

# CML Workshop Student Guide

---

<b>Introduction</b>	2
<b>Labs Summary</b>	2
<b>Lab 1 - Initial setup</b>	4
Getting Connected	4
Setting Up Workload Password	4
Create CML Project	6
Workload Password	6
Create a New Project	7
Project Environment Variables	10
<b>Lab 2 - Data Ingest</b>	11
Run the Bootstrap Script	11
Execute Data Ingest Script	14
<b>Lab 3 - Data Exploration</b>	17
Execute the data exploration script	17
<b>Lab 4 - Model Training and Experiment</b>	22
Build the model	22
Model Training and Experiments	22
Compare Experiments	24
<b>Lab 5 - Model Deploy/Serve</b>	25
Build a Model	25
Deploy a Model	28
Test a Model	28
<b>Lab 6 - Application Deployment</b>	30
Prerequisites	30
Application Deployment	31

Error! Bookmark not defined.

# Introduction

## 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.

**FF06**

**REPORT**

**Interpretability**

Interpretability, or the ability to explain why and how a system makes a decision, can help us improve models, satisfy regulations, and build better products. Black-box techniques, such as deep learning, have delivered breakthrough capabilities at the cost of interpretability. In this report we show how to make models interpretable without sacrificing their capabilities or accuracy.

**PROTOTYPES**

**Refractor**

The Refractor prototype allows users to explore and interact with a machine learning model for predicting customer churn. It features a data table with columns like Id, Churn Prob., Phone Service, Internet Service, Multi-line, Stream Movies, Stream TV, Online Secur., Online Backup, Tech Support, Contract, Paym..., Paperless, Month-By..., Total Charge, Tenure, and Partner. A modal window titled "Introducing Refractor" provides an overview of the tool's purpose and how it relates to the report's content.

The Hands on Labs will take you through the whole process of logging in, sourcing the code and building fully working applications with deployed models. This lab builds the telco churn with model interpretability projects discussed in more detail in this blog post.

## Labs Summary

In this exercise we will implement an end-to-end machine learning workflow using Cloudera

Machine Learning, including:

- Data ingest
- Data exploration
- Model training and experimentation
- Model serving
- Business applications

