

Cloudera ACE Machine Learning Workshop

Customer Churn Hands On Labs Guide

Objective

In this exercise we will implement an end-to-end machine learning workflow and cover:

- Data ingest
- Data exploration
- Model building and training
- Model serving
- Business applications and model operations

Business Use Case

In this Hands On Lab you will create a model to predict customer churn and present model-driven insights to your stakeholders.

You will use an interpretability technique to make your otherwise “black box model” explainable in an interactive dashboard. A mathematical explanation is beyond the scope of this lab but if you are interested in learning more we recommend the [Fast Forward Labs Report on Model Interpretability](#).

Finally, you will use basic ML Ops techniques to productionize and monitor your model.

Index

Part 1: Configure and deploy the Workshop Content as an AMP (Applied ML Prototype)

Part 2: Churn Model Project Overview

Part 3: CML Sessions and Workbench

Part 4: CML Jobs and Experiments

Part 5: CML Models and Applications

Part 6: CML Model Operations

Part 7: Model Lineage Tracking

Step by Step Instructions

Part 1: Configure and deploy the Workshop Content as an AMP

AMPs (Applied Machine Learning Prototypes) are reference Machine Learning projects that have been built by Cloudera Fast Forward Labs to provide quickstart examples and tutorials. AMPs are deployed into the Cloudera Machine Learning (CML) experience, which is a platform you can also build your own Machine Learning use cases on.

The screenshot shows the Cloudera Machine Learning (CML) interface. On the left, there's a sidebar with a dark theme containing links for Projects, Sessions, Experiments, Models, Jobs, Applications, User Settings, and AMPs (which is currently selected, indicated by a green bar). Below these are Runtime Catalog and Site Administration. The main content area is titled "Applied ML Prototypes" and contains several cards for different machine learning projects:

- Churn Modeling with scikit-learn**: Tags: CHURN PREDICTION, LOGISTIC REGRESSION. Description: "cuts ad spending on Facebook org amid growing boycott: WSJ org".
- Deep Learning for Image Analysis**: Tags: COMPUTER VISION, IMAGE ANALYSIS. Description: "Deep learning models based on your search for KFC chicken".
- Question Answering with Wikipedia**: Tags: WIKIPEDIA, THE FREE ENCYCLOPEDIA. Description: "What is the air speed velocity of an unladen swallow?".
- Deep Learning for Question Answering**: Tags: AUTOMATED QUESTION ANSWERING, EXTRACTIVE QUESTION AN. Description: "Deep learning models for question answering".
- Few-Shot Text Classification**: Tags: NLP, FEW-SHOT LEARNING. Description: "Few-shot learning for text classification".
- Canceled Flight Prediction**: Tags: BINARY CLASSIFICATION, XGBOOST. Description: "Predicted Value: Not canceled, Probability: 0.81".
- Streamlit**: Tags: STREAMLIT, APPLICATIONS. Description: "Streamlit logo".

Navigate to the Machine Learning tile from the CDP Menu. Workspaces are the heart of the Cloudera Machine Learning (CML).

The screenshot shows the Cloudera Data Platform (CDP) Machine Learning Workspaces page. The table has the following columns:

- Status
- Workspace
- Environment
- Region
- Creation Date
- Cloud Provider
- Actions

The single row shown is for a workspace named "cdpmluser31-ml" which is marked as "Ready". The "Cloud Provider" column shows "aws AWS". At the bottom right, there are buttons for "Provision Workspace" and "Actions". Below the table, it says "Displaying 1-1 of 1" and "25 / page".

A Workspace is a cluster that runs on a kubernetes service to provide teams of data scientists to develop, test, train, and ultimately deploy machine learning models. It is designed to deploy a small number of infra resources and then autoscale compute resources as needed when end users implement more workloads and use cases.

Click into the Workspace by clicking the Workspace name.

The screenshot shows the 'Machine Learning Workspaces' page. The left sidebar includes options like Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters, ML Workspaces (which is selected), Classic Clusters, and Global Settings. The main area has a search bar and filters for Status, Environment, Region, and Cloud Provider (AWS). A table lists workspaces with details such as creation date and region. The workspace 'cdprivaluser31' is highlighted in blue.

You can visualize all of the Projects and Resources that are part of the Projects page. Next we will create a Project where we will deploy an AMP. Click on “New Project”

The screenshot shows the 'Projects' page. The left sidebar includes Options, Sessions, Experiments, Models, Applications, User Settings, and Site Administration. The main area shows active workloads and resource usage. A table lists projects with details like creator and creation time. The project 'Full-test-2' is selected.

Create a new Project as shown below:

Project Name: Username Churn Project

Project Visibility: Private

Initial Setup: AMP's

Git URL: https://github.com/cloudera/CML_AMP_Churn_Prediction.git

* Project Name

My Churn ML Project

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

Applied ML Prototypes provide components to create a complete project. They may include jobs, models and experiments.

- Provide Git URL of your AMPs ⓘ

`https://github.com/cloudera/CML_AMP_Churn_Predictio`

- Upload a zip or tar file

Choose File Browse

Scroll down and make sure you select the “Basic” runtime with the Python 3.7 kernel

Runtime setup

[Basic](#) [Advanced](#)

Basic configuration adds the most commonly used Editors for the Kernel of your choice. To fine-tune the Editors available in the project, choose the Advanced tab.

Kernel

Python 3.7

Add GPU enabled Runtime variant

These runtimes will be added to the project:

- Workbench - Python 3.7 - 1 - 1
- Workbench - Python 3.7 - Java11 - 1
- JupyterLab - Python 3.7 - Standard - 2022.04
- PBJ Workbench - Python 3.7 - Tech Preview - 2022.04
- Workbench - Python 3.7 - Standard - 2022.04
- JupyterLab - Python 3.7 - Standard - 2021.12
- Workbench - Python 3.7 - Standard - 2021.12

Next, in order to deploy the AMP we will provide a few Environment Variables used throughout the telco churn Project.

Configure Project with the following:

DATA_LOCATION: data/churn_prototype

HIVE_DATABASE: default

HIVE_TABLE: churn_prototype_username

Default Engine: Runtime

Confirm 'Execute AMP setup steps' is checked

The latest version of the AMP has been tested for CML Runtimes with Python 3.7 and Spark 2.4.7. If the workspace does not have the exact runtime that was tested you may get a warning.

However, you can still deploy the project with other runtimes. For example, you can deploy the project with Spark 2.4.7, CDP 7.2.11, CDE 1.13 HotFix 2.

Configure Project: My Churn ML Project

AMP Name: ML Churn Prototype (v2)

Prototype to demonstrate building a churn model on CML

DATA_LOCATION	data/churn_prototype	x	Relative path that will be used to store the data used for this prototype. This should be a location you have write access to, and which is suitable for non-production data.
HIVE_DATABASE	default	x	Name of the Hive database that will be used to create the Hive table used for this prototype. This should be a Hive database you have write access to, and which is suitable for non-production data.
HIVE_TABLE	churn_prototype	x	Name of the Hive table that will be created and populated with the data used for this prototype. If the table already exists, the prototype will assume it already contains the data for this prototype.

Default Engine:

[Runtime](#) [Engine](#)

Runtime

Editor ⓘ	Kernel ⓘ	Edition ⓘ	Version
Workbench	Python 3.7	Standard	2021.12

Enable Spark ⓘ [Spark 2.4.7 - CDP 7.2.11 - CDE 1.13 - HO...](#)

Launch Project

Setup Steps

Execute AMP setup steps

[Cancel](#) [Launch Project](#)

Notice the AMP process consists in a number of automated steps for project setup. These include cloning all project artifacts from the public GitHub repository, setting Project environment variables, and deploying all CML Sessions, Jobs, etc in order for the Project setup steps to complete. The process is powered by a simple yaml file and everyone can build their own AMP by following [these instructions](#).

The AMP Setup process should take 10 minutes. This is a great time to navigate back to the Workspace homepage and [explore more AMPs](#) or, if you prefer, to simply grab a cup of coffee.

AMP Setup Steps

AMP Name: ML Churn Prototype (v1)

Prototype to demonstrate building a churn model on CML

Completed 3 of 10 steps

✓ Step 1 Job to install dependencies, set environment variables, and upload data [View details](#) completed 8/2/2021 11:31 AM

✓ Step 2 Running job to install dependencies [View details](#) completed 8/2/2021 11:34 AM

```
the qualifiedName of the hive_table object representing
    metadata:                                     #
this is a predefined key for additional metadata           #
    query: "select * from historical_data"            #
suggested use case: query used to extract training data   #
    training_file: "code/4_train_models.py"          #
suggested use case: training file used                 #
"""
with open("lineage.yml", "w") as lineage:
    lineage.write(yaml_text)
```

✓ Step 3 Job to ingest data into our Hive table [View details](#) completed 8/2/2021 11:34 AM

⌚ Step 4 Running job to ingest data [View details](#) started 8/2/2021 11:35 AM

```
#####
CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
(C) Cloudera, Inc. 2021
All rights reserved.
```

Applicable Open Source License: Apache 2.0

NOTE: Cloudera open source products are modular software products
made up of hundreds of individual components, each of which was

⌚ Step 5 Job to train models not yet started

Part 2: Churn Model Project Overview

With the AMP used to create our CML Project, we can now explore what was deployed for us.

ahansen / Churn Modeling with scikit-learn - ahansen

Churn Modeling with scikit-learn - ahansen 1 running

Build an scikit-learn model to predict churn using customer telco data.

Project creation succeeded! [View status page](#)

Step 10 of 10 Start Application View details Open

completed 8/2/2021 11:42 AM

Models

Model	Status	Replicas	CPU	Memory	Last Deployed	Actions
Churn Model API Endpoint	Deployed	1 / 1	1	2.00 GiB	Aug 2, 2021, 11:41 AM	Stop

Jobs

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Churn Model	1 / 0	00:12	Success	38 minutes ago	
Ingest data	1 / 0	00:32	Success	39 minutes ago	
Install dependencies	1 / 0	02:48	Success	42 minutes ago	

Files

Name	Size	Last Modified
code	-	37 minutes ago
flask	-	31 minutes ago
images	-	42 minutes ago
models	-	42 minutes ago
raw	-	38 minutes ago
cdsw-build.sh	45 B	42 minutes ago
LICENSE.txt	9.94 kB	42 minutes ago
lineage.yml	673 B	39 minutes ago

Workspace: **ahansen-amp-1**
Cloud Provider: **aws (AWS)**

Overview gives you access to all the features of a CML project. We will only have files copied from the Github repo currently. Initially it is good to start on the management components of a project.

trial31_admin / Telco_churn

Telco_churn

This project has no jobs yet. Create a new job to document your analytics pipelines.

