Data Lifecycle on 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 Data Warehouse from CDP PC Home:

A screenshot of a computer

Description automatically generated

1. This is a screen to select a Workspace, which is compute resource allocation for Data Science related jobs. Click on the only Workspace that appears.

A screenshot of a computer

Description automatically generated

1. Once in the Workspace, you should see the following interface. Here are the projects you have created. It is time to create a new project. Click on **Create a new project**.

A screenshot of a computer

Description automatically generated

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

**Project Name:** <username>-Telco Churn (Example: ***psemeta01-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>

Keep the rest of the settings the same. Click the button **Create Project.**

A screenshot of a computer

Description automatically generated

Click on **Advanced Options.**

Select -

Editor – **Workbench**

Kernel – **Python 3.7**

Edition – **Standard**

Version – **2024.02**

Click on **Add Runtime.**

A screenshot of a computer

Description automatically generated

The Runtime gets added as you can see below.

Click on **Create Project**.

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

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.

A screenshot of a computer

Description automatically generated

1. Make sure to select the values as shown in the screenshot. We select the previously added Runtime.

Slide the **Enable Spark** option and then select **Spark 3.2.3-CDE.xxxxx.**

**Note:** Spark 3.x is needed for Iceberg related tables and will give error if not chosen.

Click on **Start Session.**

A screenshot of a computer

Description automatically generated

1. 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 for**, and then click the button **Close**, so this window will not appear anymore.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. 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 computation 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.

A screenshot of a computer

Description automatically generated

1. 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.

A screenshot of a computer

Description automatically generated

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.

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

1. 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 records 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.

In the code change the line that has the value of **DATABASE** to your username ex – **psemeta01**

A screenshot of a computer

Description automatically generated

The screenshot below shows the changed name for the database.

Click on the button  to run the code. Once the execution is finished (approximately 1-2 minutes).

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Once it runs successfully as indicated by green bar below click on the button **Project**, located in the upper right bar of the session to go back to the project home.

A screenshot of a computer

Description automatically generated

1. 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.

A screenshot of a computer

Description automatically generated

1. On this screen you will see the three runs of this experiment. Look at the last column, where

**precision** attribute displays. This is the precision that each hyper parameter is delivering.

A screenshot of a computer

Description automatically generated

1. 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 later the only session that will appear in the list. If you didn't name your session when you started it should be called *Untitled Session*.

A screenshot of a computer

Description automatically generated

1. 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 better 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.

A screenshot of a computer

Description automatically generated

After a few seconds, you will see the following message “Deploying Model…” repeated several times, and the bottom command bar will be red initially and then will become green on successful deployment.

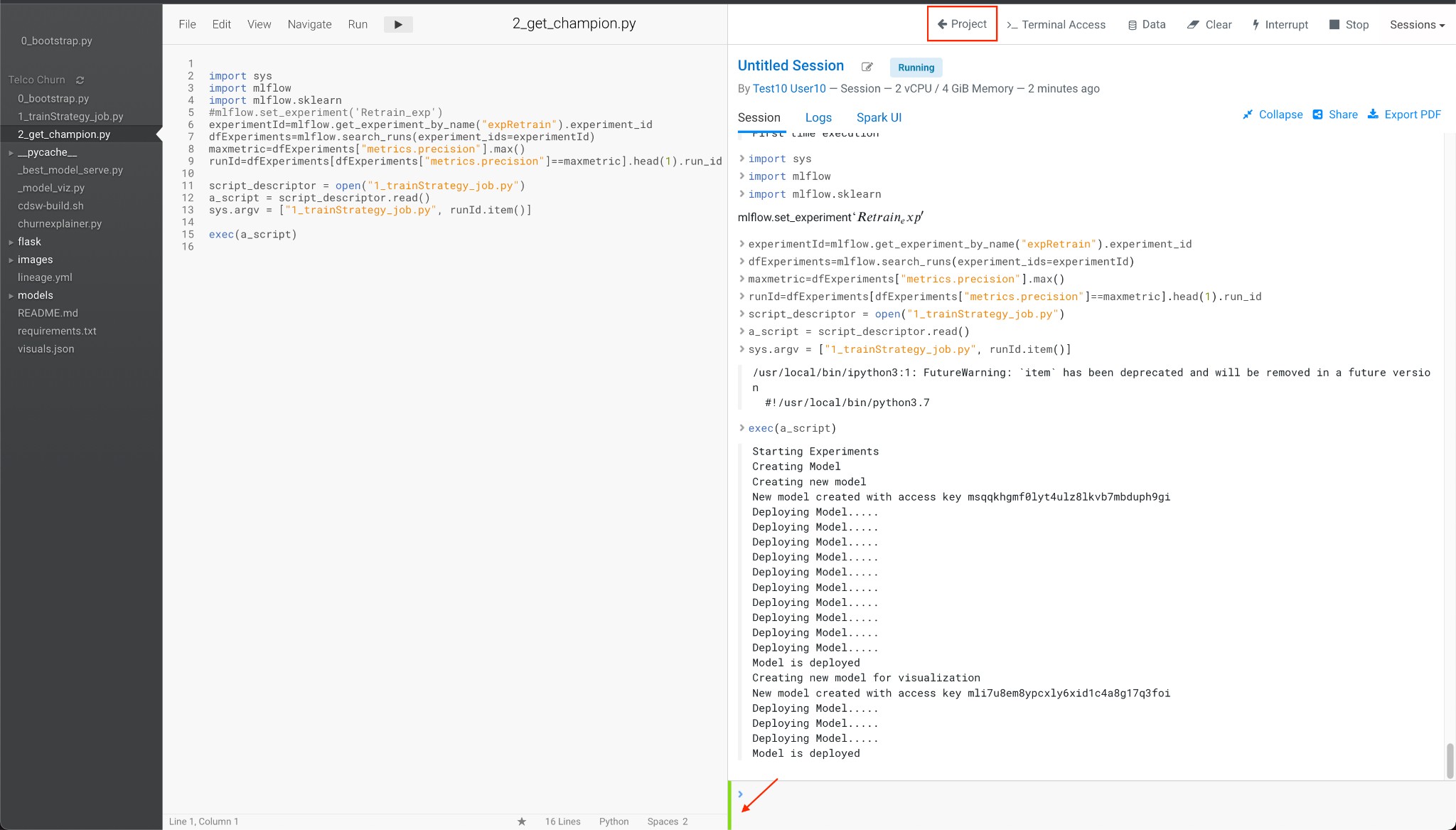
A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

