

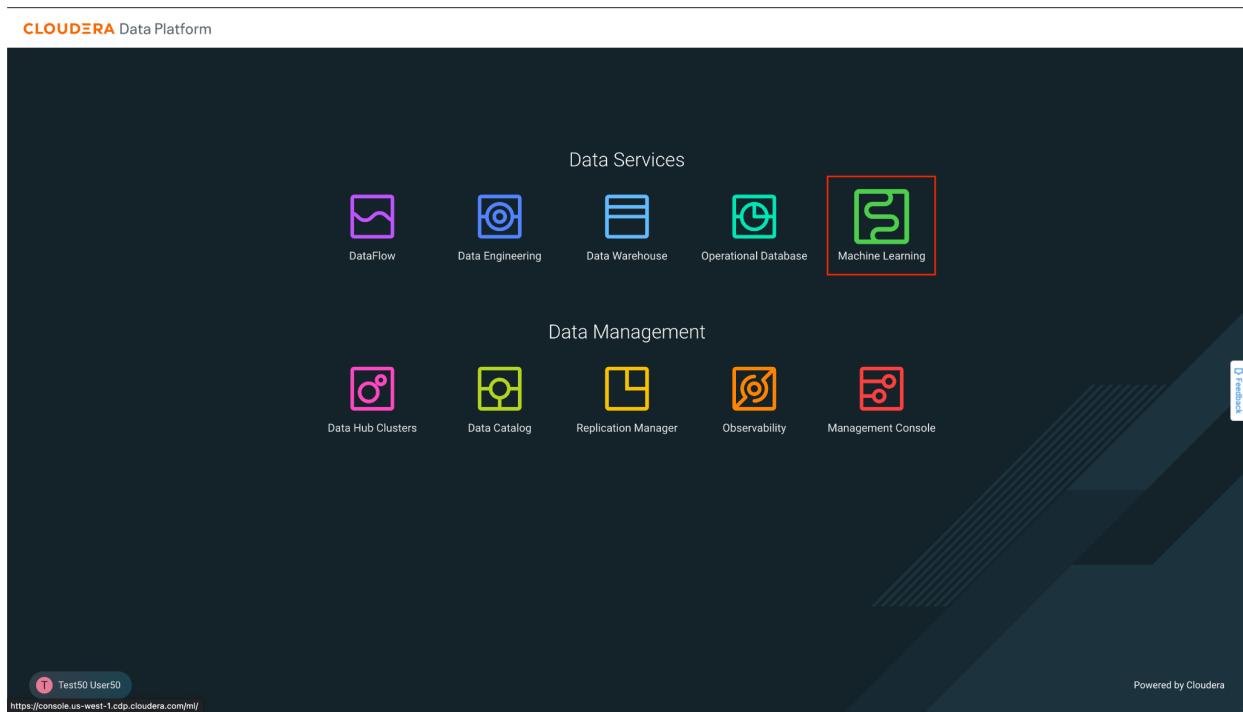
Ciclo de vida dos dados em CDP Public Cloud

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.

The screenshot shows the 'Machine Learning Workspaces' page. On the left, there's a sidebar with options like 'Workspaces', 'Workspace Backups', and 'Model Registries'. The main area has a table titled 'Machine Learning Workspaces' with columns: Status, Version, Workspace, Environment, Region, Creation Date, Cloud Provider, and Actions. A single row is listed: 'Ready' status, '2.0.38' version, 'ssa-cml-workspace' workspace name (which is highlighted with a red box), 'ssa-hol' environment, 'unknown' region, '07/07/2023 11:42 PM CEST' creation date, 'aws' cloud provider, and an 'Actions' button. At the bottom, it says 'Displaying 1 - 1 of 1'.

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**.

The screenshot shows the 'Projects' page. On the left, there's a sidebar with options like 'Sessions', 'Experiments', 'Models', 'Jobs', 'Applications', 'User Settings', 'AMPs', 'Runtime Catalog', 'Site Administration', and 'Learning Hub'. The main area has a table titled 'Projects' with columns: 'View Resource Usage Details' (with a green checkmark), 'Search Projects', 'Scope' (set to 'My Projects'), and 'Creator' (set to 'All'). Below this, there's a message: 'You currently don't have any projects' with a small cloud icon. A blue button labeled 'New Project' is highlighted with a red box. At the bottom, it says 'Workspace: ssa-cml-workspace' and 'Cloud Provider: aws (AWS)'.