Models

Jobs

Files

Name	Size	Last Modified
bootstrap.py	2.22 kB	a few seconds ago
1.data_ingest.py	7.41 kB	a few seconds ago
2.data_exploration.ipynb	602.63 kB	a few seconds ago
3.model_building.ipynb	71.53 kB	a few seconds ago
4.train_models.py	9.56 kB	a few seconds ago
5.model_explaination.py	8.10 kB	a few seconds ago
6.application.py	7.74 kB	a few seconds ago
7a_ml_ops_simulation.py	7.96 kB	a few seconds ago
7b_ml_ops_visual.py	3.59 kB	a few seconds ago
8.build_project.py	8.72 kB	a few seconds ago
cdsw-build.sh	44 B	a few seconds ago
churnexplainer.py	6.69 kB	a few seconds ago
leverage.yml	535 B	a few seconds ago
model_metrics.db	828.00 kB	a few seconds ago
README.md	11.95 kB	a few seconds ago
requirements.txt	175 B	a few seconds ago

Collaborators:

For our demo we aren't adding additional collaborators.

You can give access to other users with certain permissions for the encompassing project so teams of users can collaborate together. You can set up Admins, Contributor, Operator, and Viewer permissions.

This screenshot shows the CloudERA Machine Learning interface. On the left, there is a sidebar with the following navigation options: Overview, Sessions, Experiments, Models, Jobs, Applications, Files, Collaborators (which is currently selected), and Project Settings. The main content area displays a message stating, "This project is **private**. Only collaborators can view and edit this project. [Change Settings](#)". Below this, there is a section titled "Add Collaborator" with a note: "Email is not configured. Please contact your administrator." A search bar allows you to "Search by name, username, or email..." and an "Add" button. A table lists one collaborator: "Collaborator" (trial31_admin) and "Permission" (Admin). A yellow warning box at the bottom states: "Granting Admin or Contributor permission to other users may have security impact since it gives them full access to your project files and running sessions."

Project Settings:

Taking a look at Project Settings, this is where you can define several options for the current project. You have the ability to define different engines where your code in CML will run. There are project variables that can be defined and used throughout your code. SSH tunnels can also be configured to connect to other services as needed. More details can be found in our docs [here](#).

Please do not change the default Project Settings

This screenshot shows the CloudERA Machine Learning interface with the URL "trial31_admin / Telco_churn / Settings". The sidebar on the left is identical to the previous screenshot. The main content area is titled "Project Settings" and includes tabs for Options (which is selected), Runtime/Engine, Advanced, Tunnels, and Delete Project. The "Project Name" field contains "Telco_churn". The "Description" field has the placeholder "Description". Under "Visibility", the "Private" radio button is selected, with the note: "Only collaborators can view or edit the project." The "Public" radio button is also present with the note: "All authenticated users can view this project. Collaborators can also edit the project." At the bottom is a blue "Update Project" button.

trial31_admin / Telco_churn / Settings / Engine

All Projects

Overview Sessions Experiments Models Jobs Applications Files Collaborators Project Settings

Project Settings

Options Runtime/Engine Advanced Tunnels Delete Project

Environment Variables

Set project environment variables that can be accessed from your scripts.

Environment variable **values** are only visible to **collaborators** with **write** or higher access. They are a great way to securely store confidential information such as your AWS or database credentials. Names are available to all users with access to the project.

Name	Value	Actions
STORAGE	s3a://prod-cdptrialuser31-trycdp-com	<button>Delete</button>
		<button>Add</button>

Shared Memory Limit

Additional shared memory (in MB) that is available to sessions running within this project. If this field is blank, projects can only use 64MB of shared memory, which is the default for Docker containers.

Save Advanced Settings

trial31_admin / Telco_churn / Settings / SSH Tunnels

All Projects

Overview Sessions Experiments Models Jobs Applications Files Collaborators Project Settings

Project Settings

Options Runtime/Engine Advanced Tunnels Delete Project

SSH Tunnels

SSH tunnels allow you to easily connect to firewalled resources such as databases or Hadoop clusters. They will be created automatically every time you launch a console.

+ New Tunnel

Part 3: CML Sessions and Workbench

Sessions allow you to perform actions such as run R, Scala or Python code. They also provide access to an interactive command prompt and terminal. Sessions will be built on a specified Engine Image, which is a docker container that is deployed onto the Workspace. In addition you can specify how many resources are used per session.

From the Overview page click on New Session

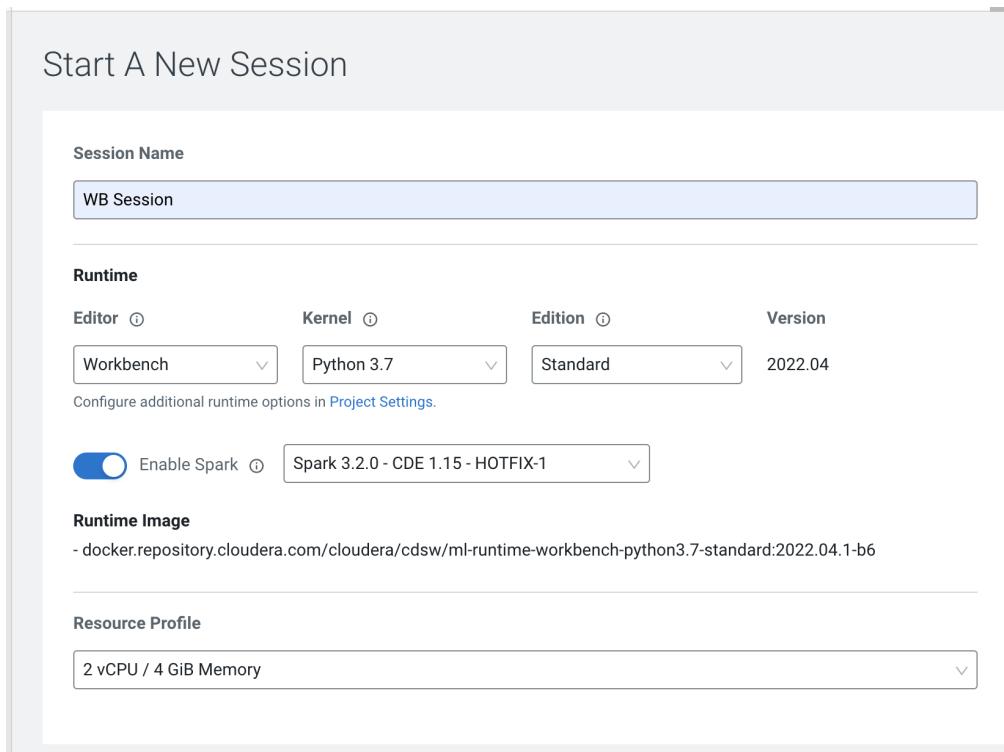
The screenshot shows the Cloudera Machine Learning (CML) interface. On the left, there's a sidebar with navigation links: Overview, Sessions (which is the current tab), Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main content area is titled "Telco_churn". Under "Models", it says "This project has no jobs yet. Create a new job to document your analytics pipelines." Below that is a "Files" section. The "Files" section contains a table with the following data:

Name	Size	Last Modified	Action
flask	2.22 kB	a few seconds ago	Edit
images	-	a few seconds ago	Edit
models	-	a few seconds ago	Edit
raw	-	a few seconds ago	Edit
0.bootstrap.py	7.41 kB	a few seconds ago	Edit
1.data_ingest.py	602.63 kB	a few seconds ago	Edit
2.data_exploration.ipynb	71.53 kB	a few seconds ago	Edit
3.model_building.ipynb	9.56 kB	a few seconds ago	Edit
4.train_models.py	8.10 kB	a few seconds ago	Edit
5.model_serve_explainer.py	7.74 kB	a few seconds ago	Edit
6.application.py	7.96 kB	a few seconds ago	Edit
7a.ml_ops_simulation.py	3.59 kB	a few seconds ago	Edit
7b.ml_ops_visual.py	8.72 kB	a few seconds ago	Edit
8.build_project.py	44 B	a few seconds ago	Edit
cdsw-build.sh	6.69 kB	a few seconds ago	Edit
churnexplainer.py	535 B	a few seconds ago	Edit
lineage.yml	828.00 kB	a few seconds ago	Edit
model.metrics.db	11.95 kB	a few seconds ago	Edit
README.md	175 B	a few seconds ago	Edit
requirements.txt	-	-	-

At the bottom right of the "Files" section, there are buttons for "Download", "New", and "Upload". The top right of the interface has a search bar, a user dropdown, and a "Fork" button. The top center shows the project name "trial31_admin / Telco_churn".

Session Name: telco_churn_session_1
Editor: Workbench
Kernel: Python 3.7
Resource Profile: 1vCPU/2 GiB Memory
Runtime Edition: Standard
Runtime Version: Any available version
Enable Spark Add On: enable any Spark version

Then select Start Session



The Workbench is now starting up and deploying a container onto the workspace at this point. Going from left to right you will see the project files, editor pane, and session pane.

Once you see the flashing red line on the bottom of the session pane turn steady green the container has been successfully started.

Script 1: Ingest Data

In this script you will ingest a raw csv file in a Spark Dataframe. The script has a .py extension and therefore is ideally suited for execution with the Workbench editor. No modifications to the code are required and it can be executed as is.

- You can execute the entire script in bulk by clicking on the “play icon” on the top menu bar. Once you do this you should notice the editor bar switches from green to red.

