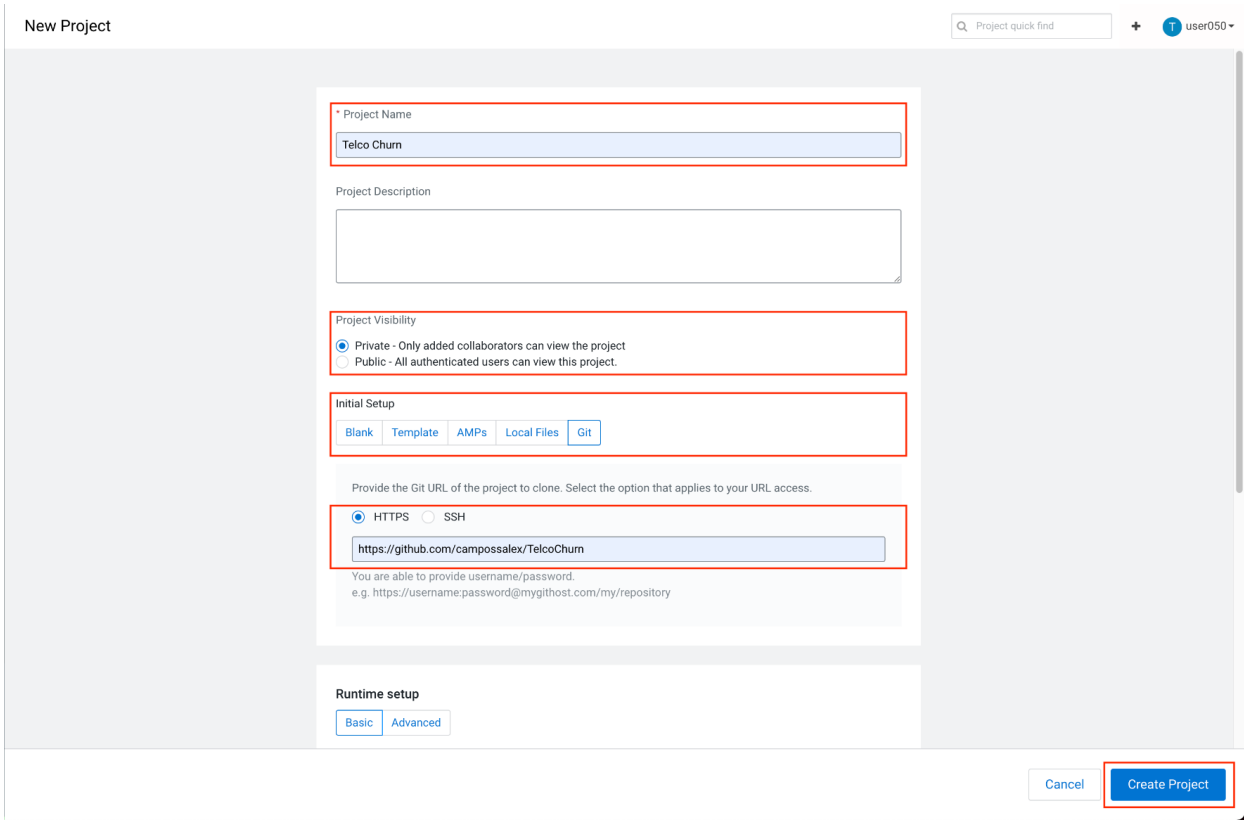
3. Uma vez no Workspace, você deverá ver a seguinte tela. Aqui estão os projetos que você criou. É hora de criar um novo projeto. Clique no botão azul **New Project**.



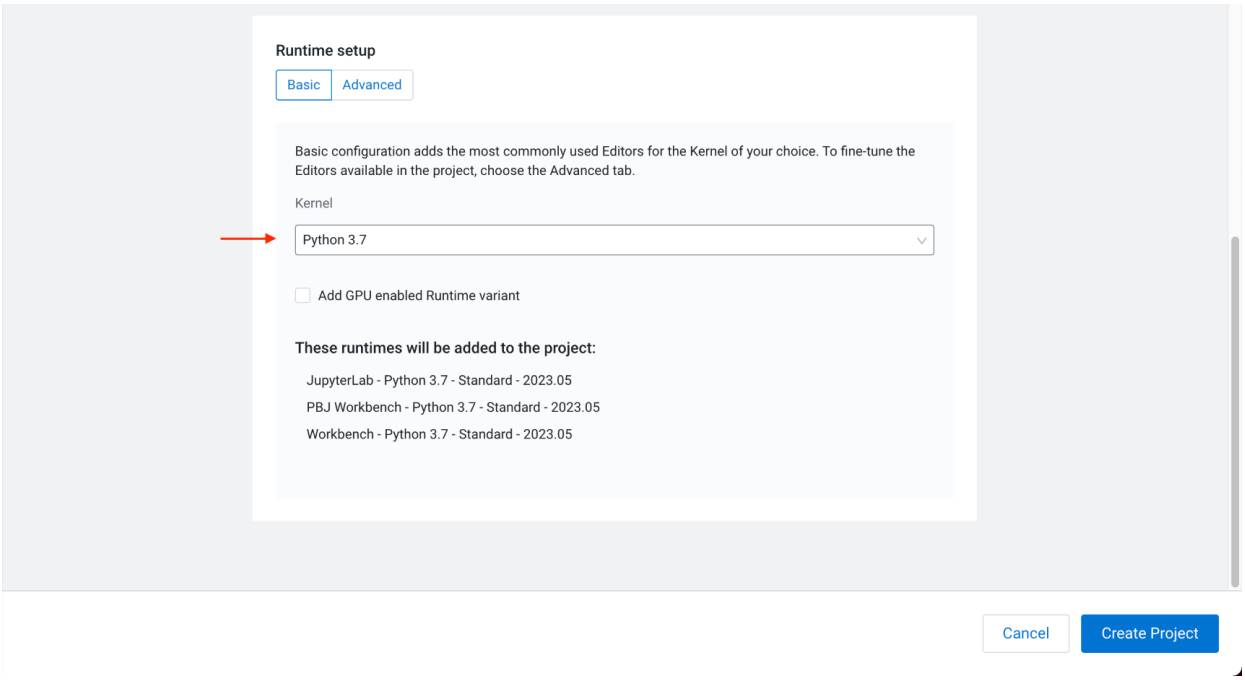
4. Insira as seguintes informações para criar um novo projeto:
Nome do Projeto: Telco Churn

Visibilidade do Projeto: Privado
Configuração inicial, selecione Git
No campo de texto abaixo de HTTPS, insira a url do repositório
git:https://github.com/campossalex/TelcoChurn