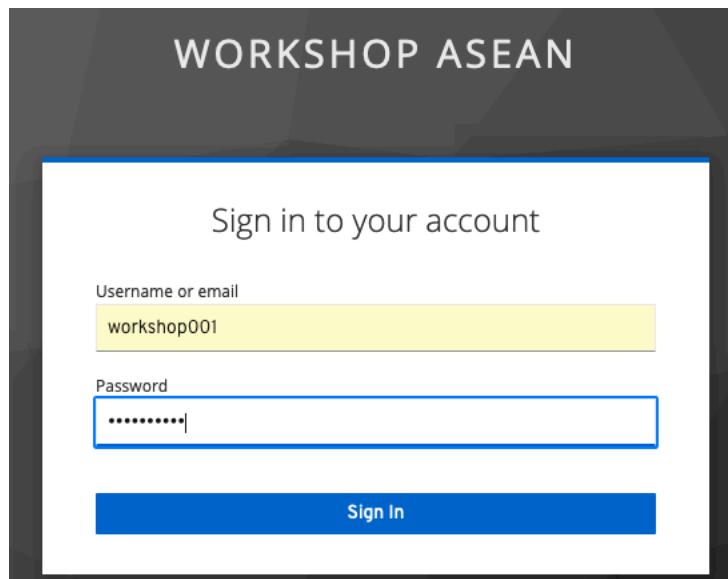
## Lab 1 - Initial setup

### Getting Connected

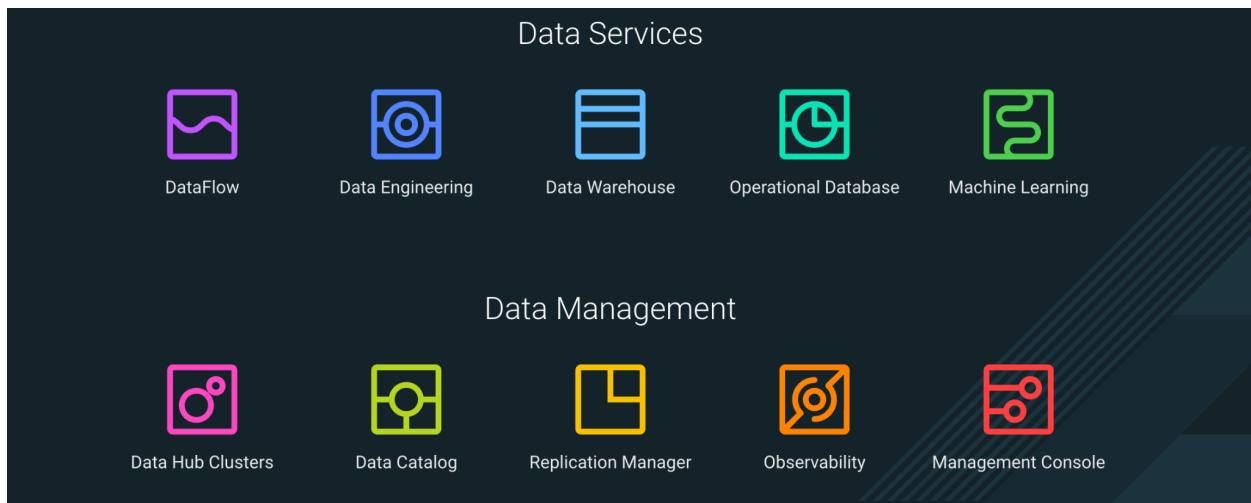
CDP Console URL	<a href="https://tinyurl.com/mascmlworkshop">https://tinyurl.com/mascmlworkshop</a>
User	Refer Google sheet (e.g. workshop0xx )
Password	Refer Google sheet
Project Github URL	<a href="https://github.com/ogakulov/CML_AMP_Churn_Prediction_mlflow">https://github.com/ogakulov/CML_AMP_Churn_Prediction_mlflow</a>

### Setting Up Workload Password (No to be performed)

- We have already take care of the below prerequisites:
  - Enable CML workspace ([mas-cml-workshop](#))
- Open the shared [link](#) and login with the credentials assigned to you.



- You should land on the CDP Console as shown below.



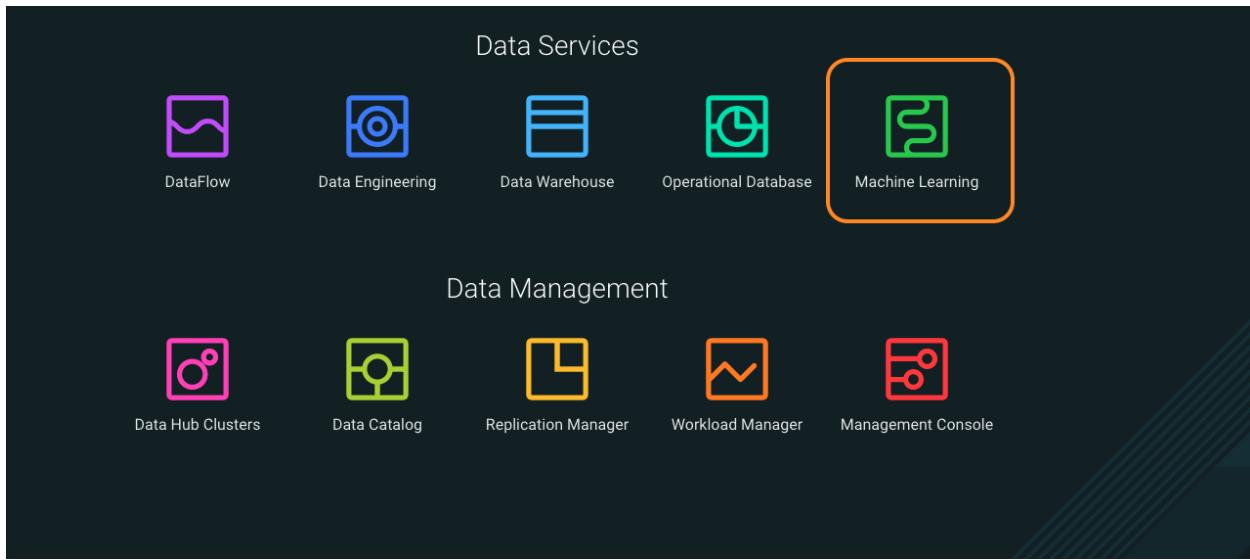
- Set the CDP Work Profile Password:
  - Click on the User Profile from the below screenshot

The screenshot shows the Cloud Era Data Platform login page at [console.altus.cloudera.com/](https://console.altus.cloudera.com/). The user profile dropdown menu is open, showing the email address `fubon20@keycloak.cloudera.com` and a red box highlighting the 'Profile' option. Other options in the menu include 'Log Out'. The main page features sections for 'Data Services' (DataFlow, Data Engineering, Data Warehouse, Operational Database, Machine Learning) and 'Data Management' (Data Hub Clusters, Data Catalog, Replication Manager, Observability, Management Console). A banner at the top right encourages users to start a 60-day free trial of CDP Public Cloud.