The screenshot shows a Jupyter Notebook interface. On the left, there is a code editor window titled "code/1_data_ingest.py". The code itself is a Python script with many comments explaining Cloudera's open source license and terms. Lines 52 through 55 are highlighted in red, indicating they are selected for execution. At the top of the code editor, there is a toolbar with "File", "Edit", "View", "Navigate", "Run", and a play button icon. Below the toolbar, there is a "Run all lines" button. To the right of the code editor, there is a "WB Session" panel. It shows the session is "Running" and was created by "Paul de Fusco". It also displays the logs, which include the Cloudera license text and the message "Access local data on your computer". The status bar at the bottom of the code editor shows "Line 124, Column 1", "276 Lines", "Python", and "Spaces 2".

```

1 # ##### CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
2 #
3 # CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
4 # (C) Cloudera, Inc. 2021
5 # All rights reserved.
6 #
7 # Applicable Open Source License: Apache 2.0
8 #
9 # NOTE: Cloudera open source products are modular software products
10 # made up of hundreds of individual components, each of which was
11 # individually copyrighted. Each Cloudera open source product is a
12 # collective work under U.S. Copyright Law. Your license to use the
13 # collective work is as provided in your written agreement with
14 # Cloudera. Used apart from the collective work, this file is
15 # licensed for your use pursuant to the open source license
16 # identified above.
17 #
18 # This code is provided to you pursuant to a written agreement with
19 # (i) Cloudera, Inc. or (ii) a third-party authorized to distribute
20 # this code. If you do not have a written agreement with Cloudera nor
21 # with an authorized and properly licensed third party, you do not
22 # have any rights to access nor to use this code.
23 #
24 # Absent a written agreement with Cloudera, Inc. ("Cloudera") to the
25 # contrary, A) CLOUDERA PROVIDES THIS CODE TO YOU WITHOUT WARRANTIES OF ANY
26 # KIND; (B) CLOUDERA DISCLAIMS ANY AND ALL EXPRESS AND IMPLIED
27 # WARRANTIES WITH RESPECT TO THIS CODE, INCLUDING BUT NOT LIMITED TO
28 # IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY AND
29 # FITNESS FOR A PARTICULAR PURPOSE; (C) CLOUDERA IS NOT LIABLE TO YOU,
30 # AND WILL NOT DEFEND, INDEMNIFY, NOR HOLD YOU HARMLESS FOR ANY CLAIMS
31 # ARISING FROM OR RELATED TO THE CODE; AND (D) WITH RESPECT TO YOUR EXERCISE
32 # OF ANY RIGHTS GRANTED TO YOU FOR THE CODE, CLOUDERA IS NOT LIABLE FOR ANY
33 # DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, PUNITIVE OR
34 # CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, DAMAGES
35 # RELATED TO LOST REVENUE, LOST PROFITS, LOSS OF INCOME, LOSS OF
36 # BUSINESS ADVANTAGE OR UNAVAILABILITY, OR LOSS OR CORRUPTION OF
37 # DATA.
38 #
39 # #####
40 #
41 # Part 1: Data Ingest
42 # A data scientist should never be blocked in getting data into their environment,
43 # so CML is able to ingest data from many sources.
44 # Whether you have data in .csv files, modern formats like parquet or feather,
45 # in cloud storage or a SQL database, CML will let you work with it in a data
46 # scientist-friendly environment.
47 #
48 # Access local data on your computer
49 #
50 # Accessing data stored on your computer is a matter of [uploading a file to the CML file sys
51 # referencing from there](https://docs.cloudera.com/machine-learning/cloud/import-data/topic
52

```

- As an alternative you can select subsets of the code and execute those only. This is great for troubleshooting and testing. To do so, highlight a number of lines of code from your script and then click on “Run” -> “Run Lines” from the top menu bar.

```
File Edit View Navigate Run ► code/1_data_ingest.py

97 #
98 # > First we specify
99 # > [environment vari
100 # > in your project s
101 # > Hive data. On AWS
102 # > on prem CDSW clus
103 #
104 # This was done for you when you ran `0_bootstrap.py`, so the following code is set up to ru
105 # It begins with imports and creating a 'SparkSession'.
106
107 import os
108 import sys
109 import subprocess
110
111 from cmlbootstrap import CMLBootstrap
112 from pyspark.sql import SparkSession
113 from pyspark.sql.utils import AnalysisException
114 from pyspark.sql.types import *
115
116 # Set the setup variables needed by CMLBootstrap
117 HOST = os.getenv("CDSW_API_URL").split(":")[0] + ":" + os.getenv("CDSW_DOMAIN")
118 USERNAME = os.getenv("CDSW_PROJECT_URL").split("/")[6] # args.username # "vdibia"
119 API_KEY = os.getenv("CDSW_API_KEY")
120 PROJECT_NAME = os.getenv("CDSW_PROJECT")
121
122 # Instantiate API Wrapper
123 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
124
```

The code is explained in the script comments. However, here are a key few highlights:

- Because CML is integrated with SDX and CDP, you can easily retrieve large datasets from Cloud Storage (ADLS, S3, Ozone) with a simple line of code
- Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.
- In Cloudera Machine Learning (CML), Spark and its dependencies are bundled directly into the CML engine Docker image.
- Furthermore, you can switch between different Spark versions at Session launch.

Notebooks 2 and 3: Interactive Analysis and Model Building with JupyterLab

In the previous section you loaded a csv file with a python script. In this section you will perform more Python commands with Jupyter Notebooks. Notebooks have a “.ipynb” extension and need to be executed with a Session using the JupyterLabs editor.

Launch a new session by selecting on the three “vertical dots” on the right side of the top menu bar.

The screenshot shows a Jupyter Notebook interface. At the top, there are tabs for 'Project', 'Terminal Access', 'Data', and a menu icon. Below the tabs, a session titled 'WB Session' is listed as 'Running'. The session was created by 'Paul de Fusco' and has a note: 'Session - 2 vCPU / 2 GB Memory'. There are three tabs: 'Session' (selected), 'Logs', and 'Spark UI'. A context menu is open over the 'WB Session' title, with the following options: 'Clear', 'Interrupt', 'Stop', 'WB Session - a few seconds ago, 2 vCPU, 2 GB Memory', and '+ New Session'. The '+ New Session' option is highlighted with a blue background. The main content area contains a note about Cloudera's open source license and a detailed disclaimer. The disclaimer states that Cloudera provides code without warranties, disclaims express and implied warranties, and is not liable for damages related to the code or its use.

Applicable Open Source License: Apache 2.0

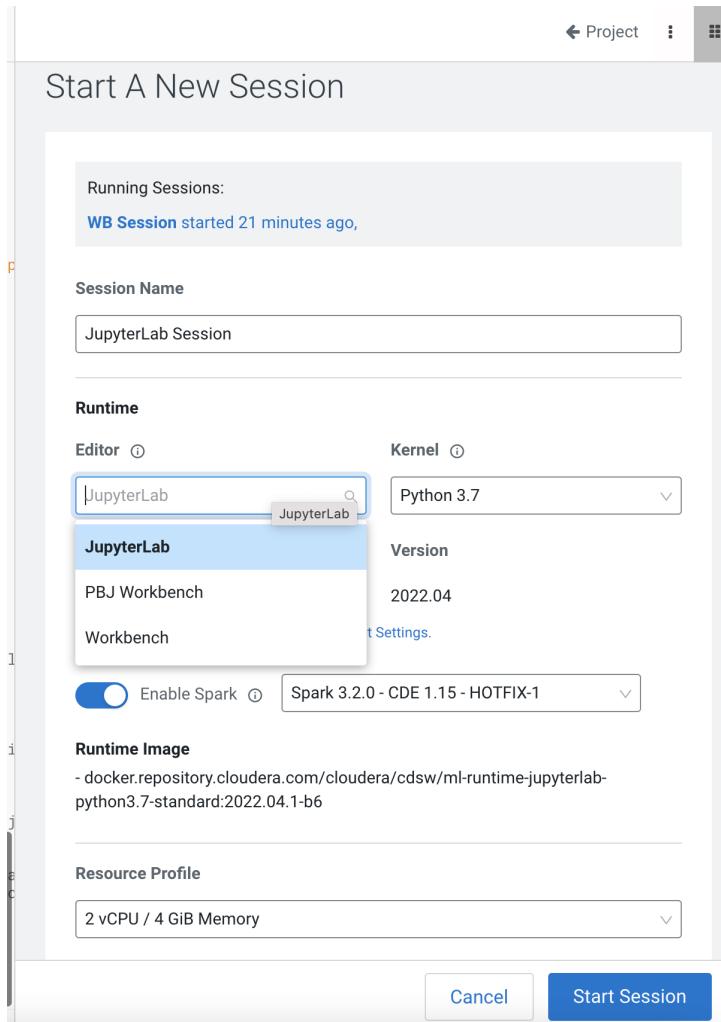
NOTE: Cloudera open source products are modular software products made up of hundreds of individual components, each of which was individually copyrighted. Each Cloudera open source product is a collective work under U.S. Copyright Law. Your license to use the collective work is as provided in your written agreement with Cloudera. Used apart from the collective work, this file is licensed for your use pursuant to the open source license identified above.

This code is provided to you pursuant a written agreement with *i* Cloudera, Inc. or *ii* a third-party authorized to distribute this code. If you do not have a written agreement with Cloudera nor with an authorized and properly licensed third party, you do not have any rights to access nor to use this code.

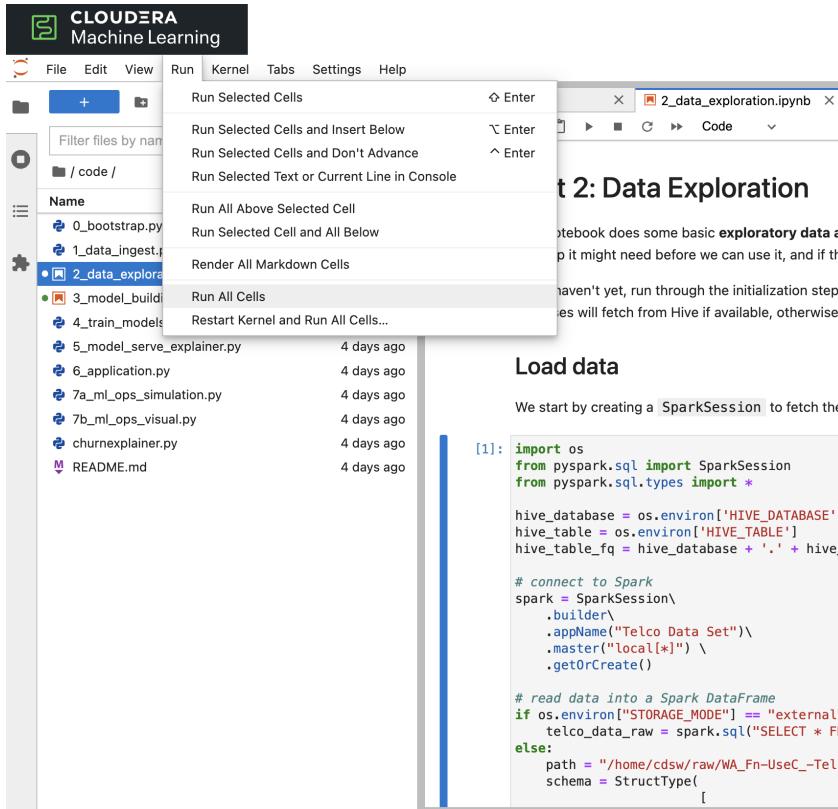
Absent a written agreement with Cloudera, Inc. “*Cloudera*” to the contrary, *A*) CLOUDERA PROVIDES THIS CODE TO YOU WITHOUT WARRANTIES OF ANY KIND; *B*) CLOUDERA DISCLAIMS ANY AND ALL EXPRESS AND IMPLIED WARRANTIES WITH RESPECT TO THIS CODE, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE; *C*) CLOUDERA IS NOT LIABLE TO YOU, AND WILL NOT DEFEND, INDEMNIFY, NOR HOLD YOU HARMLESS FOR ANY CLAIMS ARISING FROM OR RELATED TO THE CODE; AND *D*) WITH RESPECT TO YOUR EXERCISE OF ANY RIGHTS GRANTED TO YOU FOR THE CODE, CLOUDERA IS NOT LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, PUNITIVE OR CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, DAMAGES RELATED TO LOST REVENUE, LOST PROFITS, LOSS OF INCOME, LOSS OF BUSINESS ADVANTAGE OR UNAVAILABILITY, OR LOSS OR CORRUPTION OF DATA.