Mantenha o restante das configurações iguais. Clique no botão **Create Project**



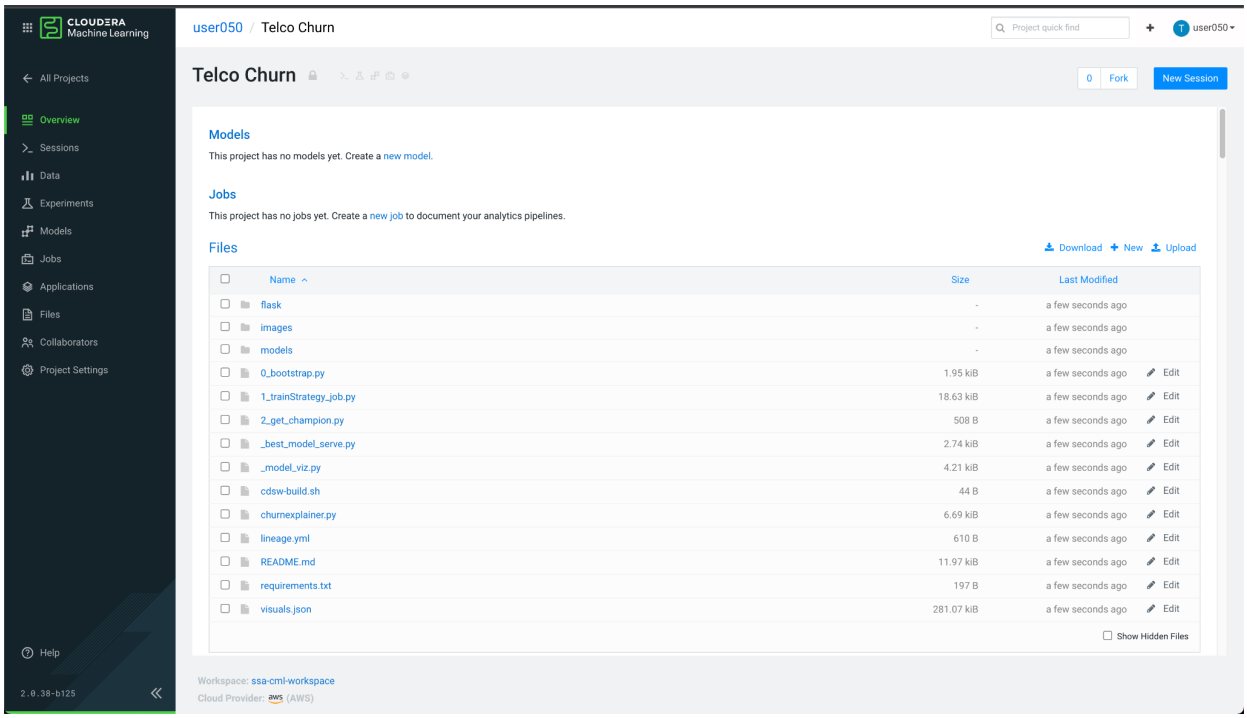
Certifique-se de selecionar **Python 3.7** na seção do kernel. Clique no botão **Create Project**



5. Uma vez criado o projeto, você deverá ver a seguinte tela:

Models, implemente e gerencie modelos como APIs REST para fornecer previsões.
Jobs, automatize e orquestre a execução de rotinas de análise em batch
Files, ativos que fazem parte do projeto, como arquivos, scripts e códigos

Este projeto consiste na execução de três scripts. A forma de execução é através de uma sessão, que é a alocação de computação isolada para cada usuário. Para isso, você deve clicar no botão azul **New Session**, localizado no canto superior direito.

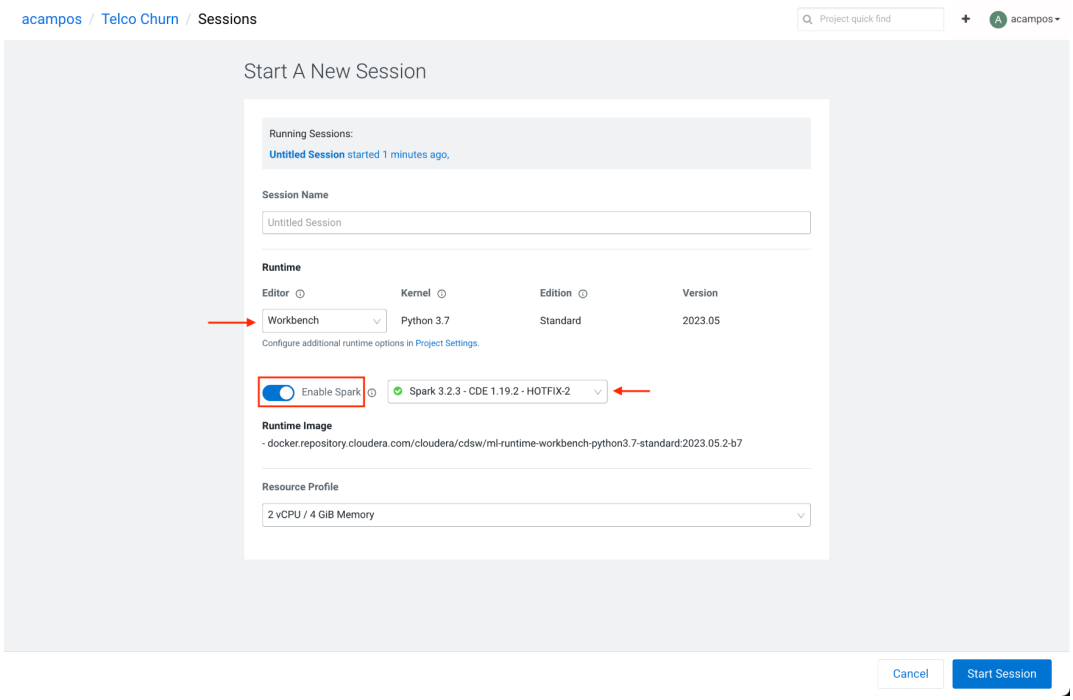


6. Certifique-se de ativar o Spark, marcando a verificação correspondente e clique no botão **Start Session**.

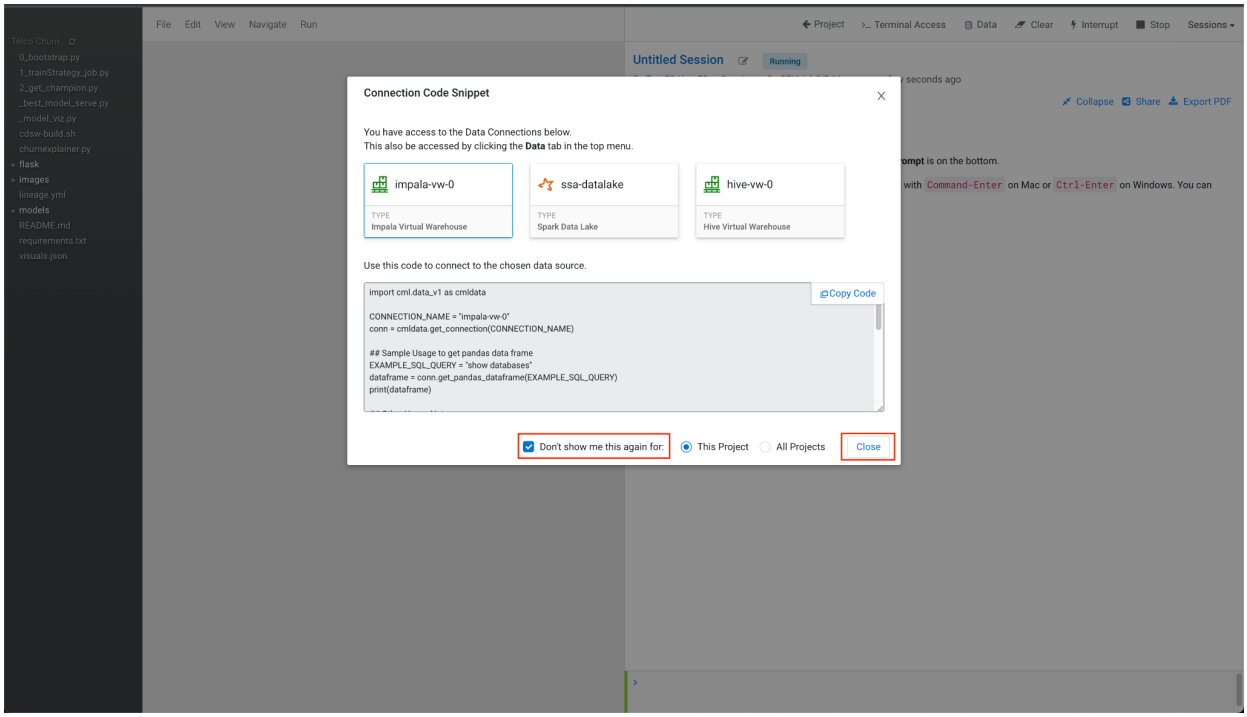
6. Ao fazer login, certifique-se de:

Selecione **Workbench** na opção Editor.
Habilitar **Spark**, marcando a opção correspondente.
Selecione **Spark 3.2.x** na opção de versão do Spark.