- Use the Workload User name from the below screenshot
  - Ex: `workshop001` is the CDP Workload username
- Click on the Set Workload Password to set the CDP Workload password
  - Ex: `P@ssw0rd@2023`

## Create CML Project

Login into CDP using the URL above and the credentials assigned to you. After logging into CDP you will have access to the main console and then go to Cloudera Machine Learning. Click on Workspace to proceed to executing the hands-on labs.



This will launch a ML workspace screen. (Workspace=mas-cml-workshop)

The screenshot shows the 'Machine Learning Workspaces' page. On the left, there's a sidebar with 'Workspaces' and 'Workspace Backups'. The main area has a table titled 'Machine Learning Workspaces' with columns: Status, Version, Workspace, Environment, Region, Creation Date, Cloud Provider, and Actions. There are four rows in the table:

Status	Version	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	2.0.38	mas-cml-workshop	workshop-asean	ap-southeast-1	05/22/2023 9:34 PM +08	AWS AWS	⋮
Ready	2.0.38	lubon-cml-workshop	workshop-asean	ap-southeast-1	05/15/2023 7:42 AM +08	AWS AWS	⋮
Suspended	2.0.38	amer-cml-ws	amer-workshop-env	us-east-2	05/08/2023 9:44 PM +08	AWS AWS	⋮
Ready	2.0.38	meta-cml-workshop	meta-workshop	eu-west-2	03/29/2023 2:06 PM +08	AWS AWS	⋮

At the bottom, it says 'Displaying 1 - 4 of 4 < 1 > 25 / page ▾'.

## Workload Password

Provide the same password as we set in the previous step.

User Settings

Profile   Outbound SSH   API Keys   Remote Editing   Environment Variables

**Reserved Environment Variables**

Reserved environment variables are required for specific features of CML. The variables specified here override the environment variables specified below.

WORKLOAD_PASSWORD	WORKLOAD_PASSWORD	Save
-------------------	-------------------	------

**Environment Variables**

Set the user's environment variables that can be accessed from your scripts.

These environment variables are only visible to you in the sessions, applications, jobs, experiments and models you launched and will not be visible to any other user.

Name	Value
[empty]	[empty]

## Create a New Project

Projects

> View Resource Usage Details ✓

Search Projects Scope My Projects Creator All

You currently don't have any projects

Projects are the heart of Cloudera Machine Learning. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

New Project

Create a new Project based on Github.

**Project Name:** <username>\_telco\_churn\_projv1

**Initial Setup:** Git

**Git URL:** [https://github.com/cloudera/CML\\_AMP\\_Churn\\_Prediction](https://github.com/cloudera/CML_AMP_Churn_Prediction)

\* Project Name

Project Description

Project Visibility  
 Private - Only added collaborators can view the project  
 Public - All authenticated users can view this project.

Initial Setup

Provide the Git URL of the project to clone. Select the option that applies to your URL access.  
 HTTPS  SSH

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

Runtime setup

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

Add GPU enabled Runtime variant

These runtimes will be added to the project:  
JupyterLab - Python 3.7 - Standard - 2022.11  
PBJ Workbench - Python 3.7 - Standard - 2022.11  
Workbench - Python 3.7 - Standard - 2022.11

The project is laid out in a familiar way with markdown and a file view that is familiar to developers.