Launch the new Session with the following settings:

Session Name: telco_churn_session_2
Editor: JupyterLab
Kernel: Python 3.7
Resource Profile: 1vCPU/2 GiB Memory
Runtime Edition: Standard
Runtime Version: Any available version
Enable Spark Add On: enable any Spark version



After a few moments the JupyterLab editor should have taken over the screen. Open Notebook "2_data_exploration.ipynb" from the left side menu and investigate the code. Notebook cells are meant to be executed individually and give a more interactive flavor for coding and experimentation.



As before, no code changes are required and more detailed instructions are included in the comments. There are two ways to run each cell.

- Click on the cell you want to run. Hit “Shift” + “Enter” on your keyboard. Use this approach if you want to execute each cell individually.
- Open the “Run” menu from the top bar and then select “Run All”. Use this approach if you want to execute the entire notebook in bulk.

When you are finished with notebook “2_data_exploration.ipynb” go ahead and move on to notebook “3_model_building.ipynb”. As before, no code changes are required.

In summary, this section outlined the following concepts:

- With CML Runtimes, you can easily switch between different editors and work with multiple editors or programming environments in parallel if needed.
- First you stored a Spark Dataframe as a Spark table in the “1_ingest_data.py” python script using the Workbench editor.
- Then you retrieved the data in notebook “2_data_exploration.ipynb” using a JupyterLab session via Spark SQL. Spark SQL allows you to easily exchange files across sessions.
- Your Spark table was tracked as Hive External Tables and automatically made available in Atlas, the Data Catalog, and CDW. This is powered by SDX integration and requires no work on the CDP Admin or Users. We will see more on this in Part 7.

- Finally, in notebook “3_model_building.ipynb” you created a model with SciKit Learn and Lime, and then stored it in your project. Optionally, you could have saved it to Cloud Storage.
- CML allows you to work with any other libraries of your choice. This is the power of CML... any open source library and framework is one pip install away.

Part 4: CML Jobs and Experiments

Sessions allow you to perform actions such as run R or Python code and are ideal for interactive development. Experiments and Jobs also run code in isolated containers but provide additional capabilities. They are key tools to automation and streamlining of ML activities and a cornerstone of ML Ops.

- With Experiments you can streamline the execution of your batch script. This makes it the tool of choice for hyperparameter tuning or to simplify session management when you are iterating with many versions of the same model.
- With Jobs you can schedule and orchestrate the execution of your batch scripts. You can use cron expressions or build chains of dependencies with other jobs. This makes it the tool of choice for processing Data Pipelines as well as automating A/B tests and other ML Ops related tasks.

Script 4: Running scripts as an Experiment

From the CML Project home page navigate to the Experiments tab. Then click on the “New Experiment” icon on the top right.

The screenshot shows the Cloudera Machine Learning interface. On the left, a dark sidebar menu includes options like All Projects, Overview, Sessions, Data, Experiments (which is selected and highlighted in green), Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main content area displays the title "pauldefusco / Churn Modeling with scikit-learn - pauldefusco 1 / Experiments" and the heading "Experiments". Below this is a table listing two runs:

Run	Script	Arguments	Kernel	Comment	Submit
3	code/4_train_models.py	1	Python 3.8	pauldef	
2	code/4_train_models.py		Python 3.7	pauldef	

The screenshot shows the Cloudera Machine Learning interface with the path "1 / Experiments" at the top. It features a search bar, a user dropdown for "pauldefusco", and a button for "Run Experiment". Below is a table showing experiment details:

Comment	Submitter	Created At ▾	train_score	test_score	Status	Duration	Actions
	pauldefusco	6/17/22 7:53 AM			Build failed		
	pauldefusco	6/17/22 7:52 AM	0.81	0.79	Success	1 mins	

Similarly to Sessions, the form allows you to select runtime options to tailor resources to your needs. Select script “4_train_model.py” from the code folder and make sure to enable the Spark Addon. Any Spark version is fine.

Experiment Name: train_model_experiment

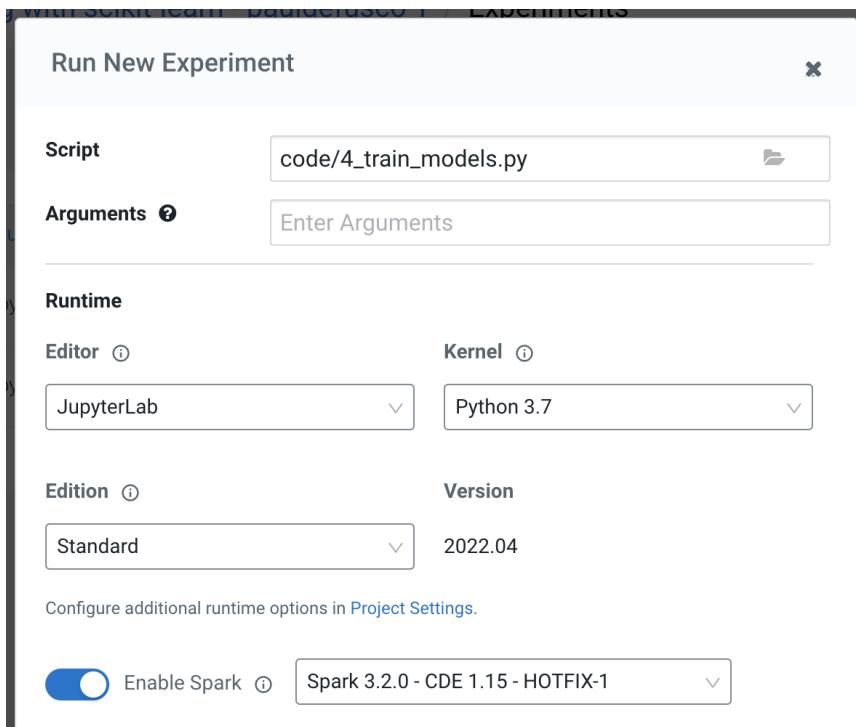
Editor: Workbench

Kernel: Python 3.7

Resource Profile: 1vCPU/2 GiB Memory

Runtime Edition: Standard

Enable Spark Add On: enable any Spark version



Launch the experiment and observe the status will change as CML automatically builds, deploys and executes the script in an isolated container.

When execution has completed, open the experiment by clicking on the ID value from the Experiments landing page. From the Overview tab inspect results. It looks like this Experiment yielded an Accuracy of 81% and 79% on the training and test sets respectively.

The screenshot shows the Cloudera Machine Learning interface. On the left, a sidebar menu lists various project components: All Projects, Overview, Sessions, Data, Experiments (which is selected and highlighted in green), Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main content area is titled "pauldefusco / Churn Modeling with scikit-learn - pauldef". It displays "Run-2" and a "New Experiment" button. Below this are tabs for Overview, Session, Logs, and Build, with "Overview" being the active tab. The "Configuration" section shows a "Script" set to "code/4_train_models.py", and other fields like Arguments, Comment, Build Snapshot, Created At (6/17/22 7:52 AM), and Submitter (pauldefusco). The "Metrics" section lists "train_score" at 0.8100 and "test_score" at 0.7900.

Script 4: Running scripts as a Job

Navigate to the CML Jobs page and select “New Job” on the top right.

The screenshot shows the Cloudera Machine Learning interface. The sidebar menu is identical to the previous screenshot, with "Jobs" selected. The main content area is titled "pauldefusco / Churn Modeling" and has a large "Jobs" heading. Below it, a message says "Job Dependencies for Samp". A form is present with a "Name" field containing "Dependent Job" and a "Sample Job" field below it.

Duration	Status	Latest Run	Actions
00:01	Failure	3 days ago	<button>Run</button>
00:32	Success	3 days ago	<button>Run</button>

Familiarize yourself with the form. Notice that you can set alerts and attachments in relation to each execution. Pick the following options and then create the job.

Create a Job

General

Name
Model Training Job

Script
code/4_train_models.py

Arguments
Arguments

Runtime

Editor ⚡ Kernel ⚡
Workbench Python 3.7

Edition ⚡ Version
Standard 2022.04

Configure additional runtime options in [Project Settings](#).

Enable Spark ⚡ Spark 2.4.8 - CDE 1.15 - HOTFIX-1

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2022.04.1-b6

Job Name: train_model_job
Script: code/4_train_models.py
Editor: Workbench

Kernel: Python 3.7

Resource Profile: 1vCPU/2 GiB Memory

Runtime Edition: Standard

Enable Spark Add On: any Spark version

Schedule: Manual

Jobs New Job

Job Dependencies for Sample Job

Sample Job  Dependent Job

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Model Training Job	0 / 0	00:00	Not Yet Run	-	<button>Run</button>
Dependent Job	1 / 1	00:01	Failure	3 days ago	<button>Run</button>
Sample Job	2 / 1	00:32	Success	3 days ago	<button>Run</button>

Notice the Jobs landing page has been populated with a new entry corresponding to your new job. Its status is “Not Yet Run”. Before executing the job, create another one with the following configurations. This time you will schedule it as “Dependent”.

Create a Job

General

Name
Model Simulation Job

Script
code/7a_ml_ops_simulation.py

Arguments
Arguments

Runtime

Editor  **Kernel** 

Workbench  Python 3.7 

Edition  **Version**
Standard  2022.04

Configure additional runtime options in [Project Settings](#).

Enable Spark  Spark 2.4.8 - CDE 1.15 - HOTFIX-1 

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2022.04.1-b6

Enable Spark ⓘ Spark 2.4.8 - CDE 1.15 - HOTFIX-1 ▾

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2022.04.1-b6

Schedule
Dependent ▾
Model Training Job ▾

Resource Profile
1 vCPU / 2 GiB Memory ▾

Timeout In Minutes (optional) Kill on Timeout
Jobs exceeding timeout send warning email if notifications enabled.

Environment Variables

Name	Value	Actions
<input type="text"/>	<input type="text"/>	<input type="button" value="Add"/>

Environment variables will override the [project environment](#).

Job Name: model_simulation_job

Script: code/7a_ml_ops_simulation.py

Editor: Workbench

Kernel: Python 3.7

Resource Profile: 1vCPU/2 GiB Memory

Runtime Edition: Standard

Enable Spark Add On: any Spark version

Schedule: Dependent -> Module Training Job

Navigate back to the Jobs landing page and manually trigger the “Module Training Job”. Observe it run for a minute or two. Notice that once it completes, the “Module Simulation Job” will launch automatically.