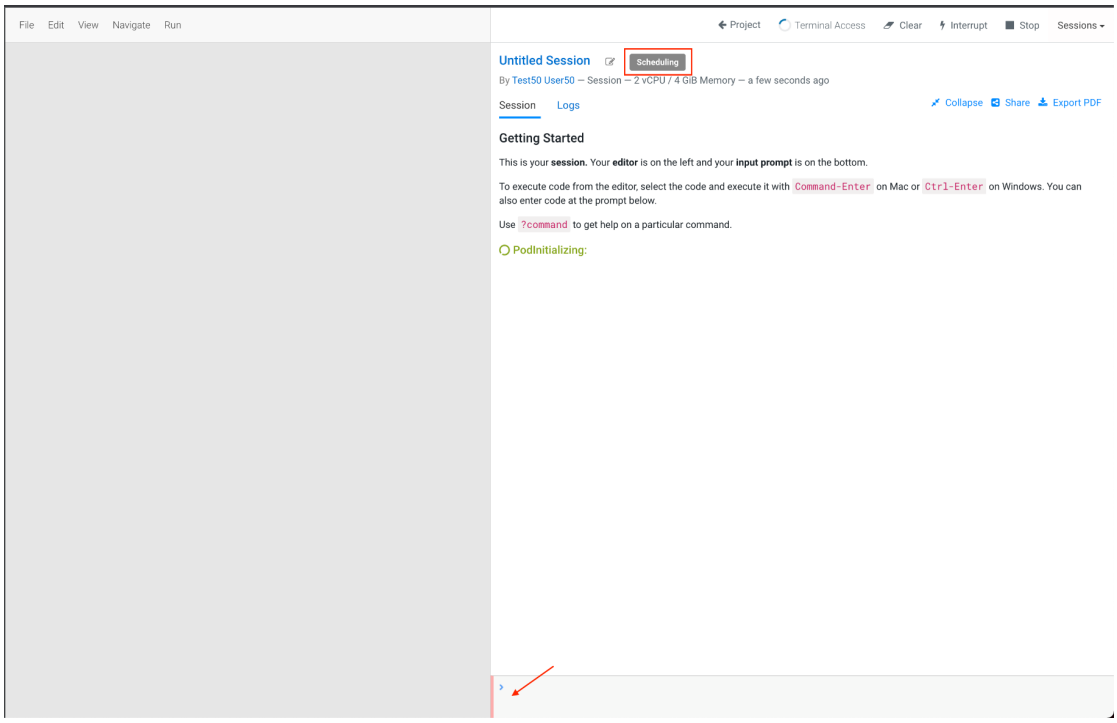
Clique em **Start Session**



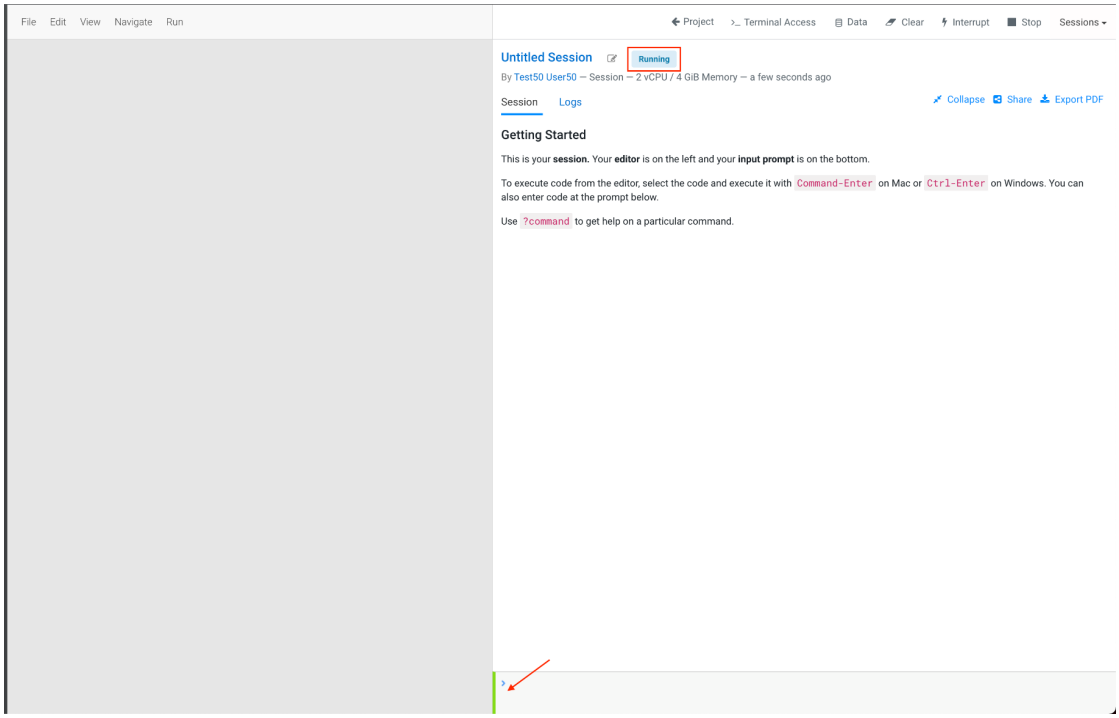
7. Ao iniciar a sessão pela primeira vez, ele perguntará se você deseja usar uma conexão de dados. Este projeto não precisa deste tipo de conexão. marcar o checkbox de **Don" show me this again for** e, em seguida, clique no botão **Close**.