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**

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**

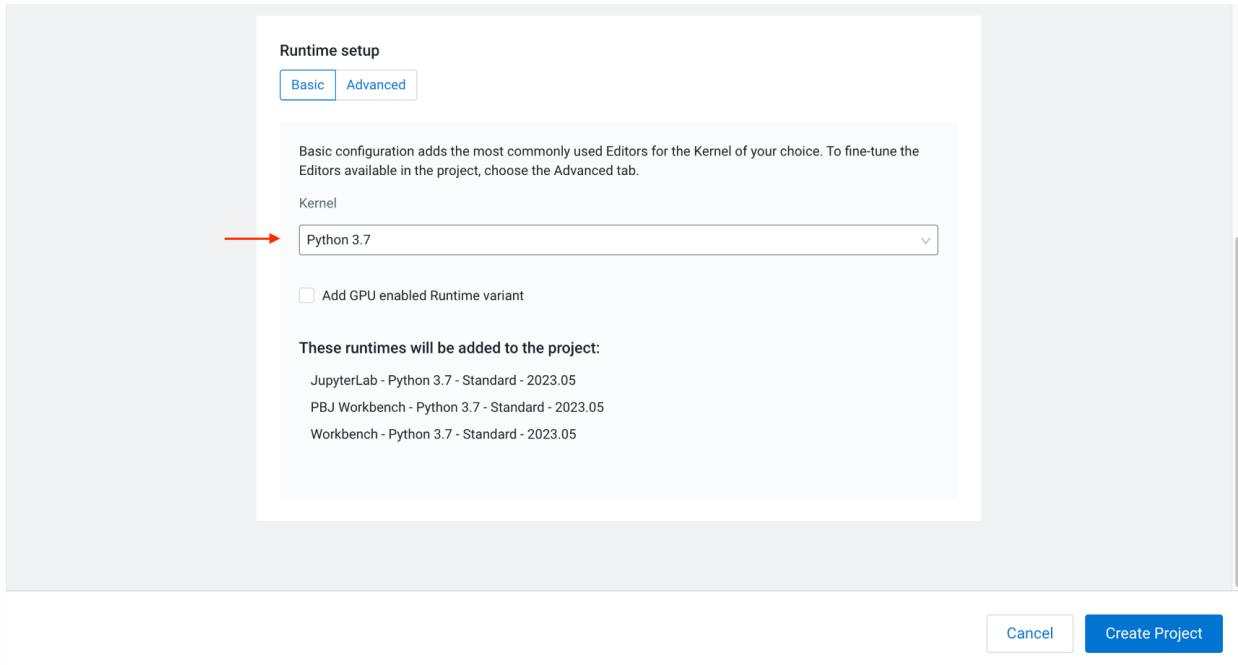
Provide the Git URL of the project to clone. Select the option that applies to your URL access.
 HTTPS SSH
https://github.com/campossalex/TelcoChurn

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

Runtime setup
Basic Advanced

Cancel **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.

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 of contents for a directory structure. The table includes columns for Name, Size, and Last Modified, with edit icons for each file. At the bottom of the table, there's a link to 'Show Hidden Files'. The bottom of the interface shows workspace and provider information: 'Workspace: ssa-cml-workspace' and 'Cloud Provider: AWS (AWS)'.