Jobs

New Job

Job Dependencies for Sample Job

Sample Job → Dependent Job

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Model Simulation Job	0 / 0	00:00	Not Yet Run	-	<button>Run</button>
Model Training Job	1 / 0	00:03	Running	a few seconds ago	<button>Stop</button>
Dependent Job	1 / 1	00:01	Failure	3 days ago	<button>Run</button>
Sample Job	2 / 1	00:32	Success	3 days ago	<button>Run</button>

Jobs

New Job

Job Dependencies for Sample Job

Sample Job → Dependent Job

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Model Simulation Job	1 / 0	00:06	Running	a few seconds ago	<button>Stop</button>
Model Training Job	1 / 0	00:41	Success	a few seconds ago	<button>Run</button>
Dependent Job	1 / 1	00:01	Failure	3 days ago	<button>Run</button>
Sample Job	2 / 1	00:32	Success	3 days ago	<button>Run</button>

Jobs

New Job

Job Dependencies for Sample Job

Sample Job → Dependent Job

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Model Simulation Job	2 / 1	03:28	Success	a few seconds ago	<button>Run</button>
Model Training Job	1 / 0	00:41	Success	5 minutes ago	<button>Run</button>
Dependent Job	1 / 1	00:01	Failure	3 days ago	<button>Run</button>
Sample Job	2 / 1	00:32	Success	3 days ago	<button>Run</button>

Optional: familiarize yourself with the code that just executed. Open the “Model Simulation Job”, then select the “History” tab and open the execution that succeeded.

The screenshot shows the “Model Simulation Job” interface. At the top right are “Success” and “Run” buttons. Below is a navigation bar with “Overview”, “History” (which is underlined), “Dependencies”, and “Settings”. The main area displays a table of executions:

Run	Status	Creator	Started	Endtime	Duration
Model Simulation Job	Success	Paul de Fusco	06/20/2022 3:48 PM	06/20/2022 3:52 PM	3m 28s
Model Simulation Job	Failure	Paul de Fusco	06/20/2022 3:44 PM	06/20/2022 3:45 PM	30s

You can use the contents in the “Session” tab to display exactly the code that was executed. This is particularly useful if you have a failure.

Scroll down and validate that the script sampled the data in our table and then “Added records” to a model. That model is actually the same classifier we just trained in the first job. It is hosted by an endpoint to serve requests over http. We will see more in the next section.

The screenshot shows the “Session” tab of the “Model Simulation Job” interface. At the top right is a “Success” button. Below is a summary: “By Paul de Fusco – Session – 1 vCPU / 2 GiB Memory – 7 minutes ago” with a “See job details” link. The “Logs” tab is selected. The log output shows the following Python code and its execution results:

```
# **note** this is an easy way to interact with a model in a script
response = cdsw.call_model(Model_AccessKey, no_churn_record)
response_labels_sample.append(
    {
        "uuid": response["response"]["uuid"],
        "final_label": churn_error(record["Churn"], percent_counter / percent_max),
        "response_label": response["response"]["prediction"]["probability"] >= 0.5,
        "timestamp_ms": int(round(time.time() * 1000)),
    }
)
Added 0 records
Added 50 records
Added 100 records
Added 150 records
Added 200 records
Added 250 records
Added 300 records
Added 350 records
Added 400 records
Added 450 records
Added 500 records
Added 550 records
Added 600 records
Added 650 records
Added 700 records
Added 750 records
Added 800 records
Added 850 records
Added 900 records
Added 950 records
```

The “ground truth” loop adds the updated actual label value and an accuracy measure every 100 calls to the model.

In summary, this section outlined the following concepts:

- With Experiments you can streamline your development process to more quickly iterate over model development trials. A typical use case is Hyperparameter tuning. The Experiments feature in CML allows you to track and visualize outcomes methodically.

- With Jobs you can schedule and orchestrate your batch scripts. Jobs allow you to build complex pipelines and are an essential part of any CI/CD or ML Ops pipeline. Typical use cases span from Spark ETL, Model Batch Scoring, A/B Testing and other model management related automations.
- To train a model is to run a batch script that uses a ML framework of choice to parameterize a function based on input data. Once the training routine is complete, the instance of the model is stored as a file. The model file is different from the model script. The model file can be reloaded from other scripts or notebooks for reuse.

Part 5: CML Models and Applications

Once a model is trained its predictions and insights must be put to use so they can add value to the organization. Generally this means using the model on new, unseen data in a production environment that offers key ML Ops capabilities.

One such example is Batch Scoring via CML Jobs. The model is loaded in a script and the predict function provided by the ML framework is applied to data in batch. The script is scheduled and orchestrated to perform the scoring on a regular basis. In case of failures, the script or data are manually updated so the scoring can resume.

This pattern is simple and reliable but has one pitfall. It requires the user or system waiting for the scoring job to run at its scheduled time. What if predictions are required on a short notice? Perhaps when a prospect navigates on an online shopping website or a potential anomaly is flagged by a third party business system?

- CML Models allow you to deploy the same model script and model file in a REST Endpoint so the model can now serve responses in real time. The endpoint is hosted by a container.
- CML Models provides tracking, metadata and versioning features that allow you to manage models in production.
- Similarly, CML Applications allows you to deploy visual tools in an endpoint container. This is typically used to host apps with open source libraries such as Flask, Shiny, Streamlit and more.
- Once a model is deployed to a CML Models container, a CML Application can forward requests to the Model endpoint to provide visual insights powered by ML models.

Inspecting a Model Endpoint

Navigate to the CML Models landing page. Notice that a model has already been created for you by the AMP setup automation.

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with various project management options like Overview, Sessions, Data, Experiments, Models (which is selected and highlighted in green), Jobs, Applications, Files, Collaborators, and Project Settings. The main area is titled 'Models' and shows a single entry: 'Churn Model API Endpoint' with a status of 'Deployed'. There are tabs for Model, Status, and Run.

Click on the Model name and explore the UI. The code for a sample request is provided on the left side.

This screenshot shows the 'Churn Model API Endpoint' details page. At the top, there's a navigation bar with tabs: Overview (selected), Deployments, Builds, Monitoring, Logs, and Settings. Below the tabs, there's a 'Description' field containing the text: 'This model API endpoint is used to predict churn'. Underneath it, there's a 'Sample Code' section with three buttons: Shell (selected), Python, and R. A code block shows a curl command to make a POST request to the API endpoint with JSON data. Below this, there's a 'Test Model' section with an 'Input' field containing a JSON object. At the bottom, there are 'Test' and 'Reset' buttons.

```
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-e849b6db-648.natx-aw.j9it-iyk2.cloudera.site/model -d '{"accessKey": "mnkmsdm8gz8ex1kwqunm12fgsi843zjv", "request": {"StreamingTV": "No", "MonthlyCharges": 70.35, "PhoneService": "No", "PaperlessBilling": "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"}}
```

```
{"OnlineSecurity": "No", "MultipleLines": "No", "InternetService": "DSL", "SeniorCitizen": "No", "TechSupport": "No"}
```

On the right side observe the model's metadata. Each model is assigned a number of attributes including Model Name, Deployment, Build and creation timestamp.

The screenshot shows a user interface for managing machine learning models. At the top, there are four buttons: 'Deployed' (highlighted in blue), 'Stop', 'Restart', and 'Deploy New Build'. Below this, there is a section titled 'Model Details' containing the following information:

Model Id	17
Model CRN	crn:cdp:ml:us-west-1:3b938e1b-8421-49e9-b073-2e74b44262c0:workspace:ad4e22f9-36f6-45a6-914d-13c75d75c667/6401a771-a2f9-49eb-bc24-83e53e37d005
Deployment Id	20
Deployment CRN	crn:cdp:ml:us-west-1:3b938e1b-8421-49e9-b073-2e74b44262c0:workspace:ad4e22f9-36f6-45a6-914d-13c75d75c667/71da05bb-14fd-43af-af59-aedd85fcc26f
Build Id	17
Build CRN	crn:cdp:ml:us-west-1:3b938e1b-8421-49e9-b073-2e74b44262c0:workspace:ad4e22f9-36f6-45a6-914d-13c75d75c667/ee54c335-ad4a-4a59-849c-edfffab4a026
Deployed By	pauldefusco
Comment	Build churn model
Runtime Image	Python 3.7 (Standard)
File	code/5_model_serve_explainer.py
Function	explain

Below the 'Model Details' section is another section titled 'Model Resources' with the following information:

Replicas	1
Total CPU	1 vCPUs

Scroll back to the left side and click on the “Test” icon. The window is pre-populated for you with a sample json dictionary. The dictionary is a set of key value pairs representing a customer’s attributes. For example, a customer who is currently on a DSL Internet Service plan.

The test simulates a request submission to the Model endpoint. The model processes the input and returns the output along with metadata and a prediction for the customer. In addition, the request is assigned a unique identifier. We will use this metadata for ML Ops later in part 6.

Test Model

Input

```
"OnlineSecurity": "No",
"MultipleLines": "No",
"InternetService": "DSL",
"SeniorCitizen": "No",
"TechSupport": "No"
}
```

[Test](#) [Reset](#)

Result

Status	● success
{ "model_deployment_crn": "crn:cdp:ml:us-west-1:3b938e1b-84: "prediction": { "data": { "Contract": "Month-to-month", "Dependents": "No", "DeviceProtection": "No", "InternetService": "DSL", "MonthlyCharges": 70.35, "MultipleLines": "No", "OnlineBackup": "No", "OnlineSecurity": "No", "PaperlessBilling": "No", "Partner": "No", "PaymentMethod": "Bank transfer (automatic)", "PhoneService": "No", } } }	

```

{
  "model_deployment_crn": "crn:cdp:ml:us-west-1:3b938e1b-8421-49e9-b07c",
  "prediction": {
    "data": {
      "Contract": "Month-to-month",
      "Dependents": "No",
      "DeviceProtection": "No",
      "InternetService": "DSL",
      "MonthlyCharges": 70.35,
      "MultipleLines": "No",
      "OnlineBackup": "No",
      "OnlineSecurity": "No",
      "PaperlessBilling": "No",
      "Partner": "No",
      "PaymentMethod": "Bank transfer (automatic)",
      "PhoneService": "No",
      "SeniorCitizen": "No",
      "StreamingMovies": "No",
      "StreamingTV": "No",
      "TechSupport": "No",
      "TotalCharges": 1397.475,
      "gender": "Female",
      "tenure": 29
    },
    "explanation": {
      "Contract": 0.11753693425620225,
      "InternetService": -0.15772997525799545,
      "MonthlyCharges": 0.0526064589033855,
      "MultipleLines": -0.02942972434722526,
      "OnlineSecurity": 0.05260308649410543,
      "PhoneService": -0.032876986659731695,
      "StreamingMovies": -0.03410779919944891,
      "StreamingTV": -0.045846594458528366,
      "TotalCharges": -0.07970724880651439,
      "tenure": 0.10511782642448833
    },
    "probability": 0.03353621326546117
  },
  "uuid": "fd4943c0-dc3a-41c8-97cd-d32e8e0550f9"
}

```

Script 5: Inspecting a Model Script

Navigate back to the Project Overview page and open the “5_model_serve_explainer.py” script. Scroll down and familiarize yourself with the code.