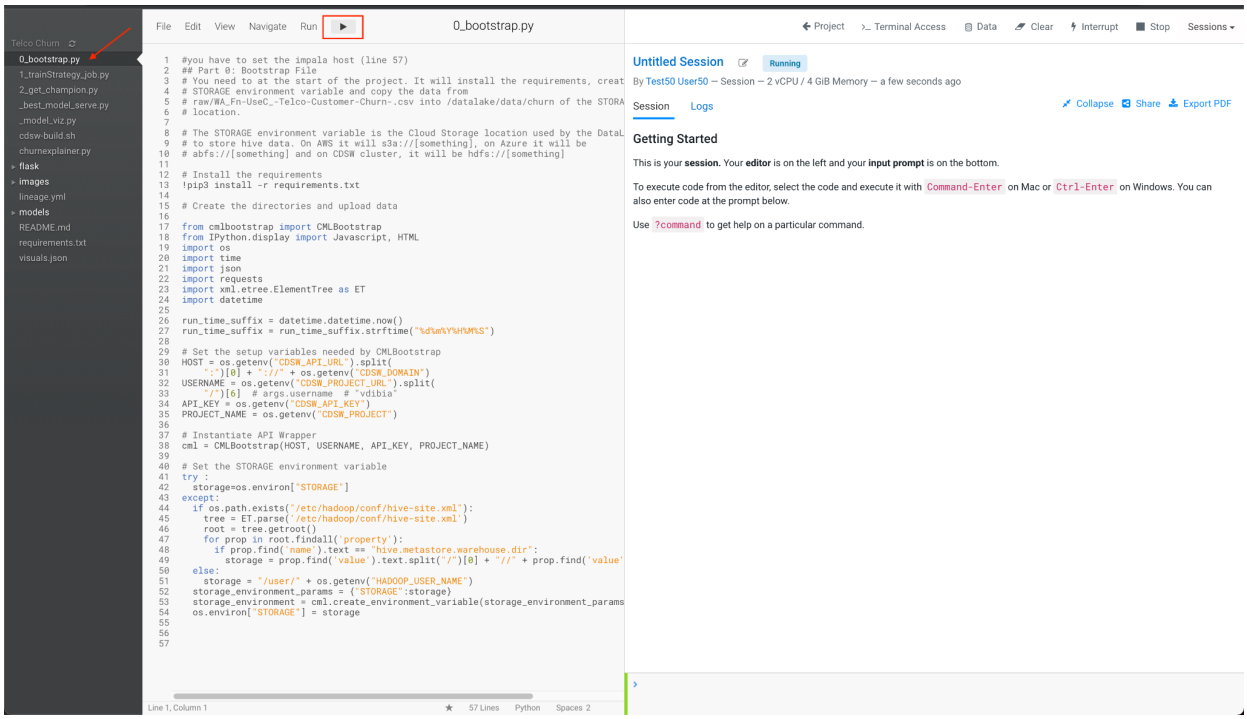
8. O editor/notebook localizado no lado direito da janela estará no estado **Scheduling**, e a barra de comandos inferior piscando em vermelho. Isso significa que o CML está alocando recursos para sua sessão.



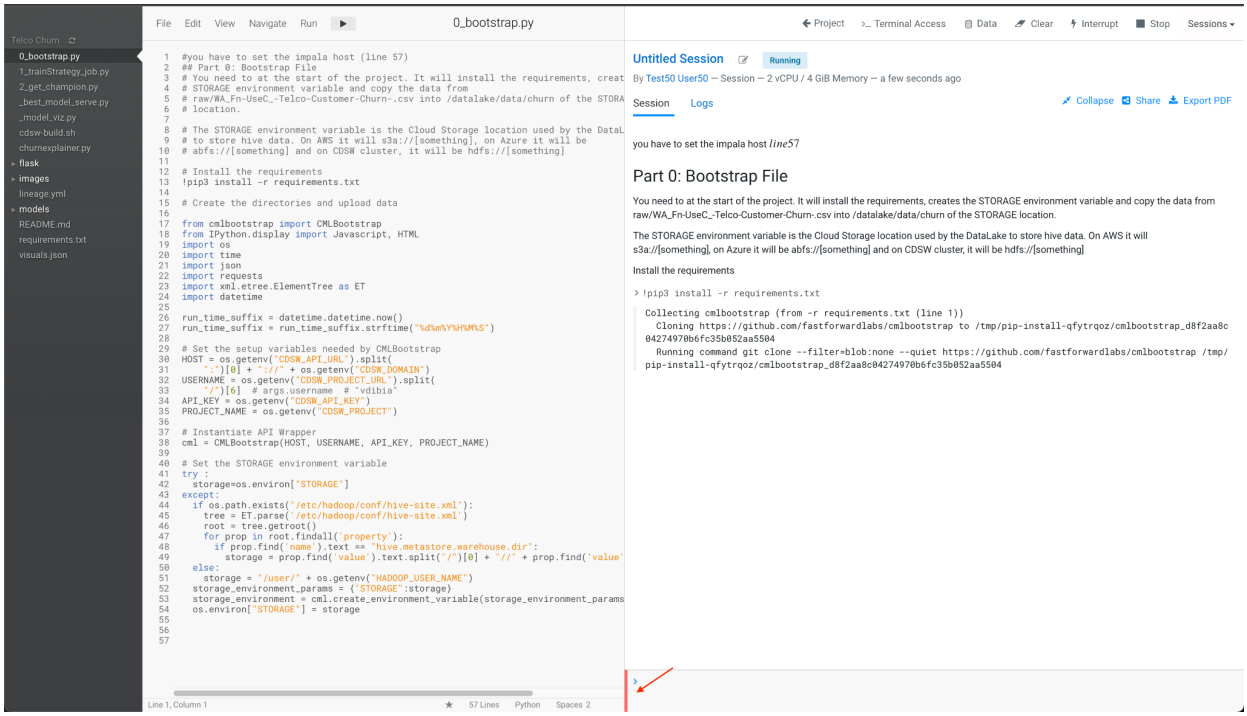
Após alguns segundos, o status muda para **Running**, e a barra de comandos para verde. Isso significa que a sessão está pronta para executar o código.



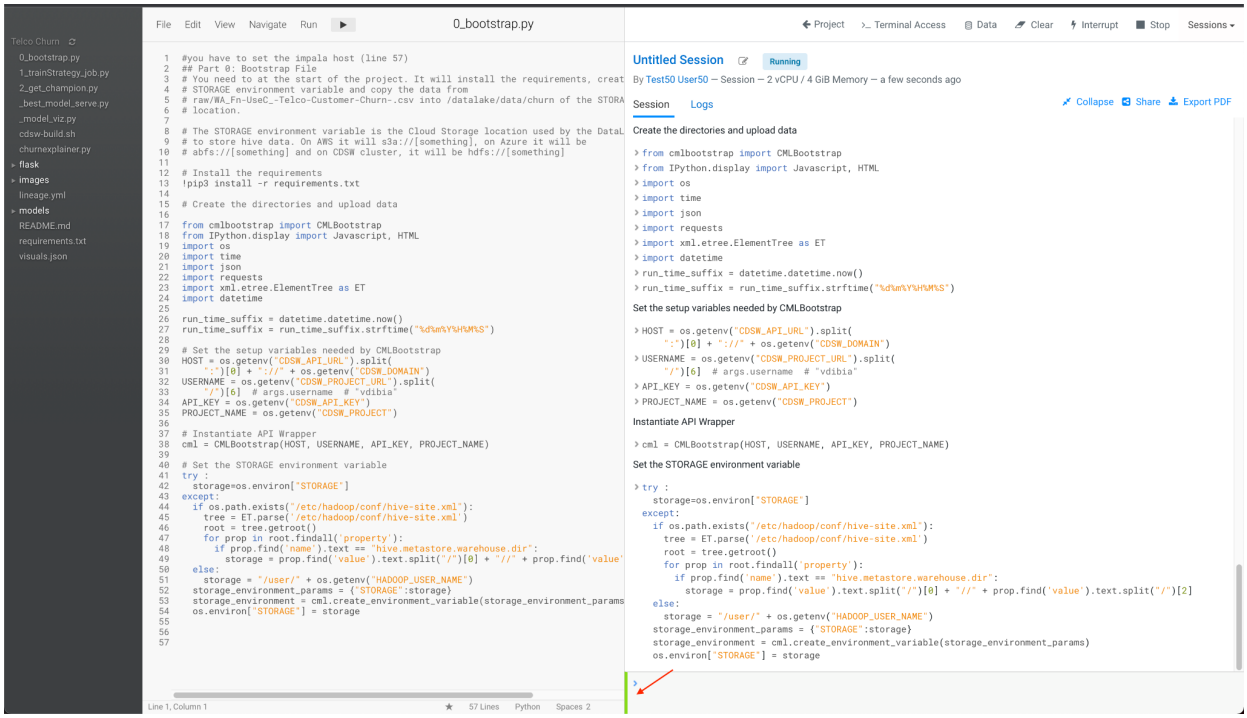
9. O primeiro script/código a ser executado é **0_bootstrap.py**. Este código Python configura as bibliotecas necessárias para o projeto e a integração com o Lakehouse. Selecione (apenas um clique) o arquivo na barra localizada no lado esquerdo da interface, isso fará com que o código apareça no editor. Após selecionar o arquivo, clique no botão ▶ para executar o código.