Next select the project which was created just now and this will take you to below screen

Name	Size	Last Modified
churn_prediction.egg-info	-	a minute ago
code	-	a minute ago
flask	-	a minute ago
images	-	a minute ago
logs	-	a minute ago
models	-	a minute ago
raw	-	a minute ago
src	-	a minute ago
cdsw-build.sh	45 B	a minute ago
cm_catalog.yaml	1.01 kB	a minute ago
LICENSE.txt	9.94 kB	a minute ago
lineage.yaml	745 B	a minute ago
model_metrics.db	828.00 kB	a minute ago

## Project Environment Variables

The screenshot shows the 'Project Settings' page for a project named 'fubon20'. The left sidebar has a 'Project Settings' link with a red arrow pointing to it. The main content area shows the 'Environment Variables' section with the following table:

Variable Name	Value	-	+
CDSW_APP_POLLING_ENDPOINT	/	-	+
PROJECT_OWNER	fubon20	-	+
DATA_LOCATION	data/churn_prototype	-	+
HIVE_DATABASE	default	-	+
HIVE_TABLE	churn_prototype_fubon20	-	+
STORAGE_MODE	external	-	+

A red box highlights the rows for DATA\_LOCATION, HIVE\_DATABASE, HIVE\_TABLE, and STORAGE\_MODE.

Add below variable in Project Settings

Variable Name	Value
DATA_LOCATION	data/churn_prototype
HIVE_DATABASE	default
HIVE_TABLE	churn_prototype_<username>
STORAGE_MODE	external

## Lab 2 - Data Ingest

In this and the following lab, you will work on the Data Ingest Stage.

### Run the Bootstrap Script

Start a “**NEW SESSION**” and use the below configuration.

**Name:** workshop0xx\_session1

**Editor:** Workbench

**Engine Kernel:** Python 3

**Resource Profile:** 2 vCPU / 4 GiB Memory

**Enable Spark :** Yes

A screenshot of a Jupyter Notebook interface. At the top, the project name "fubon20\_telco\_churn\_projV1" is displayed. To the right, there are buttons for "0", "Fork", and a prominent blue "New Session" button, which has an orange arrow pointing to it. Below the title, the text "fubon20\_telco\_churn\_projV1" is shown. The interface is divided into sections: "Models" (with the note "This project has no models yet. Create a new model."), "Jobs" (with the note "This project has no jobs yet. Create a new job to document your analytics pipelines."), and "Files". The "Files" section shows a list of directories: "churn\_prediction.egg-info", "code", "flask", "images", "logs", "models", "raw", and "src". Each item in the list has a small checkbox to its left, and columns for "Size" and "Last Modified" showing "- 5 minutes ago". At the bottom right of the "Files" section, there are buttons for "Download", "New", and "Upload".

## Start A New Session

**Session Name**  
fubon20\_session1

**Runtime**

Editor	Kernel	Edition	Version
Editor ⓘ	Kernel ⓘ	Edition ⓘ	Version
Workbench	Python 3.7	Standard	2022.11

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

Enable Spark ⓘ Spark 2.4.8 - CDE 1.17 - HOTFIX-1

**Runtime Image**  
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2022.11.2-b2

**Resource Profile**  
2 vCPU / 4 GiB Memory

Click on **Start Session**

Close the Code Snippet

Connection Code Snippet X

You have access to the Data Connections below.  
This also be accessed by clicking the **Data** tab in the top menu.

 fubon-workshop-datalake
TYPE Spark Data Lake

Use this code to connect to the chosen data source.

```
import cmldata_v1 as cmldata
CONNECTION_NAME = "fubon-workshop-datalake"
conn = cmldata.get_connection(CONNECTION_NAME)
spark = conn.get_spark_session()

# Sample usage to run query through spark
EXAMPLE_SQL_QUERY = "show databases"
spark.sql(EXAMPLE_SQL_QUERY).show()
```

Don't show me this again for:  This Project  All Projects Close

- Select the **code/0\_bootstrap.py** on the left file browser

The screenshot shows the Cloudera Manager interface. On the left, there is a file browser window titled 'muser00\_telco\_churn\_projV1'. It lists several files and directories, with '0\_bootstrap.py' highlighted. A red arrow points to this file. On the right, there is a code editor window titled 'code/0\_bootstrap.py'. The code in the editor is a Python script for a CML project. At the top of the code, there is a license notice from Cloudera. The code itself includes comments about installing requirements and setting up storage.

```

# #####################################################################
# # CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
# # Copyright © 2021 Cloudera, Inc. 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
# # developed by a different person and company. As such, the terms of
# # each open source license still apply to the individual modules.
# # See the component structure at the root of this project for
# # an outline of each module and each individual's contribution.
# #
# # This code is provided to you pursuant to 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 install, use, modify, or redistribute 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 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR
# # FITNESS FOR ANY 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 REGARD TO YOUR EXERCISE
# # OF ANY RIGHTS IN 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.
# #
# ## Part 0: Bootstrap File
# # You need to run this at the start of the project. It will install the requirements,
# # set the STORAGE environment variables and copy the data from
# # raw/NA_Fn-UseC-Telco-Customer-Churn.csv into specified path of the STORAGE
# # location, if applicable.
# #
# # The STORAGE environment variable is the Cloud Storage location used by the DataLake
# # to store hive data. On AWS it will be s3://[something], on Azure it will be
# # abfs://[something] and on a CDSW cluster, it will be hdfs://[something]
# #
# # Install the requirements
# !pip3 install -r requirements.txt
# 
```