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

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

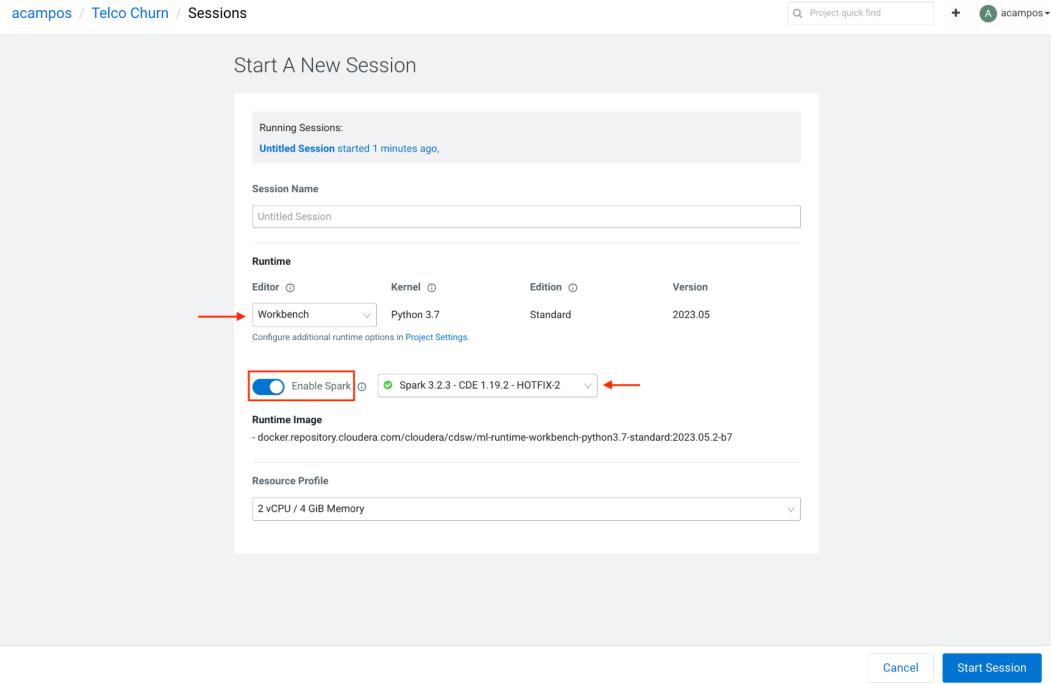
6. Ao fazer login, certifique-se de:

Selecionar **Workbench** na opção Editor.

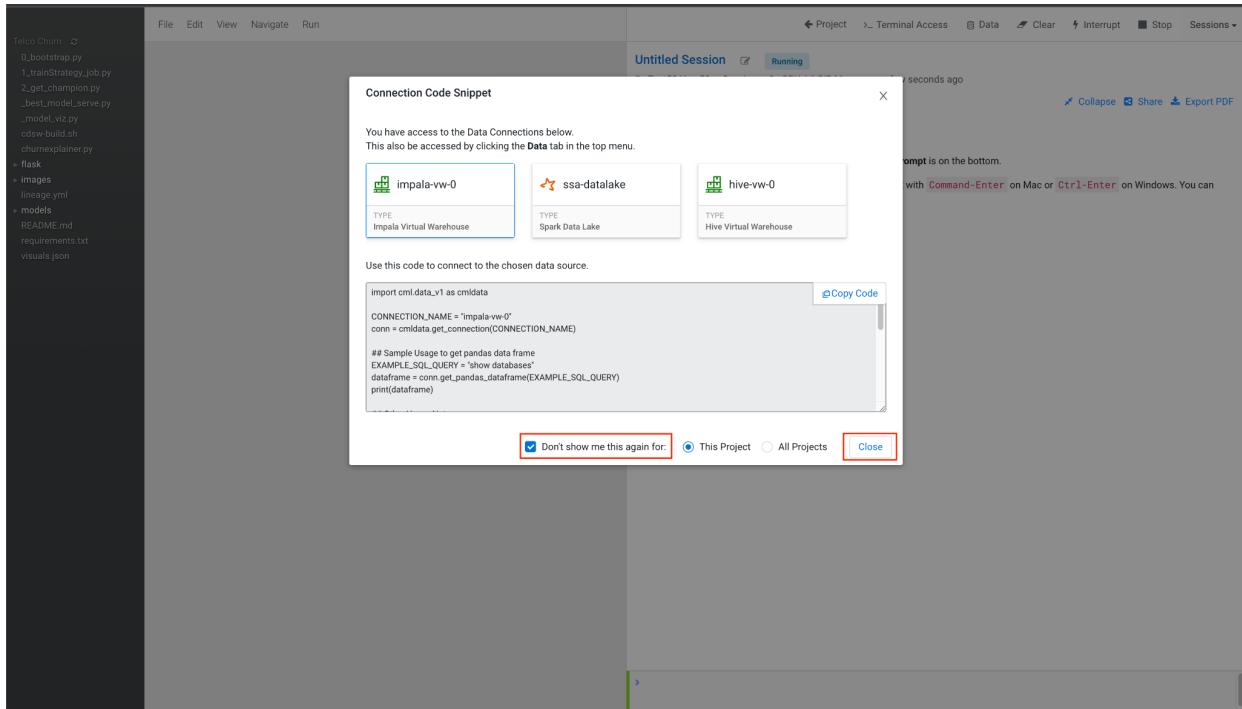
Habilitar **Spark**, marcando a opção correspondente.