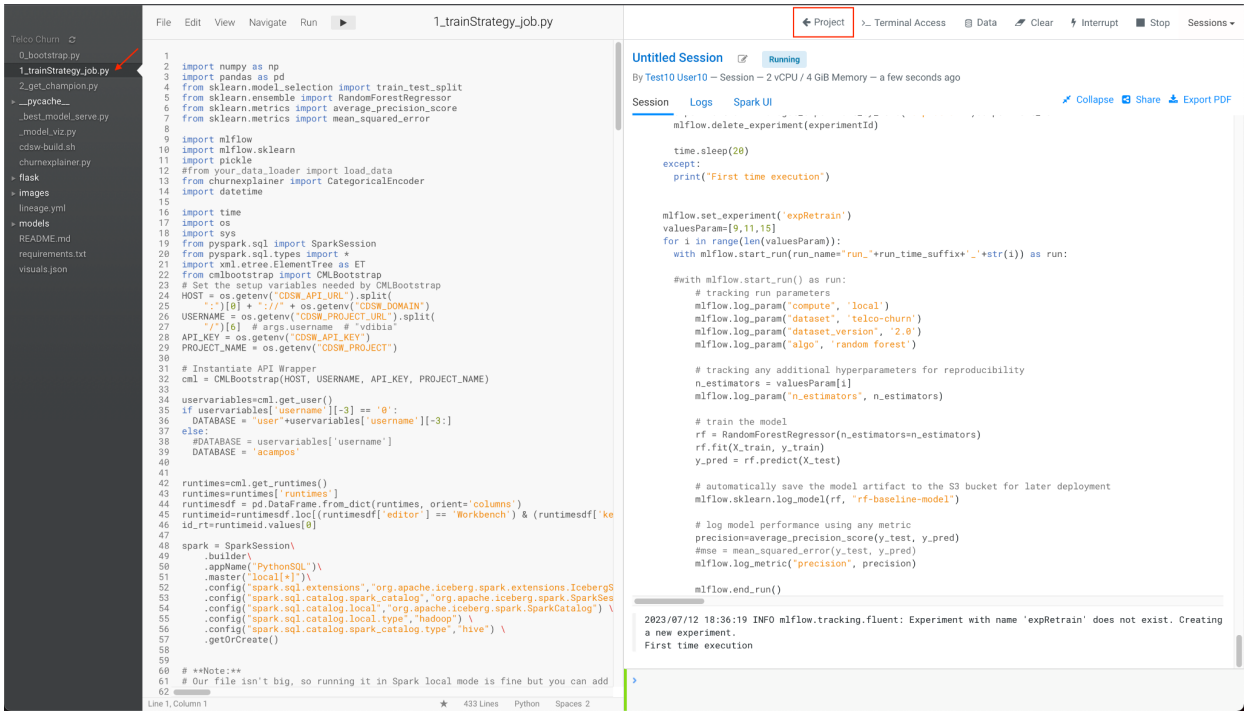
Ao iniciar a execução, você verá os detalhes da execução do código no lado direito da interface e a barra de comandos inferior ficará vermelha, indicando que está ocupada.



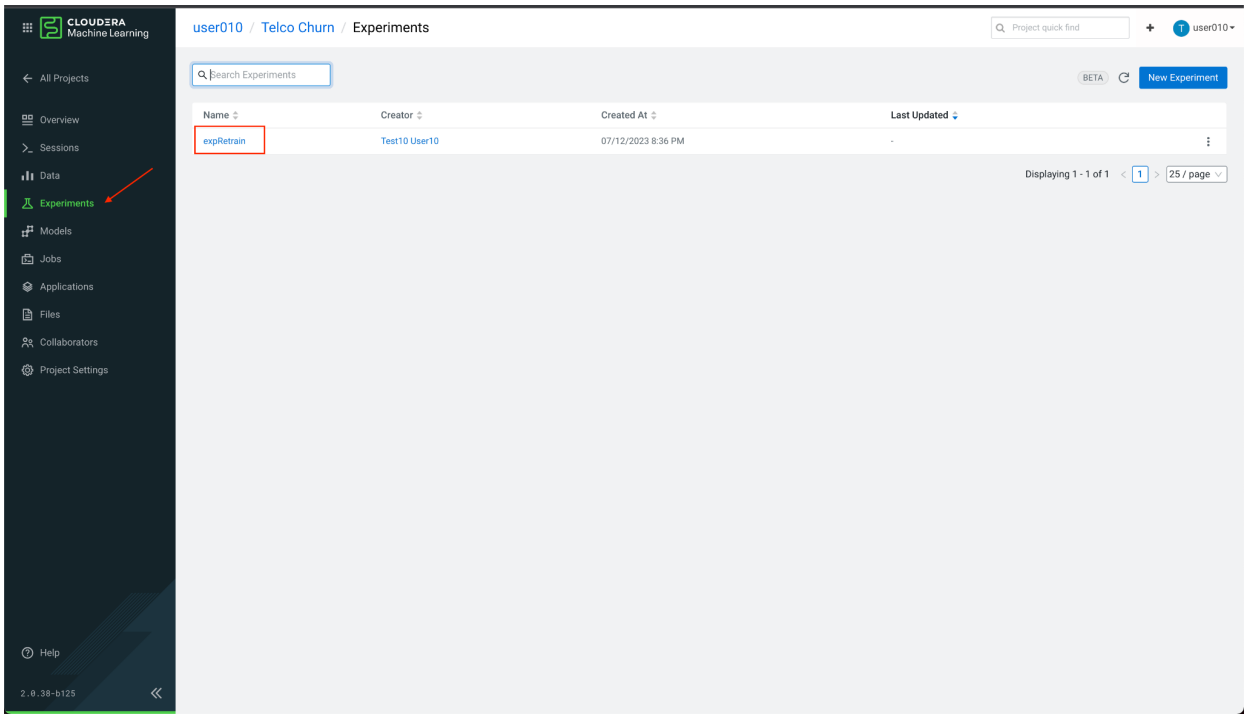
A barra de comando verde indica que a execução do código foi concluída. Este código leva de 3 a 4 minutos para ser executado.