- Notice the method “explain” method. This is the Python function whose purpose is to receive the Json input as a request and return a Json output as a response.
- Within the method, the classifier object is used to apply the model object’s predict method.
- In addition, notice that a decorator named “@cdsw.model_metrics” is applied to the “explain” method. Thanks to the decorator you can use the “cdsw.track_metric” methods inside the “explain” method to register each scalar value associated with each request.
- The values are saved in the Model Metrics Store, a built in database used for tracking model requests.

Navigate back to the Project Overview page. Open the “models/telco_linear” subfolder and notice the presence of the “telco_linear.pkl” file. This is the physical model file loaded by the .py script you just inspected above.

Files / code / 5_model_serve_explainer.py

```
from collections import ChainMap
import cdsw, numpy
from churnexplainer import ExplainedModel

# Load the model saved earlier.
em = ExplainedModel.load(model_name="telco_linear")

# *Note:* If you want to test this in a session, comment out the line
# `@cdsw.model_metrics` below. Don't forget to uncomment when you
# deploy, or it won't write the metrics to the database

@cdsw.model_metrics
# This is the main function used for serving the model. It will take in the JSON formatted arguments , calculate the probability of
# churn and create a LIME explainer explained instance and return that as JSON.
def explain(args):
    data = dict(ChainMap(args, em.default_data))
    data = em.cast_dct(data)
    probability, explanation = em.explain_dct(data)

    # Track inputs
    cdsw.track_metric("input_data", data)

    # Track our prediction
    cdsw.track_metric("probability", probability)

    # Track explanation
    cdsw.track_metric("explanation", explanation)

    return {"data": dict(data), "probability": probability, "explanation": explanation}
```

Interacting with the Visual Application

Navigate to the CML Applications landing page. Notice that an application has already been created for you by the AMP setup automation.

The screenshot shows the Cloudera Machine Learning (CML) interface. On the left, there's a dark sidebar with various project management and data science tools. The 'Applications' option is currently selected and highlighted in green. The main right-hand panel is titled 'Applications' and contains a search bar at the top. Below the search bar, there's a single application card. The card has a small icon, the name 'Application to Serve Churn UI' followed by a link icon, and a green checkmark indicating it's 'Running since 5 minutes ago'. At the bottom of the card, there's a table with four columns: 'Project' (Churn Modeling with scikit-learn - pauldefusco), 'Created by' (pauldefusco), 'Last Updated' (06/20/2022 7:23 PM), and an empty column.

Click on the application in order to open it. This will automatically redirect you to the Visual Application landing page where the same data you worked with earlier is presented in an interactive table.

On the left side notice the probability column. This is the target variable predicted by the Machine Learning Model. It reflects the probability of each customer churning. The value is between 0 and 1. A value of 0.49 represents a 49% probability of the customer churning. By default, if the probability is higher than 50% the classifier will label the customer as “will churn” and otherwise as “will not churn”.

The 50% threshold can be increased or decreased implying customers previously assigned a “will churn” label may flip to “will not churn” and vice versa. This has important implications as it provides an avenue for tuning the level selectivity based on business considerations but a detailed explanation is beyond the scope of this content.

Next, click on the customer at the top of the table to investigate further.

Refractor

id	Probability	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	D
2672	0.584	Femal	No	No	No	4	Yes	No	Fiber	No	No	N
1540	0.123	Male	No	No	No	63	Yes	Yes	Fiber	No	Yes	Y
1724	0.061	Male	No	Yes	No	64	Yes	Yes	Fiber	Yes	Yes	Y
2085	0.060	Femal	No	Yes	Yes	48	No	No ph	DSL	No	No	N
2434	0.045	Male	No	Yes	Yes	17	Yes	No	No	No in	No in	N
5073	0.009	Male	No	Yes	Yes	53	Yes	No	DSL	Yes	Yes	N
1150	0.009	Femal	No	Yes	Yes	45	Yes	Yes	No	No in	No in	N
4151	0.008	Male	No	Yes	Yes	39	Yes	Yes	No	No in	No in	N
4249	0.003	Femal	No	Yes	No	70	Yes	Yes	No	No in	No in	N
1712	0.001	Male	No	Yes	No	72	Yes	No	No	No in	No in	N

A more detailed view of the customer is automatically loaded. The customer has a 58% chance of churning.

The Lime model applied to the classifier provides a color coding scheme highlighting the most impactful features in the prediction label being applied to this specific customer.

For example, this customer’s prediction of “will churn” is more significantly influenced by the “Internet Service” feature.

- The dark red color coding signals that the customer is negatively impacted by the current value for the feature.

- The current values of Monthly Charges and Phone Service also increase the likelihood of churn while the values of the Streaming Movies and Total Charges features decrease the likelihood of churn.

Single Prediction View

Churn Probability 0.585

Contract	Month-to-month	0.12	Month-to-month	One year	Two year	
Dependents	No	0.04	No	Yes		
DeviceProtection	No	0	No	No internet service	Yes	
InternetService	Fiber optic	0.20	DSL	Fiber optic	No	
MonthlyCharges	70.05	0.05	mean 64.80 min 18.25 max 118.75	<input type="text"/>	<input type="button" value="Submit"/>	
MultipleLines	No	0	No	No phone service	Yes	
OnlineBackup	No	0	No	No internet service	Yes	
OnlineSecurity	No	0.04	No	No internet service	Yes	
PaperlessBilling	Yes	0	No	Yes		
Partner	No	0	No	Yes		
PaymentMethod	Credit card (automatic)	0	Bank transfer (automatic)	Credit card (automatic)	Electronic check	Mailed check
PhoneService	Yes	0.04	No	Yes		
SeniorCitizen	No	0	No	Yes		
StreamingMovies	No	-0.04	No	No internet service	Yes	
StreamingTV	No	-0.04	No	No internet service	Yes	
TechSupport	No	0	No	No internet service	Yes	
TotalCharges	266.9	-0.12	mean 2283.30 min 18.80 max 8684.80	<input type="text"/>	<input type="button" value="Submit"/>	
gender	Female	0	Female	Male		

Let's see what happens if we change the value for the most impactful feature in this given scenario i.e. "Internet Service". Currently the value is set to "Fiber Optic". Hover over the entry and select "DSL".

Single Prediction View

Churn Probability	0.156
Contract	Month-to-month 0.11 Month-to-month One year Two year
Dependents	No 0.03 No Yes
DeviceProtection	No 0 No No internet service Yes
InternetService	DSL -0.17 DSL Fiber optic No
MonthlyCharges	70.05 0.05 mean 64.80 min 18.25 max 118.75 <input type="button"/> Submit
MultipleLines	No -0.04 No No phone service Yes
OnlineBackup	No 0 No No internet service Yes
OnlineSecurity	No 0 No No internet service Yes
PaperlessBilling	Yes 0 No Yes

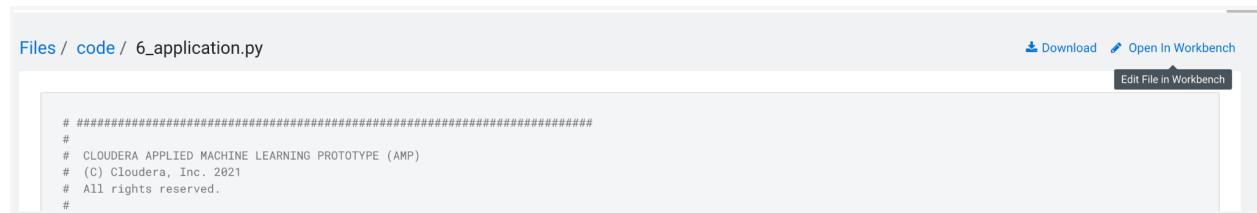
The table has now reloaded and the churn probability for this customer has dramatically decreased to roughly 15%.

This simple analysis can help the marketer optimize strategy in accordance to different business objectives. For example, the company could now tailor a proactive marketing offer based on this precious information. In addition, a more thorough financial analysis could be tied to the above simulation perhaps after adjusting the 50% threshold to increase or decrease selectivity based on business constraints or customer lifetime value assigned to each customer.

Script 6: Exploring the Application Script

Navigate back to the CML Project Home folder. Open the “Code” folder and then script “6_application.py”. This is a basic Flask application that serves the HTML and some specific data used for.

Click on “Open in Workbench” to visualize the code in a more reader friendly-mode.



```
# ######
# CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
# (C) Cloudera, Inc. 2021
# All rights reserved.
```

Now you will be able to explore the code with the Workbench Editor. The “Launch Session” form will automatically load on the right side of your screen. There is no need to launch a session so you can just minimize it.

As always no code changes are required. Here are some key highlights::

- At lines 177 - 191 we load the model and use the “Explain” method to load a small dataset in the file. This is similar to what you did in script 5. If you want to display more data or fast changing data there are other ways to do this, for example with Cloudera SQL Stream Builder.
- At line 248 we run the app on the "CDSW_APP_PORT". This value is already preset for you as this is a default environment variable. You can reuse this port for other applications.

Part 6: CML Models Operations

The following steps assume you have executed the model_simulation_job CML Job as shown in Part 4. If you haven't done it please go back and make sure to run the model simulation script.

Navigate back to the project overview and launch a new session with the following configurations.

Session Name: telco_churn_ops_session

Editor: Workbench

Kernel: Python 3.7

Resource Profile: 1vCPU/2 GiB Memory

Runtime Edition: Standard

Runtime Version: Any available version

Enable Spark Add On: any Spark version

Once the session is running, open script “7b_ml_ops_visual.py” and explore the code in the editor. You can execute the whole script end to end without modifications.

Observe the code outputs on the right side. Here are the key highlights:

- Model predictions are tracked in the CML Models Metrics Store. This is enabled by the use of the Python decorator and the use of “cdsw.track_metrics” methods in script 5. What is being tracked is completely up to the script developer.
- You can then extract the predictions and related metadata and put the information in a Pandas dataframe. Again, the Python library you use does not matter and is entirely up to the developer.
- This is exactly what the first diagram on the right side of your screen shows. Each column represents a prediction request reaching your CML Model endpoint. Each row represents a metric you are tracking in the CML Models Metrics Store.