Selecionar **Spark 3.2.x** na opção de versão do Spark.

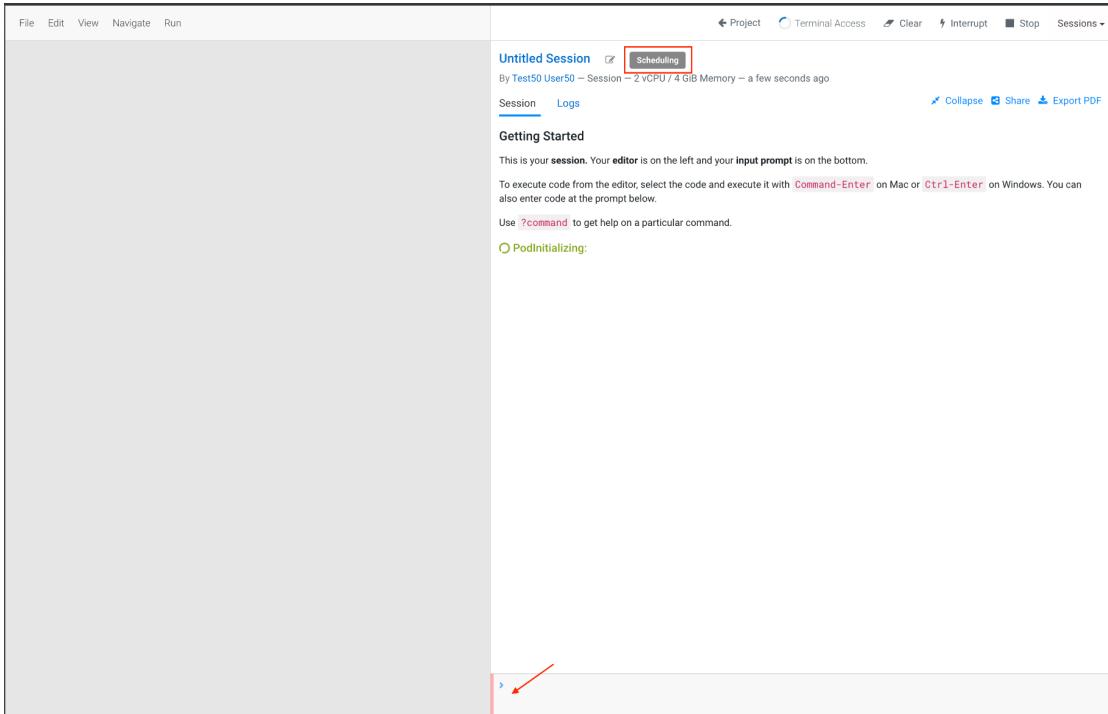
Clique em **Start Session**



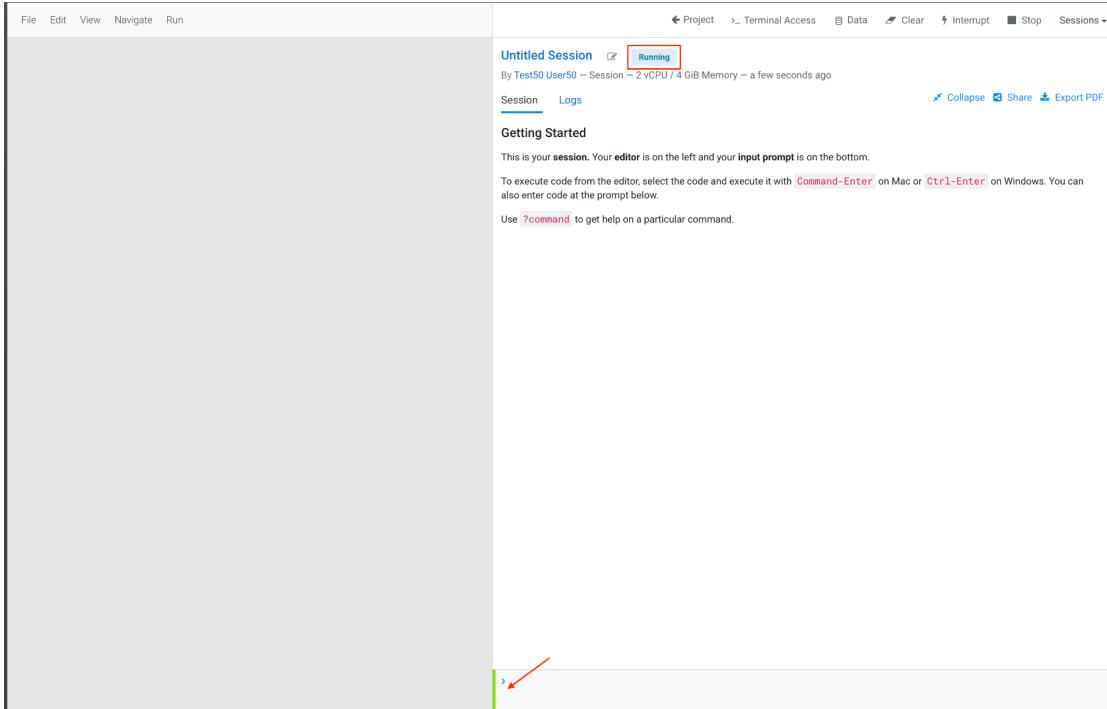
7. Ao iniciar a sessão pela primeira vez, ele perguntará se você deseja usar uma conexão de dados. Este projeto não precisa deste tipo de conexão. marcar o checkbox de **Don't show me this again for** e, em seguida, clique no botão **Close**.