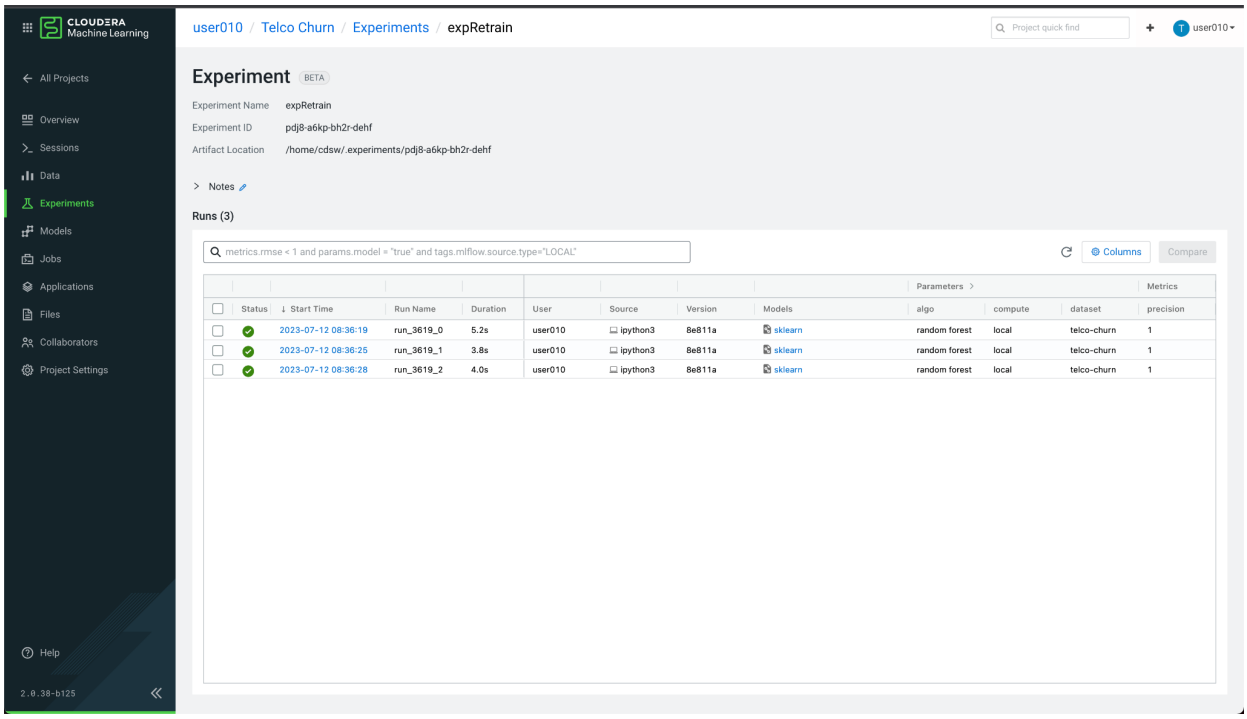
10. O segundo script/código a ser executado é o **1_trainStrategy_job.py**. Este código Python criará o experimento para executar o modelo com três hiperparâmetros diferentes e registrar a precisão. Selecione (apenas um clique) o arquivo na barra localizada no lado esquerdo da interface, isso fará com que o código apareça no editor. Após selecionar o arquivo, clique no botão ▶ para executar o código. Terminada a execução (aproximadamente 1 minuto), clique no botão **Project**, localizado na barra superior direita da sessão para retornar à página inicial do projeto.



11. De volta à interface principal do projeto, clique na opção **Experiments**, no menu à esquerda e, em seguida, em **expRetrain** na lista de experimentos exibida.

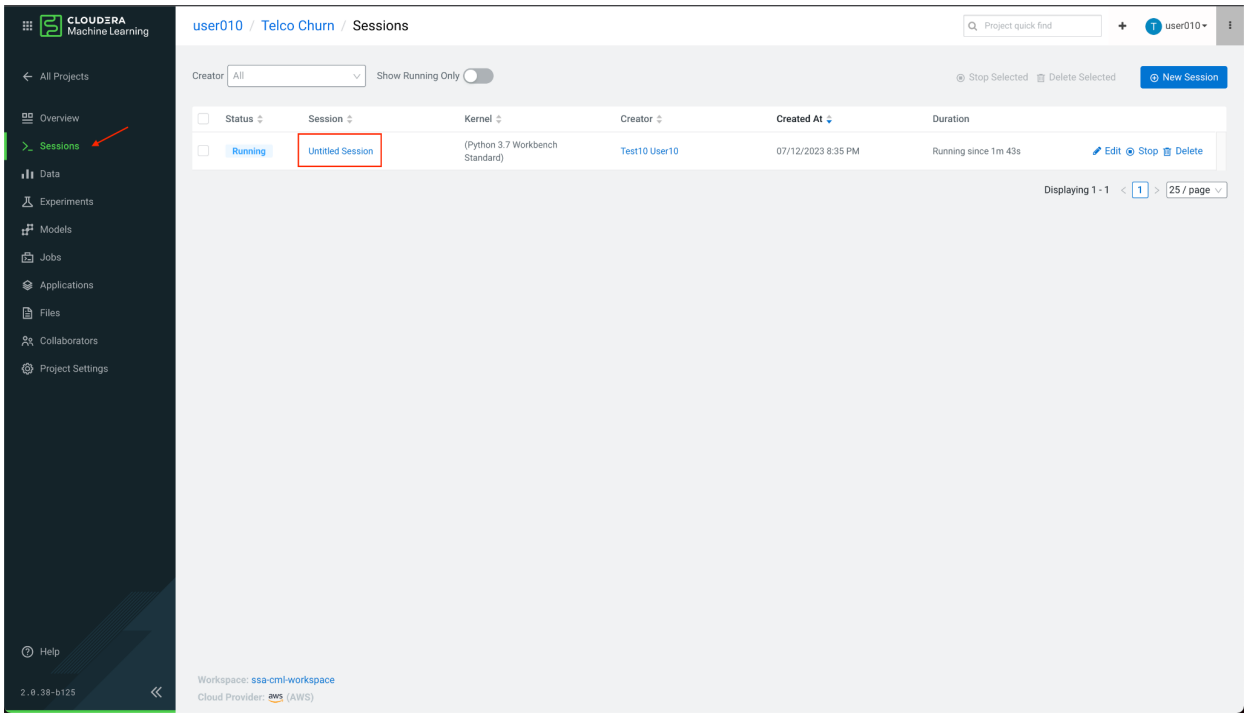


12. Nesta tela você verá as três execuções do experimento. Olhe para a última coluna, onde aparece **precision**. Esta é a precisão que cada hiperparâmetro está entregando.