- Run **code/0\_bootstrap.py** to download the dependencies.

The screenshot shows the Cloudera Manager interface. On the left, there is a file browser window titled 'muser00\_telco\_churn\_projV1'. It lists several files and directories. On the right, there is a code editor window titled '0\_bootstrap.py'. The code in the editor is the same as in the previous screenshot. At the top of the code editor, there is a 'Run' menu with several options: 'Run Line(s)', 'Run All', 'Run Experiment...', and 'Run All'. A red arrow points to the 'Run All' option. To the right of the code editor, there is a sidebar with various sections like 'mus', 'By An', 'Sessi', 'Getti', 'This i', 'To exi', 'also e', and 'Use'.

- Dependencies are starting to get downloaded.

The screenshot shows a terminal window with two panes. The left pane displays Python code for a project setup, including imports for `os`, `time`, `datetime`, `requests`, and `xml.etree.ElementTree`. It defines variables like `PROJECT\_NAME` and `API\_KEY` from environment variables, and uses them to construct URLs for `CMBootstrap` and `S3Transfer`. The right pane shows the terminal session running under the user 'muser00' with session ID 'muser00\_session\_1'. It lists several pip package installations, notably `joblib` and `cmlladdons`, and highlights the download of `werkzeug` with an orange arrow pointing to it. The status bar at the bottom indicates the session is 'Running'.

```
File Edit View Navigate Run ▶ 0_bootstrap.py

1 ## Bootstrap File
2 # You need to at the start of the project. It will install the requirements, creates th
3 #e environment variable
4 # The STORAGE environment variable is the Cloud Storage location used by the DataLake t
5 #
6 # Install the requirements
7 !pip3 install -r requirements.txt

8 # Create the directories and upload data
9
10 from cibbootstrap import CMBootstrap
11 from IPython.display import Markdown, HTML
12 import os
13 import time
14 import datetime
15 import requests
16 import xml.etree.ElementTree as ET
17 import datetime
18
19 run_time_suffix = datetime.datetime.now()
20 run_time_suffix = run_time_suffix.strftime("%Y%m%d%H%M%S")
21
22 # Set the setup variables needed by CMBootstrap
23 HOST = os.getenv("COSM_API_URL").split(
24     "/")[-1]
25 PROJECT_NAME = os.getenv("COSM_PROJECT_NAME")
26 USERNAME = os.getenv("COSM_PROJECT_URL").split(
27     "/")[-1] # args.username
28 API_KEY = os.getenv("COSM_API_KEY")
29 PROJECT_NAME = os.getenv("COSM_PROJECT")
30
31 # Instantiate API wrapper
32 cm = CMBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
33
34 # Set the STORAGE environment variable
35 try:
36     storageos.environ['STORAGE']
37 except:
38     tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
39     root = tree.getroot()
40
41 for prop in root.findall('property'):
42     if prop.tag == 'name' and prop[1].text == 'storageos.sto...
```

You can open up requirements.txt to see what is loaded

# *Scikit learn for Machine learning Flask for Application serving*

All dependencies now live with the project and can be shared with others and your team through project forking. This may take a few minutes to install the requirements. At the end you will see the below output.

# Execute Data Ingest Script

In the same Workbench, open the script “code/1\_data\_ingest.py”

This will load the data from an S3 bucket using Spark. It demonstrates how to read from files and tables using Spark file and SQL operators.

The screenshot shows a Jupyter Notebook environment with two main panes. The left pane displays a file named `code/1_data_ingest.py`, which contains Python code for setting up a CMLBootstrap instance and managing storage environments. The right pane shows session details for "muser00\_session\_1", indicating it is running on a cluster with 2 vCPU and 4 GiB Memory. The session status is "Running". A large orange arrow points from the "Run All" button in the top right of the code editor to the "Run All" button in the session details pane.