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.

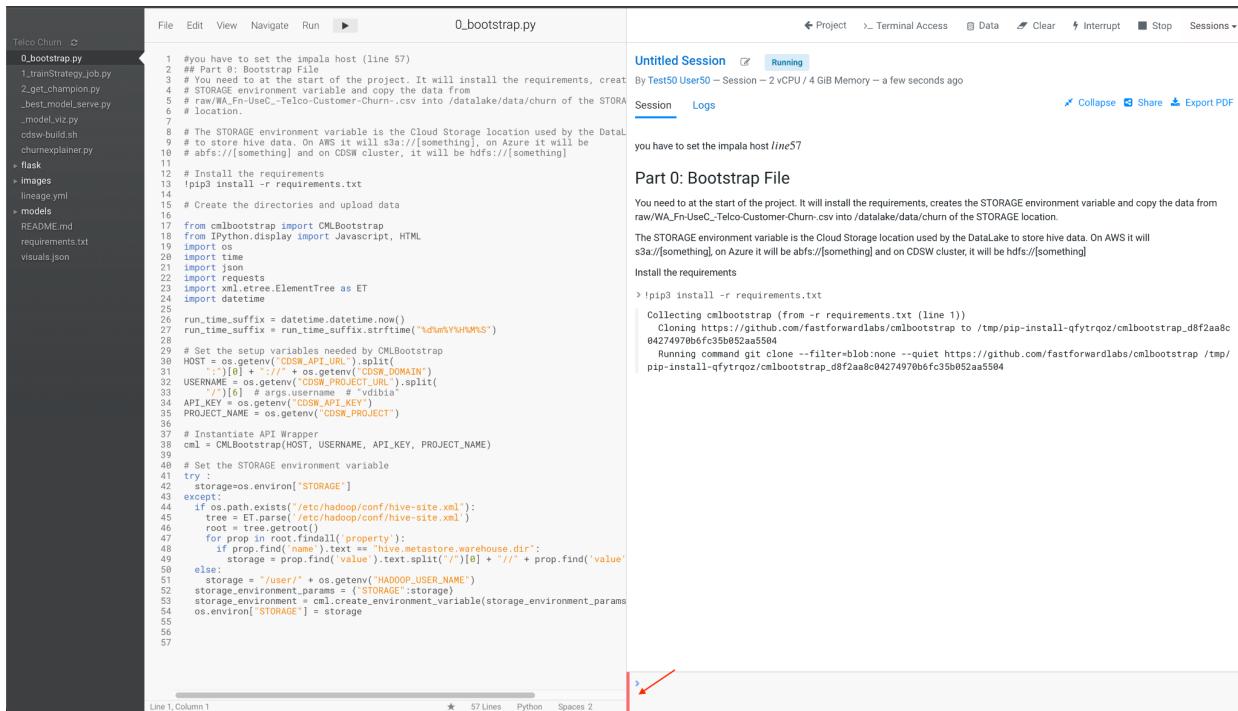
```

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 set at the start of the project. It will install the requirements, create
4 # STORAGE environment variable and copy the data from
5 # raw/NA_Fn-Use-Celco-Customer-Churn-.csv into /datalake/data/churn of the STORA
6 #
7 #
8 # The STORAGE environment variable is the Cloud Storage location used by the Data
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 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%b%Y%H%M%S")
28
29 # Set the python variables needed by CMLBootstrap
30 HOST = os.getenv("CDSW_API_URL").split(
31     ":" )[0] + "/" + os.getenv("CDSW_DOMAIN")
32 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
33     ":" )[0] + os.getenv("CDSW_PROJECT_NAME") + "liba"
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_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     storage=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.split('/')[1]
50             else:
51                 storage = "user/" + os.getenv("HADOOP_USER_NAME")
52         storage_environment_params = {'STORAGE':storage}
53         storage_environment = cml.create_environment_variable(storage_environment_params)
54         os.environ['STORAGE']= storage
55
56
57