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.

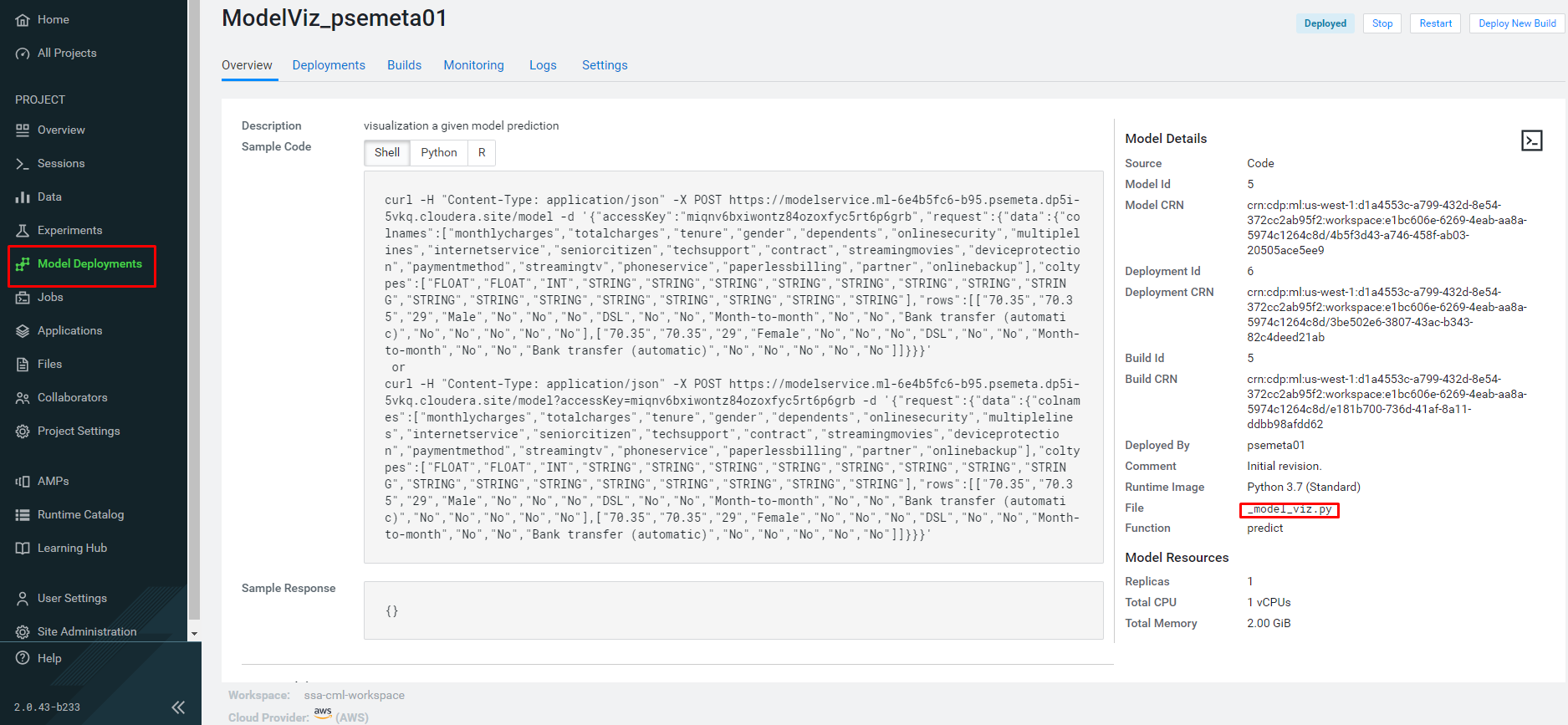


1. Once on the home page of the project, you will see the Models displayed, which are two. Click on the one that starts with **ModelOpsChurn.**

A screenshot of a computer

Description automatically generated

1. 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 *Not* 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.

A screenshot of a computer

Description automatically generated

At this point note the value of the **url** and **accessKey** somewhere as you will use it to call the churn related values given by model in the Data viz application.

url - <https://modelservice.ml-6e4b5fc6-b95.psemeta.dp5i-5vkq.cloudera.site/model>

accessKey - miqnv6bxiwontz84ozoxfyc5rt6p6grb