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

0_bootstrap.py
muser00.telco.chum.projV1
project-metadata.yaml
cdsw-build.sh
churn_prediction.egg-info
cmr_catalog.yaml
code
0_bootstrap.py
1_data_ingest.py
2_data_exploration.ipynb
3_model_building.ipynb
4_train_models.py
5_model_serve_explainer.ipynb
6_application.py
7a_ml_ops_simulation.py
7b_ml_ops_visual.py
churnexplainer.py
logs
README.md
flask
images
LICENSE.txt
lineage.yml
logs
model_metrics.db
models
raw
README.md
requirements.txt
setup.py
src

100 #
101 # This was done for you
102 # It begins with impo
103 #
104 import os
105 import sys
106 import subprocess
107
108 from cmlbootstrap import CMLBootstrap
109 from pyspark.sql import SparkSession
110 from pyspark.sql.utils import AnalysisException
111 from pyspark.sql.types import *
112
113 # Set the setup variables needed by CMLBootstrap
114 HOST = os.getenv("CDSW_API_URL").split(":")[-1] + ":" + os.getenv("CDSW_DOMAIN")
115 USERNAME = os.getenv("CDSW_PROJECT_URL").split("/")[6] # args.username # vdibia
116 API_KEY = os.getenv("CDSW_PROJECT_KEY") # args.key
117 PROJECT_NAME = os.getenv("CDSW_PROJECT")
118
119 # Instantiate API Wrapper
120 cm1 = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
121
122 # Set the STORAGE environment variable
123 # TODO: this code block is repeated from 0_bootstrap.py -- fix this
124 try:
125     storage = os.environ["STORAGE"]
126     storage = os.environ["STORAGE"]
127     storage = "/user/" + os.getenv("HADOOP_USER_NAME")
128
129     # set the default external storage location
130     storage = "/user/" + os.getenv("HADOOP_USER_NAME")
131
132     # if available, set the external storage location for PBC
133     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
134         tree = ElementTree.parse("/etc/hadoop/conf/hive-site.xml")
135         root = tree.getroot()
136         for prop in root.findall("property"):
137             if prop.find("name").text == "hive.metastore.warehouse.dir":
138                 if os.getenv("previous_pvc_external_storage_locale"):
139                     if len(prop.find("value").text.split("/")) > 5:
140                         storage += (
141                             prop.find("value").text.split("/")[-1]
142                             +
143                             "/"
144                             +
145                             prop.find("value").text.split("/")[-2])
146
147     # create and set storage environment variables
148     storage_environment = cm1.create_environment_variable({"STORAGE": storage})
149     os.environ["STORAGE"] = storage
150
151
152     spark = SparkSession.builder.appName("PythonSQL").master("local[*]").getOrCreate()
153
154     # **Note**:
155     # Our file isn't big, so running it in Spark local mode is fine but you can add the fol
156     # if you want to run Spark on the kubernetes cluster
157
```

Click on Run → Run All.

Session output will show the code execution results. Observe the database, table, and data from the table.

ml-02030da5-5dc.partners.dp5i-5vkq.cloudera.site/partner10/telco-churn-10/engines/r7wvbz5xu5y5xg[b]

File Edit View Navigate Run ▶ 1\_data\_ingest.py

Telco Churn 10 ↗

0.build\_project.py  
1\_data\_ingest.py  
2\_data\_exploration.ipynb  
3\_train\_models.py  
4\_model\_exveExplained.py  
5\_application\_ipynb  
6\_codeBuild.sh  
7\_churnexplainer.ipynb  
8\_fraud.ipynb  
9\_models.ipynb  
10\_tsw.ipynb  
README.md  
requirements.txt  
utils

Untitled Session (Running)  
By partner10 - Python 3 Session - 1 vCPU / 2 GB Memory - just now

Session Logs Spark UI