```

Line 1, Column 1 ★ 57 Lines Python Spaces 2

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.



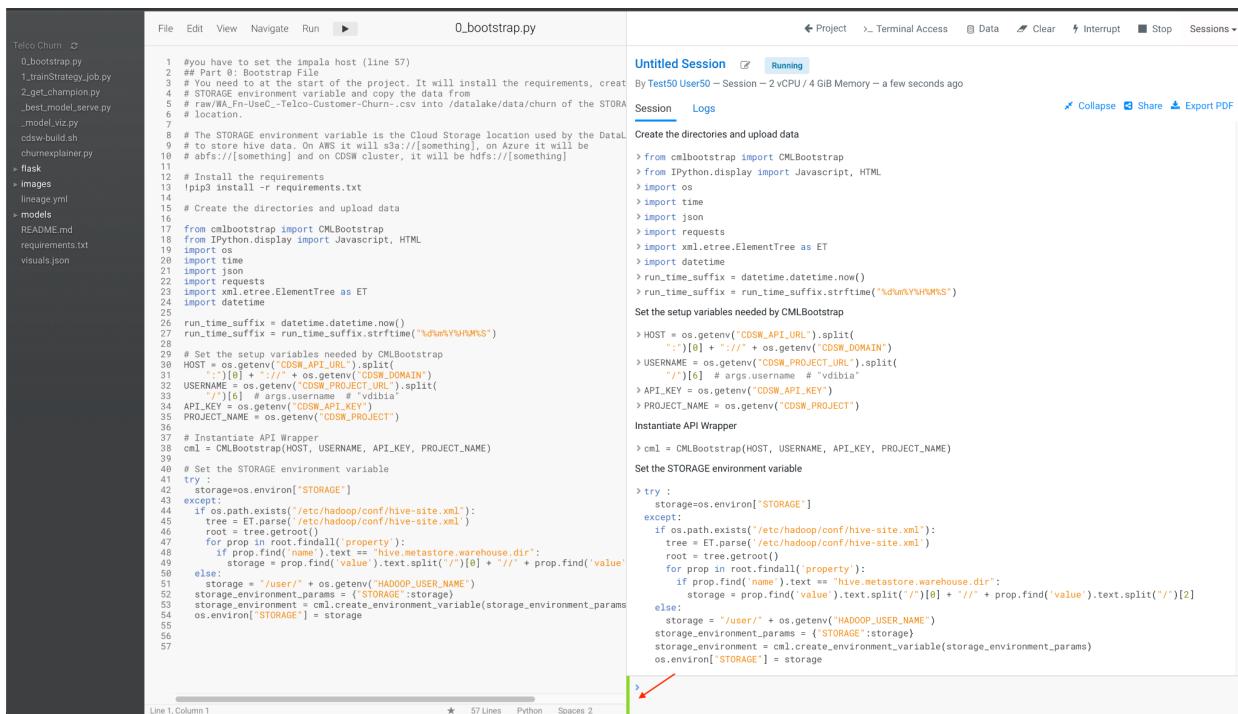
```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1_trainStrategy.job.py
2_get_champion.py
3_best_model_serve.py
4_model_viz.py
cdsw-build.sh
churnexplainer.py
> flask
> images
lineage.yml
> models
README.md
requirements.txt
visuals.json

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 # STORAG environment variable and copy the data from
5 # raw/MA_Fn-UseC_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 DataL
9 # to store hive data. On AWS it will s3a://[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%b%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSW_API_URL").split(
31     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
32 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
33     "/" )[6] # args.username # "vdibia"
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try :
42     storage=os.environ["STORAGE"]
43 except:
44     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
45         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('/')[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
Line 1, Column 1  ★ 57 Lines  Python  Spaces 2