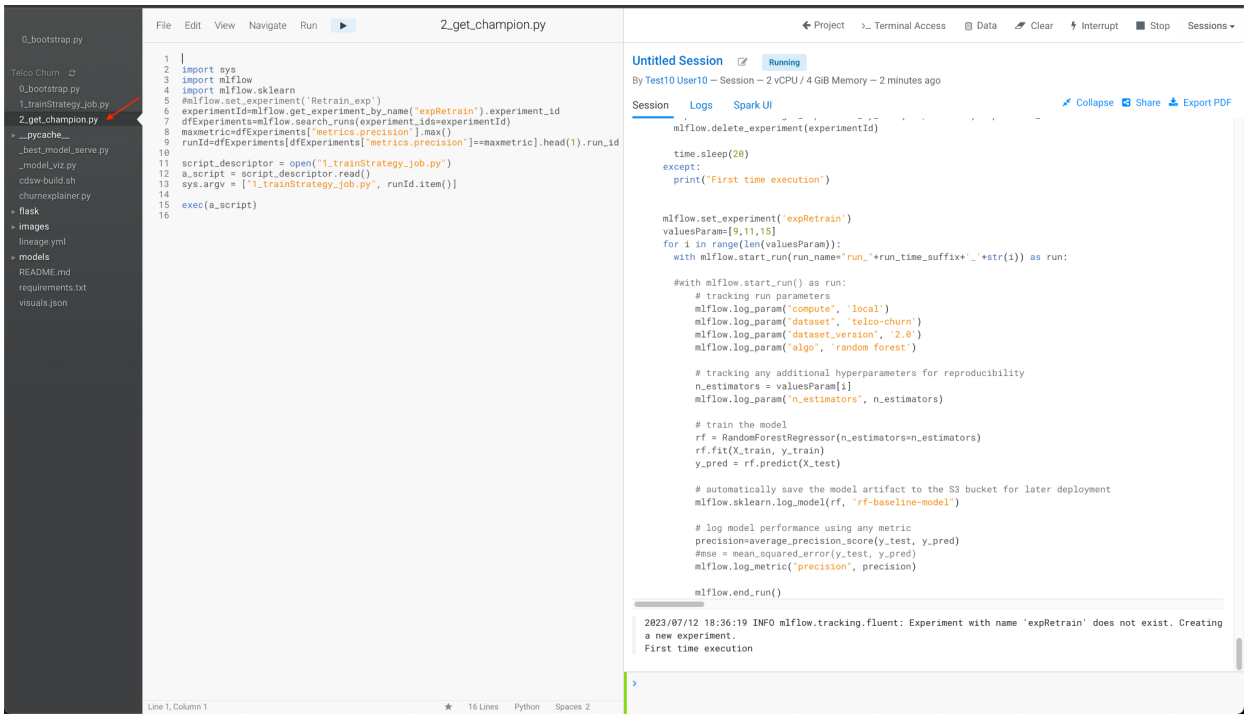


13. Voltemos à sessão para executar o último código. Como as sessões estão sendo executadas em contêineres do Kubernetes, é muito fácil voltar para onde estávamos. Clique na

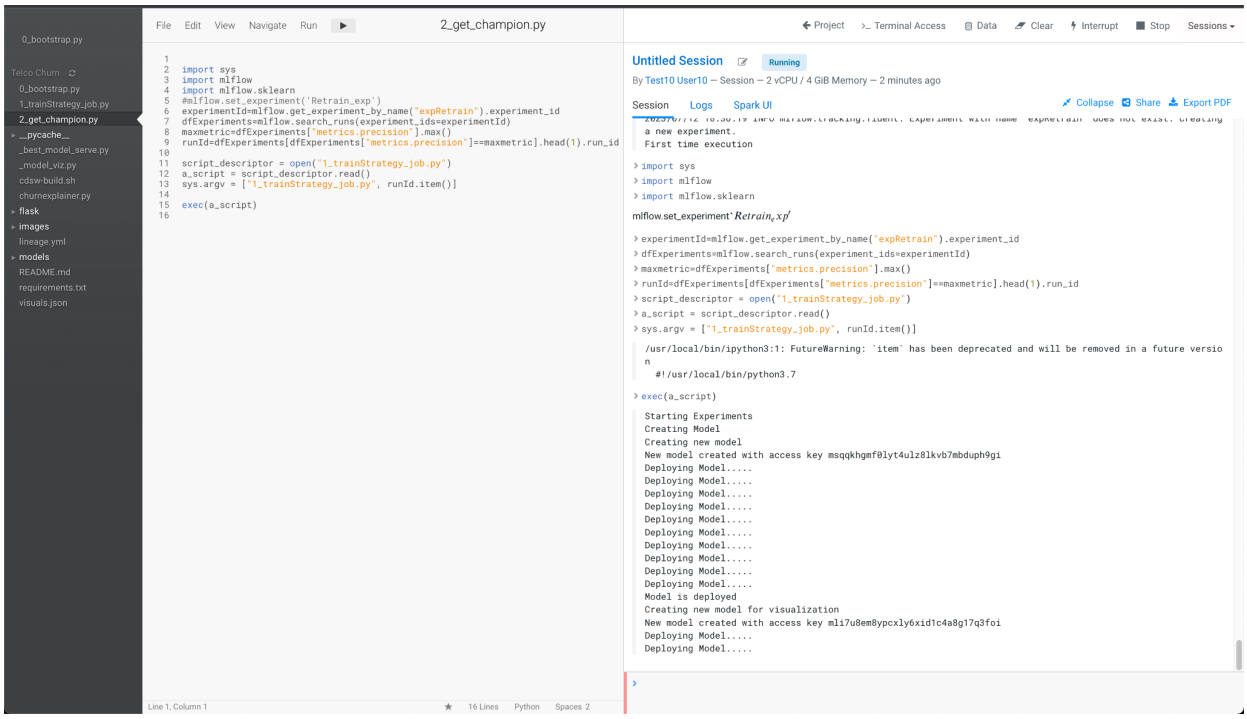
opção **Sessions** no menu à esquerda e, posteriormente, na única sessão que aparecerá na lista. Se você não adicionou um nome à sua sessão ao iniciá-la (etapa 6), ela deve se chamar *Untitled Session*.



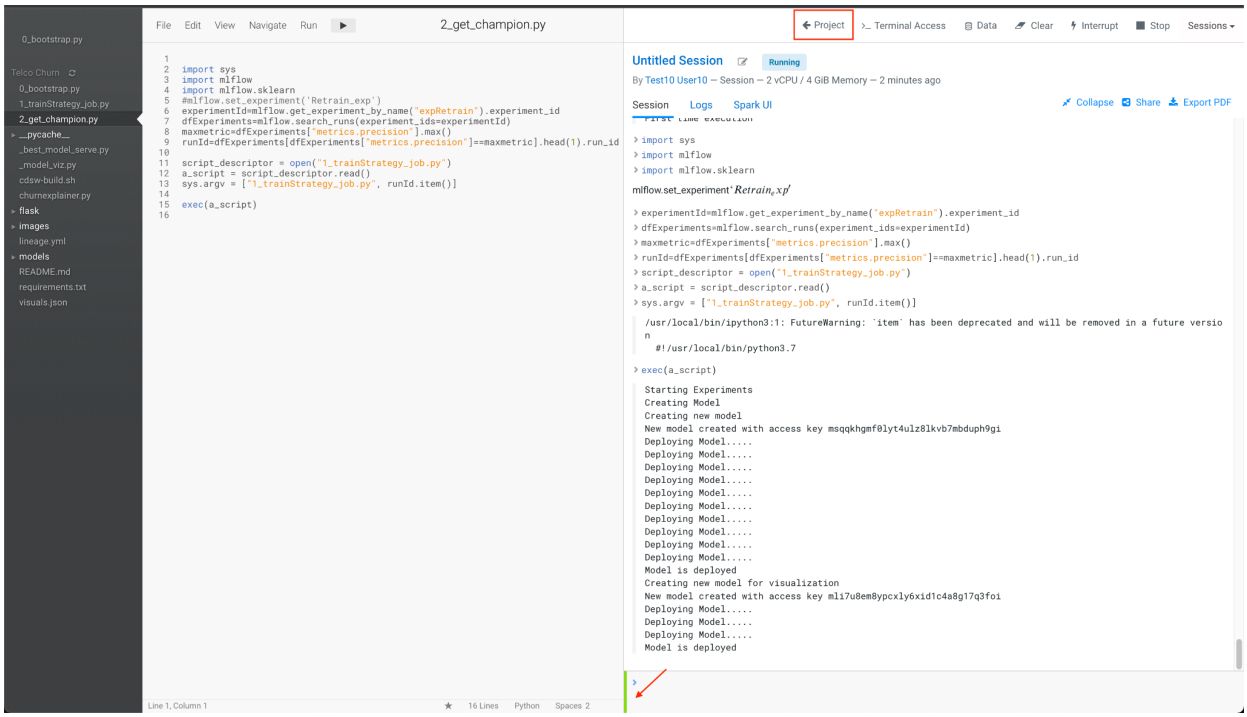
14. O terceiro e último script/código a ser executado é o **2_get_champion.py**. Este código Python pega o hiperparâmetro da execução do Experimento com melhor precisão e exibe dois Modelos na API REST, um para ser integrado à Visualização de Dados e outro para uso da unidade para chamadas. Selecione (apenas um clique) o arquivo na barra localizada no lado esquerdo da interface, isso fará com que o código apareça no editor. Após selecionar o arquivo, clique no botão ▶ para executar o código.