WB Session Running

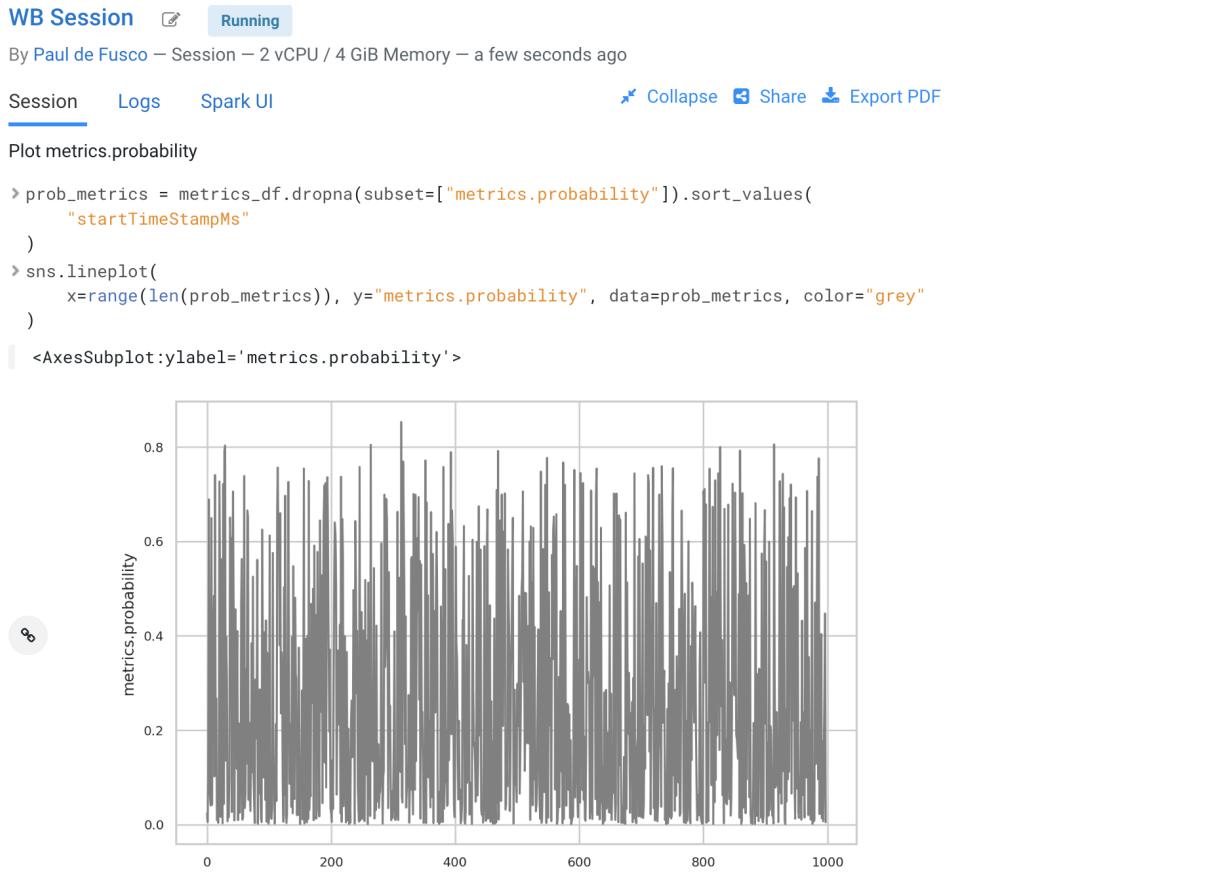
By Paul de Fusco – Session – 2 vCPU / 4 GiB Memory – a few seconds ago

Session Logs Spark UI ✖ Collapse Share Export PDF

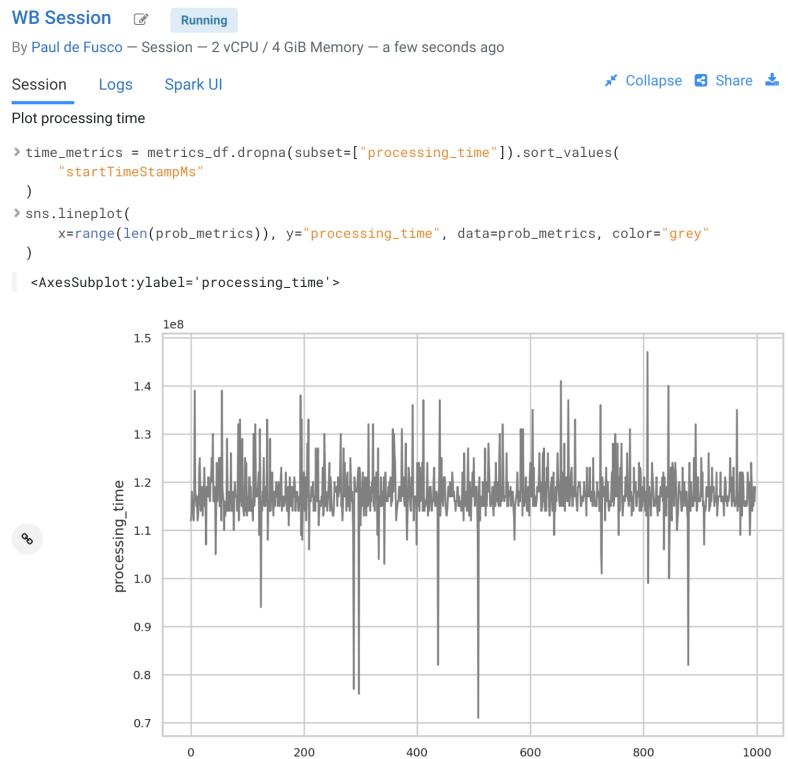
```
> metrics_df.tail().T
```

	1002	1003	1004	1005	1006
modelDeploymentCrn	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...
modelBuildCrn	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...
modelCrn	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...	crn:cdp:ml:us-west-1:8a1e15cd-04c2-48aa-8f35-b...
startTimestampMs	1656176961642	1656176928419	1656177003335	1656176938329	1656176990582
endTimestampMs	1656176961759	1656176928540	1656177003454	1656176938443	1656176990694
predictionUuid	ca709fe7-c203-4cd3-8929-0c1718e73951	b55f497f-312d-404c-82ce-808ecd68b779	19c044bf-39e9-4f3e-b5a0-caa17a951c66	fa8db4e6-8bd7-43e4-874d-56e50d404904	b11623dd-0885-4fd4-8278-8c16ba25f2c2
metrics.explanation.tenure	-0.274445	-0.281124	0.283526	0.120965	-0.278241
metrics.explanation.Contract	-0.119719	-0.12295	0.123687	0.110389	-0.117963
metrics.explanation.StreamingTV	0.0868918	-0.0653112	NaN	NaN	-0.0492012
metrics.explanation.TechSupport	NaN	-0.0701575	NaN	NaN	NaN
metrics.explanation.PhoneService	0.0468938	NaN	0.0405731	0.0477574	NaN
metrics.explanation.TotalCharges	0.196146	NaN	-0.0885498	NaN	0.189178
metrics.explanation.MultipleLines	0.0480728	0.0453388	0.0413393	0.0314279	0.0410217
metrics.explanation.MonthlyCharges	-0.137922	0.324689	-0.13177	-0.139677	-0.254845
metrics.explanation.InternetService	-0.167079	-0.0815516	0.207632	0.197279	0.188008
metrics.explanation.StreamingMovies	-0.0489575	NaN	-0.0508605	-0.0494431	0.0871053
metrics final label	False	False	True	True	False

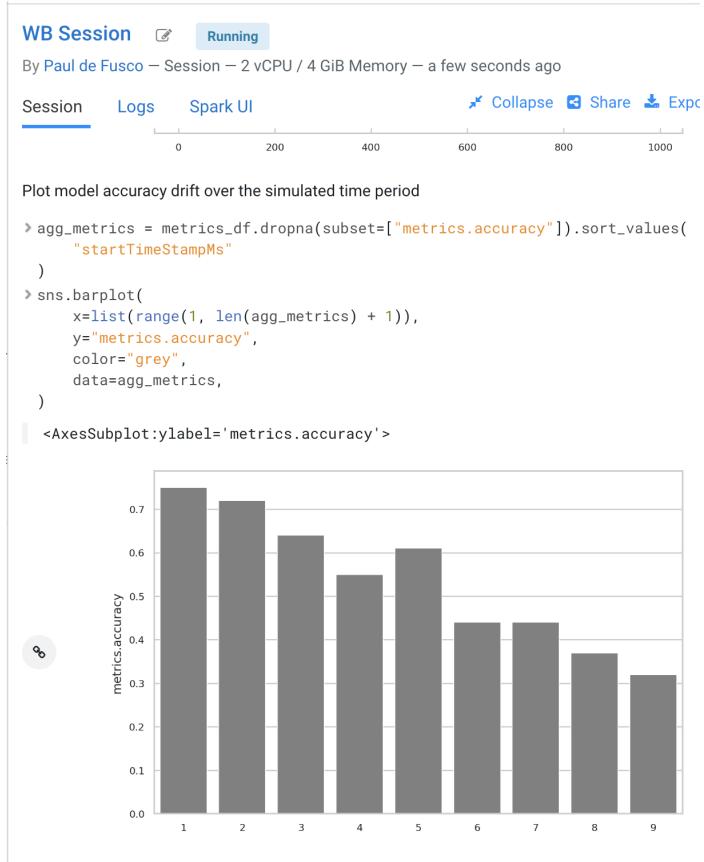
- Once the tracked metrics have been saved to a Python data structure they can be used for all sorts of purposes.
- For example, the second diagram shows a basic line plot in Seaborn where the models' output probabilities are plotted as a function of time. On the X axis you can see the timestamp associated with each request. On the Y axis you can find the associated output probability.



- Similarly, you can plot processing time as shown in the third diagram. This represents the time duration required to process a particular request.
- As an example, this information could be used to trigger the deployment of more resources to support this model endpoint when a particular threshold is passed. You can deploy more resources manually via the UI, or programmatically and in an automated CI/CD pipeline with CML APIv2 and CML Jobs.



- You can also monitor the model's accuracy over time. For example, the below diagram shows a line plot of prediction accuracy sorted over time. As you can see, the trend is negative and the model is making increasingly less accurate predictions.
- Just like with processing time and other metrics, CML allows you to implement ML Ops pipelines that automate actions related to model management. For example, you could use a combination of CML Jobs and CML APIv2 to trigger the retraining and redeployment of a model when its accuracy reaches a particular threshold over a particular time period.
- As always this is a relatively basic example. CML is an open platform for hands-on developers which gives users the freedom to implement more complex ML Ops pipelines.