```

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



```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1_trainStrategy.job.py
2_get_champion.py
3_best_model_serve.py
4_model_viz.py
cdsw-build.sh
churnexplainer.py
> flask
> images
lineage.yml
> models
README.md
requirements.txt
visuals.json

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 # STORAG environment variable and copy the data from
5 # raw/MA_Fn-UseC_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 DataL
9 # to store hive data. On AWS it will s3a://[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%b%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSW_API_URL").split(
31     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
32 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
33     "/" )[6] # args.username # "vdibia"
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try :
42     storage=os.environ["STORAGE"]
43 except:
44     if os.path.exists('/etc/hadoop/conf/hive-site.xml'):
45         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('/')[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
Line 1, Column 1  ★ 57 Lines  Python  Spaces 2

```

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.

The screenshot shows a Jupyter Notebook environment with two panes. The left pane displays a Python script named `1_trainStrategy.job.py`, which includes imports for numpy, pandas, sklearn, and various ML models. It also contains logic for reading data from a CSV file, splitting it into training and testing sets, and fitting a Random Forest Regressor. The right pane shows the execution of this script in an "Untitled Session" notebook. The session output indicates that the experiment 'expRetrain' does not exist and is being created. The session status bar at the bottom shows the date and time as 2023/07/12 18:36:19.

```
File Edit View Navigate Run Project Terminal Access Data Clear Interrupt Stop Sessions

0.bootstrappy
1.trainStrategy.job.py
2.get_champion.py
3._pycache_/_best_model.serve.py
4._model_viz.py
5.cdsw-build.sh
6.churnexplainer.py
7.flask
8.images
9.lineage.yml
10.models
11 README.md
12 requirements.txt
13 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 Collapseshare Export PDF

milflow.delete_experiment(experimentId)

time.sleep(20)
except:
    print("First time execution")

milflow.set_experiment('expRetrain')
valuesParam=[1, 10, 15]
for i in range(len(valuesParam)):
    with milflow.start_run(run_name=run_name+run_time_suffix+'_'+str(i)) as run:
        #with milflow.start_run() as run:
            # tracking run parameters
            milflow.log.param('compute', 'local')
            milflow.log.param('dataset', 'telco-churn')
            milflow.log.param('dataset_version', '2.0')
            milflow.log.param('alg', 'random forest')

            # tracking any additional hyperparameters for reproducibility
            n_estimators = valuesParam[i]
            milflow.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
            milflow.sklearn.log.model(rf, 'rf-base-line-model')

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

milflow.end_run()

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

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.

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, Data, Experiments (which is highlighted with a green arrow), Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main area is titled "user010 / Telco Churn Experiments". It features a search bar labeled "Search Experiments" and a table with columns: Name, Creator, Created At, and Last Updated. A single experiment named "expRetrain" is listed, created by "Test10 User10" on "07/12/2023 8:36 PM". At the bottom right of the table, it says "Displaying 1 - 1 of 1". There are also buttons for "New Experiment" and "BETA".

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.

The screenshot shows the details of the "expRetrain" experiment. The left sidebar remains the same. The main title is "user010 / Telco Churn Experiments / expRetrain". Below the title, it shows the Experiment Name as "expRetrain", Experiment ID as "pdj8-a6kp-bh2r-dehf", and Artifact Location as "/home/cdsw/experiments/pdj8-a6kp-bh2r-dehf". Under the "Notes" section, there is a link. The "Runs (3)" section contains a table with three rows of data. The columns are: Status, Start Time, Run Name, Duration, User, Source, Version, Models, algo, compute, dataset, and precision. The data is 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. 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*.

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with various project management options like Overview, Sessions (which has a red arrow pointing to it), Data, Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main area is titled "user010 / Telco Churn Sessions". It displays a table of sessions with the following data:

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

At the bottom of the interface, it says "Displaying 1 - 1 < 1 > 25 / page". The footer indicates the workspace is "ssa-cml-workspace" and the cloud provider is "AWS (AWS)".

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.