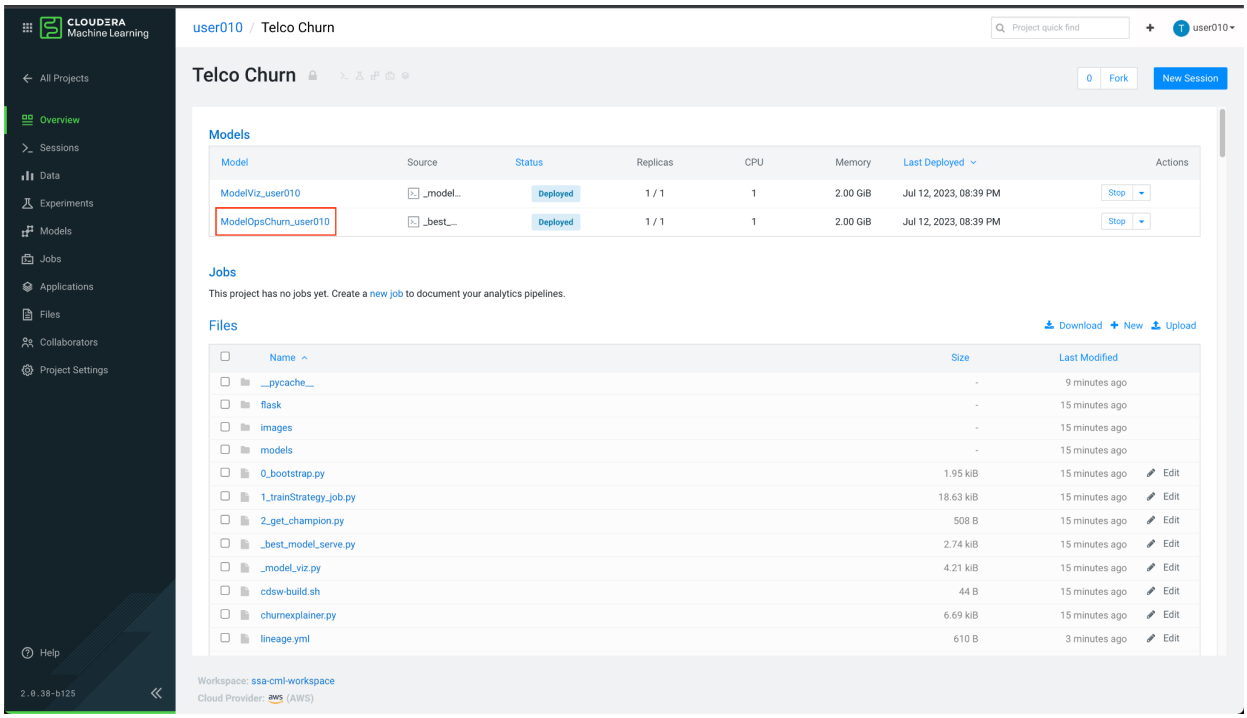
Após alguns segundos, você verá a seguinte mensagem “Deploying Model…” repetida várias vezes, e a barra de comandos inferior ficará vermelha.



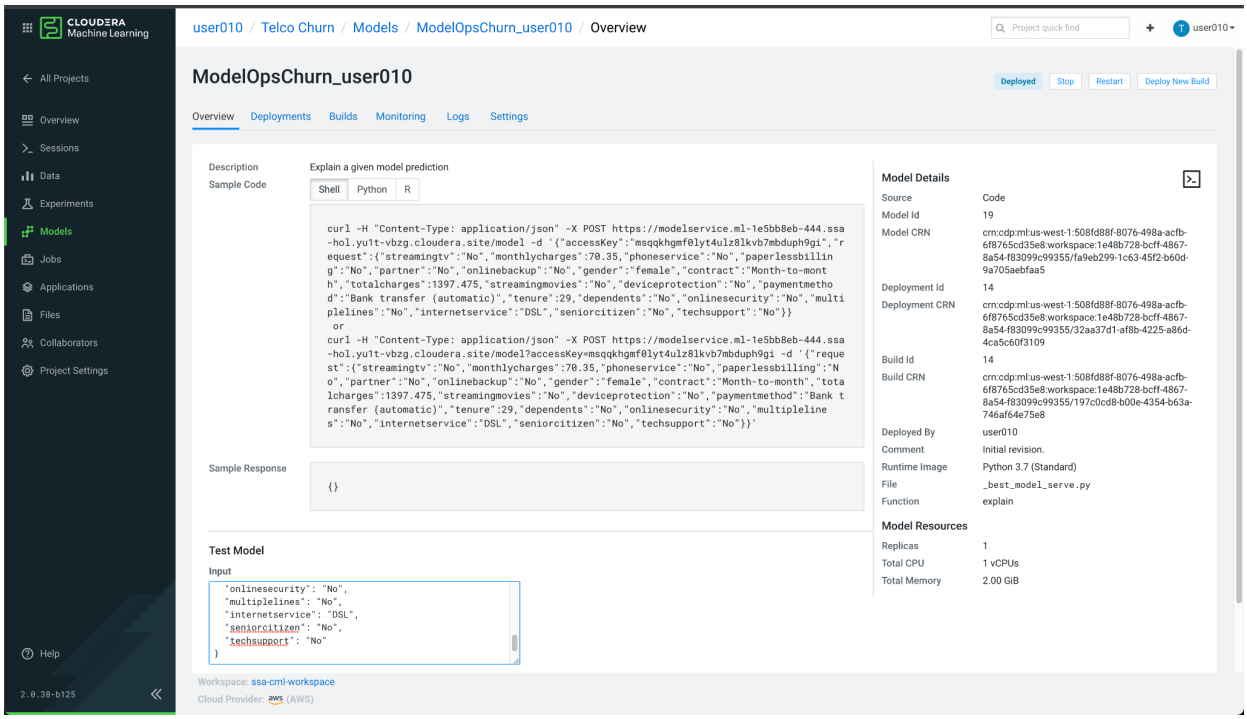
Após cerca de 2 minutos, a última mensagem deve ser "Model is deployed" e a barra ficará verde. Significa que a implantação dos dois Modelos está finalizada. Clique no botão **Project**, localizado na barra superior direita da sessão para retornar à página inicial do projeto.



15. Uma vez na página inicial do projeto, você verá os dois modelos exibidos. Clique naquele que começa com o nome **ModelOpsChurn.**



16. Aqui você verá as informações e Configurações do modelo na guia Visão geral.



Para testá-lo e fazer uma solicitação ao modelo, role a página para cima e clique no botão **Test** Este irá pegar o valor em formato JSON que está no campo **Input** e fará a chamada para o modelo. O que você vê no campo **Result** é a resposta do modelo no formato JSON. Se

desejar, você pode alterar alguns dos parâmetros do campo de entrada (por exemplo, alterar No para Yes), chamar o modelo novamente e verificar o valor do atributo *probability* da resposta, para ver se houve alguma mudança.