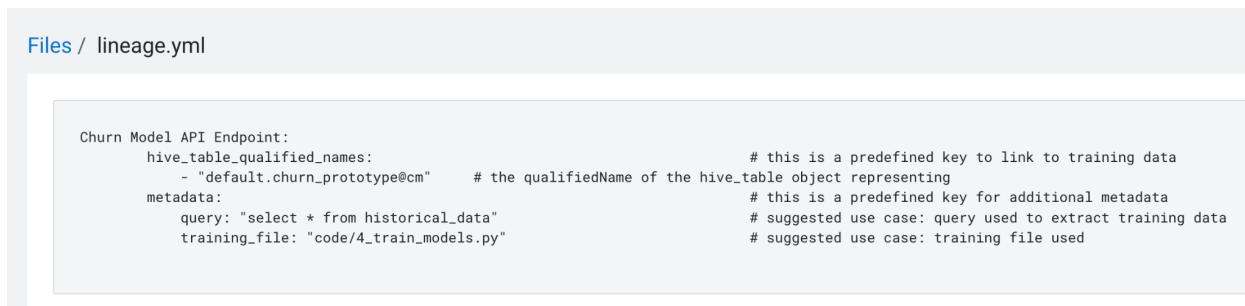
- Ground truth metrics can be collected with the `cdsw.track_delayed_metrics` method. This allows you to compare your predictions with the actual event after the prediction was output. In turn, this allows you to calculate the model accuracy and create visualizations such as the one above.
- For an example of the `cdsw.track_delayed_metrics` method open the “`7a_ml_ops_simulation.py`” script and review lines 249 - 269. Keep in mind that this is just a simulation.
- In a real world scenario the requests would be coming from an external system or be logged in a SQL or NoSQL database. In turn, the script above would be used to set ground truth values in batch via a CML Job or in real time with a CML Model endpoint.

Part 7: Model Lineage Tracking

CDP is an end-to-end hybrid enterprise data platform. Every user, workload, and dataset and machine learning model can be governed from a central location via SDX, the Shared Data Experience.

Under the hood, SDX tracks and secures activity related to each CDP Data Service via “Hooks” and “Plugins”, including CML. If you want your models to be logged in SDX you have to add them to the `lineage.yml` file located in your project home folder.

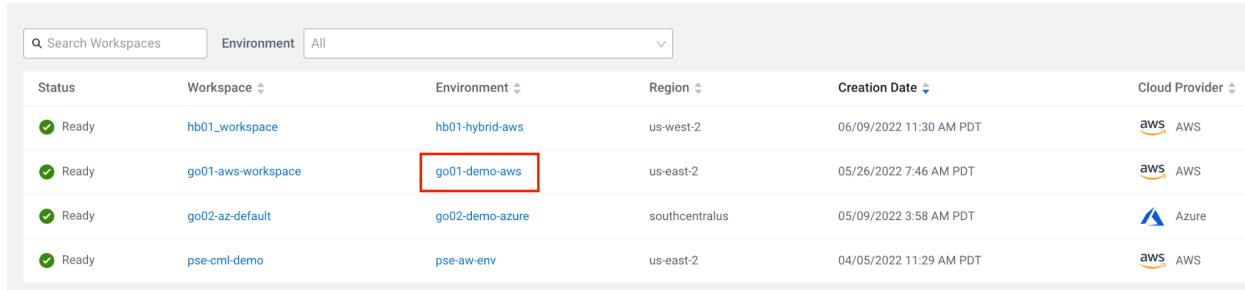
Navigate back to the Project Home folder and open the lineage.yml file located in the files section.



```
Churn Model API Endpoint:
hive_table_qualified_names:
  - "default.churn_prototype@cm"      # the qualifiedName of the hive_table object representing
metadata:
  query: "select * from historical_data"          # this is a predefined key to link to training data
  training_file: "code/4_train_models.py"           # suggested use case: query used to extract training data
                                                # suggested use case: training file used
```

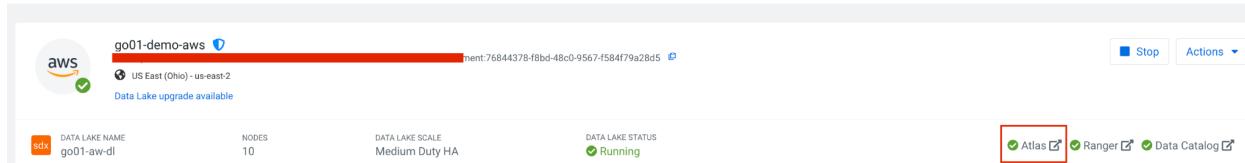
Navigate out of your CML project and go back to the Workspaces landing page. Click on the Data Lake associated with your CML Workspace and then open the Apache Atlas Service.

Machine Learning Workspaces



Status	Workspace	Environment	Region	Creation Date	Cloud Provider
Ready	hb01_workspace	hb01-hybrid-aws	us-west-2	06/09/2022 11:30 AM PDT	aws AWS
Ready	go01-aws-workspace	go01-demo-aws	us-east-2	05/26/2022 7:46 AM PDT	aws AWS
Ready	go02-az-default	go02-demo-azure	southcentralus	05/09/2022 3:58 AM PDT	Azure Azure
Ready	pse-cml-demo	pse-aw-env	us-east-2	04/05/2022 11:29 AM PDT	aws AWS

Environments / go01-demo-aws



From the Atlas UI, search for ML models by entering the “ml_model_build” type. Notice that there are various Atlas entities to browse for models.

The screenshot shows the Apache Atlas search interface. At the top, there are three navigation tabs: 'SEARCH', 'CLASSIFICATION', and 'GLOSSARY'. Below them are two search modes: 'Basic' (selected) and 'Advanced'. A refresh icon is also present. The main area is titled 'Search By Type' and features a dropdown menu. The dropdown is set to '_ALL_ENTITY_TYPES' and contains a list of entity types. The 'model' type is selected and highlighted in blue. Other listed types include 'ml_model_build (32)', 'ml_model_deploy_process (36)', 'ml_model_deployment (36)', 'ml_model_train_build_process (32)', and 'spark_ml_model'. At the bottom of the dropdown are 'Clear' and 'Search' buttons.

In the output, select the model you created. Notice that each model is assigned a unique ID at the end. That ID corresponds to the Model Build from CML. Open your model by selecting the Model Name.

The screenshot shows the Apache Atlas search results for the entity type 'ml_model_build'. The search bar at the top has 'ml_model_build (32)' entered. The results table has four columns: 'Name', 'Owner', 'Description', and 'Type'. There are three entries listed:

Name	Owner	Description	Type
Churn Model API Endpoint-13	pauldefusco		ml_model_build
Churn Model API Endpoint-17	efernandes		ml_model_build
Churn Model API Endpoint-21	agillan		ml_model_build

Familiarize yourself with the Model properties tab. Notice that each model logged is associated with rich metadata. You can customize Atlas Model metadata by editing the `lineage.yml` file in the CML Project Home folder



Churn Model API Endpoint-13 (ml_model_build)

Classifications: [+](#)

Terms: [+](#)

Properties

Lineage

Relationships

Classifications

Audits

Tasks

Technical properties

createTime	06/09/2022 06:05:10 PM (PDT)
defaultCpuMillicores	1000
defaultGpus	0
defaultMemoryMb	2048
exampleRequest	{"StreamingTV": "No", "MonthlyCharges": 70.35, "PhoneService": "No", "PaperlessBilling": "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"}
exampleResponse	" "
imageHash	Not Available
imageTag	172.20.200.38:5000/bd6a7c60-ea6a-426b-9f63-67548e599597
metadata	{ parent_name: "Churn Modeling with scikit-learn - pauldefusco". }

User-defined properties

Labels

Business Metadata



Churn Model API Endpoint-13 (ml_model_build)

Classifications: [+](#)

Terms: [+](#)

Properties

Lineage

Relationships

Classifications

Audits

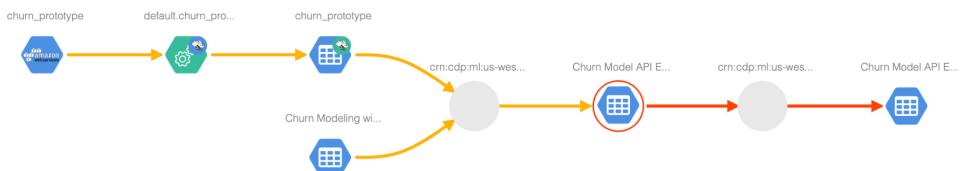
Tasks

Current Entity

In Progress

Lineage

Impact



Atlas and Ranger provides a rich set of Governance and Security capabilities. For example, you can apply Atlas tags to your entities across Data Services and then propagate Ranger policies to automatically secure applications across complex pipelines.

A detailed exploration of SDX in the context of CML is not in scope for this workshop but please visit the “Next Steps” section to find out more on this and other topics.

Conclusions

In this workshop you created an end to end project to support a Machine Learning model in Production.

- You easily created a Spark Session and explored a large dataset with the PySpark library. Thanks to CML Runtimes and Sessions you were able to switch between editors, resources, and optionally Python and Spark versions at the click of a button.
- You created a Model REST Endpoint to serve predictions to internal or external business applications. Then, you built an interactive dashboard to make the “black box model” interpretable for your business stakeholders.
- You explored the foundations of a basic ML Ops pipeline to easily retrain, monitor, and reproduce your model in production. With the CML Models interface you unit tested and increased model observability. Then, you monitored its performance with CML APIv2.
- Finally, you used CDP SDX to log and visualize Model Metadata and Lineage.

Next Steps

If you want to learn more about CML and CDP we invite you to visit the following assets and tutorials or ask your Cloudera Workshop Lead for a follow up.

- [Learn how to use Cloudera Applied ML Prototypes](#) to discover more CML Projects using MLFlow, Streamlit, Tensorflow, PyTorch and other popular libraries. The AMP Catalog is maintained by the Cloudera Fast Forward Labs team and allows you to automatically deploy complex use cases within minutes.
- [CML HowTo](#): A series of tips and tricks for the CML beginner
- [Sentiment Analysis in R](#): an end to end ML project with SparklyR and GPU training
- [CSA2CML](#): Build a real time anomaly detection dashboard with Flink, CML, and Streamlit
- [SDX2CML](#): Explore ML Governance and Security features in more detail to increase legal compliance and enhance ML Ops best practices.
- [CML2CDE](#): Create CI/CD Pipelines for Spark ETL with CML Notebooks and CDE Virtual Cluster
- [API v2](#): Familiarize yourself with API v2, CML's goto Python Library for ML Ops and DevOps
- [Distributed PyTorch with Horovod](#): A quickstart for distributing Horovod with the CML Workers API