```

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

```

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

```

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
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 Model
Creating Model
New model created with access key msqqkhgmf0lyt4ulz8lkvb7mbduph9gi
Deploying Model.....
Model is deployed
Creating new model for visualization
New model created with access key mli7u0em8ypcxl6xidlc4s8gj7q3foi
Deploying Model.....
Deploying Model.....

```

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.

The screenshot shows a Jupyter Notebook environment. On the left, a sidebar displays project files including `0_bootstrap.py`, `Telco Churn`, `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`, and `visuals.json`. The main area shows a code cell with the following Python script:

```
1 import sys
2 import mlflow
3 import mlflow.sklearn
4 #mlflow.set_experiment('Retrain_exp')
5 experimentId=mlflow.get_experiment_by_name("expRetrain").experiment_id
6 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
7 maxmetric=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 script_descriptor = open("1_trainStrategy.job.py")
11 a_script = script_descriptor.read()
12 sys.argv = ["1_trainStrategy.job.py", runId.item()]
13 exec(a_script)
14
15
16
```

To the right, a terminal window titled "Untitled Session" shows the execution of the script. The output indicates the creation of a new model and its deployment:

```
Untitled Session [Running]
By Test10 User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago
Session Logs Spark UI
> import sys
> import mlflow
> import mlflow.sklearn
mlflow.set_experiment'Retrain_exp'
> experimentId=mlflow.get_experiment_by_name("expRetrain").experiment_id
> dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
> maxmetric=dfExperiments["metrics.precision"].max()
> runId=dfExperiments[dfExperiments["metrics.precision"]==maxmetric].head(1).run_id
> script_descriptor = open("1_trainStrategy.job.py")
> a_script = script_descriptor.read()
> sys.argv = ["1_trainStrategy.job.py", runId.item()]
/usr/local/bin/python3.7: FutureWarning: 'item' has been deprecated and will be removed in a future version
n
#!/usr/local/bin/python3.7
> exec(a_script)
Starting Experiments
Creating Model
Creating New Model
New model created with access key msqqkhgmf0lyt4ulz8ikv7mbdph9gi
Deploying Model...
Deploying Model....
Deploying Model.....
Model is deployed
Creating new model for visualization
New model created with access key ml17u8em8ypcxly6xid1c4a8g17q3foi
Deploying Model...
Deploying Model....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Model is deployed
```

A red arrow points to the bottom of the terminal window, highlighting the final message "Model is deployed".

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

The screenshot shows the Cloudera Machine Learning interface for the 'Telco Churn' project. In the 'Models' section, two models are listed: 'ModelViz_user010' and 'ModelOpsChurn_user010'. Both are in a 'Deployed' status with 1/1 replicas, 1 CPU, and 2.00 GiB memory, last deployed on Jul 12, 2023, at 08:39 PM. The 'Jobs' section indicates no jobs have been created. The 'Files' section shows a directory structure with files like .pycache, flask, images, models, and various Python scripts, along with their sizes and last modified times.

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

The screenshot shows the 'Overview' page for the 'ModelOpsChurn_user010' model. The 'Model Details' section provides information such as Model Id (19), Model CRN (cm:cdp:ml:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e48b728-bcf4-4867-8a54-f8309c99355/faeb2991c63-45f2-b60d-9a70Saefba5), Deployment Id (14), Deployment CRN (cm:cdp:ml:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e48b728-bcf4-4867-8a54-f8309c99355/32aa37d1-af8b-4225-a86d-4ca5c60f3109), Build Id (14), Build CRN (cm:cdp:ml:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:workspace:1e48b728-bcf4-4867-8a54-f8309c99355/197cc0d8-b00e-4354-b63a-746af64e75e8), Deployed By (user010), and Comment (Initial revision). The 'Test Model' section contains input and output JSON fields, and the 'Model Resources' section shows 1 replica, 1 vCPUs, and 2.00 GiB total memory.

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.

The screenshot shows the Cloudera Machine Learning interface for a project named 'user010'. The left sidebar has a green highlight on the 'Models' section. The main area is titled 'Overview' for 'ModelOpsChurn_user010'. It shows a 'Test Model' section with an input JSON object:

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

Below the input is a 'Test' button (highlighted with a red box) and a 'Reset' button. The 'Result' section shows a status of 'success' with a green dot. The response JSON is displayed:

```
{
  "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"
}
```

The 'Replica ID' field contains 'modelopschurn-user010-19-14-6c5d7947ff-52kzg'. At the bottom, it says 'Workspace: ssa-cml-workspace' and 'Cloud Provider: AWS (AWS)'.