```
1 import os
2 import sys
3 from pyspark.sql import SparkSession
4 from pyspark.sql.types import *
5 spark = SparkSession.builder \
6     .appName("PythonSQL") \
7     .master("local[*]") \
8     .getOrCreate()
9
10 # Add the following config if you want to run on the k8s cluster and remove "local[*]"
11 # .config(spark.hadoop.fs.s3a.s3guard.ddb.region, "us-east-1")
12 # .config(spark.yarn.access.hadoopFileSystems, "s3a://demo-aws-2//")
13
14
15 schema = StructType([
16     StructField("customerID", StringType(), True),
17     StructField("gender", StringType(), True),
18     StructField("SeniorCitizen", StringType(), True),
19     StructField("Partner", StringType(), True),
20     StructField("Dependents", StringType(), True),
21     StructField("tenure", DoubleType(), True),
22     StructField("PhoneService", StringType(), True),
23     StructField("MultipleLines", StringType(), True),
24     StructField("InternetService", StringType(), True),
25     StructField("OnlineSecurity", StringType(), True),
26     StructField("OnlineBackup", StringType(), True),
27     StructField("DeviceProtection", StringType(), True),
28     StructField("TechSupport", StringType(), True),
29     StructField("StreamingTV", StringType(), True),
30     StructField("StreamingMovies", StringType(), True),
31     StructField("PaperlessBilling", StringType(), True),
32     StructField("PaymentMethod", StringType(), True),
33     StructField("MonthlyCharges", DoubleType(), True),
34     StructField("TotalCharges", DoubleType(), True),
35     StructField("Churn", StringType(), True)
36 ])
37
38 s3_bucket = os.environ['STORAGE']
39
40 telco_data = spark.read.csv(
41     f'{s3_bucket}/data/churn/WA_Fn-UseC-Telco-Customer-Churn-.csv'.format(s3.bucket),
42     header=True,
43     inferSchema=True,
44     sep=',',
45     nullValue='N/A'
46 )
47
48 telco_data.show()
49 telco_data.printSchema()
50
51 telco_data.coalesce(1).write.csv(
52     f'{os.environ["HOME"]}/code/data/raw/telco-data/',
53     mode='overwrite',
54     header=True
55 )
56
57 spark.sql("show databases").show()
58 spark.sql("show tables in default").show()
```

this code is here to create the table in Hive used be the other parts of the project. It has already been run telco\_data  
write format('parquet')  
mode('overwrite')  
saveAsTable('default.telco\_churn')

```
> spark.sql("select * from default.telco_churn").show()
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
(7598-WHVG)	Female	No	No	No	1.0	No	No	No	No	No	No	No	No	No	No	No	No	29.85	Yes	
No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	29.85	No	
SPS-0VDE	Male	No	No	No	34.0	No	Yes	No	No	No	No	No	No	No	No	DSL	Yes	56.95	1889	
Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Mailed check	Yes	56.95	No	
(3668-OPVK)	Male	No	No	No	No	No	No	No	No	No	No	No	No	No	No	DSL	Mailed check	Yes	53.85	188.15
Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	DSL	Yes	53.85	No	
(7795-CFOC)	Male	No	No	No	45.0	No	No	No	No	No	No	No	No	No	No	DSL	Yes	No	No	

Line 1, Column 1

\* 70 Lines Python Spaces 2

Also examine the logs and Spark UI for details of the run.

The screenshot shows the Apache Spark UI interface. At the top, it says "Untitled Session" (Running) and "By partner10 – Python 3 Session – 1 vCPU / 2 GiB Memory – just now". Below this, there are tabs for "Session", "Logs", and "Spark UI", with "Spark UI" being active. The main content area is titled "Spark Jobs (3)". It shows the following information for three completed jobs:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2020/04/22 01:59:49	0.8 s	1/1	1/1
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2020/04/22 01:59:46	0.6 s	1/1	1/1
0	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2020/04/22 01:59:45	0.4 s	1/1	1/1

This concludes Lab 2 - Data Ingest.

## Lab 3 - Data Exploration

---

In this lab, you will explore some dataset using a different editor from the previous lab.

In fact, in this lab we are going to use a popular notebook, **Jupyter**, to show the flexibility of CML.

### Execute the data exploration script

Start a “**NEW SESSION**” and use the below configuration.

**Editor:** Jupyter Notebook

**Engine Profile:** *2vCPU x 4GiB Memory*

## Start A New Session

Session Name **Session Name**  
fubon20\_session2

---

Runtime **Editor**  
Editor ⓘ Kernel ⓘ  
JupyterLab Python 3.7

Edition ⓘ Version  
Standard 2022.11

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

Enable Spark ⓘ **Spark Version**  
Spark 2.4.8 - CDE 1.17 - HOTFIX-1

---

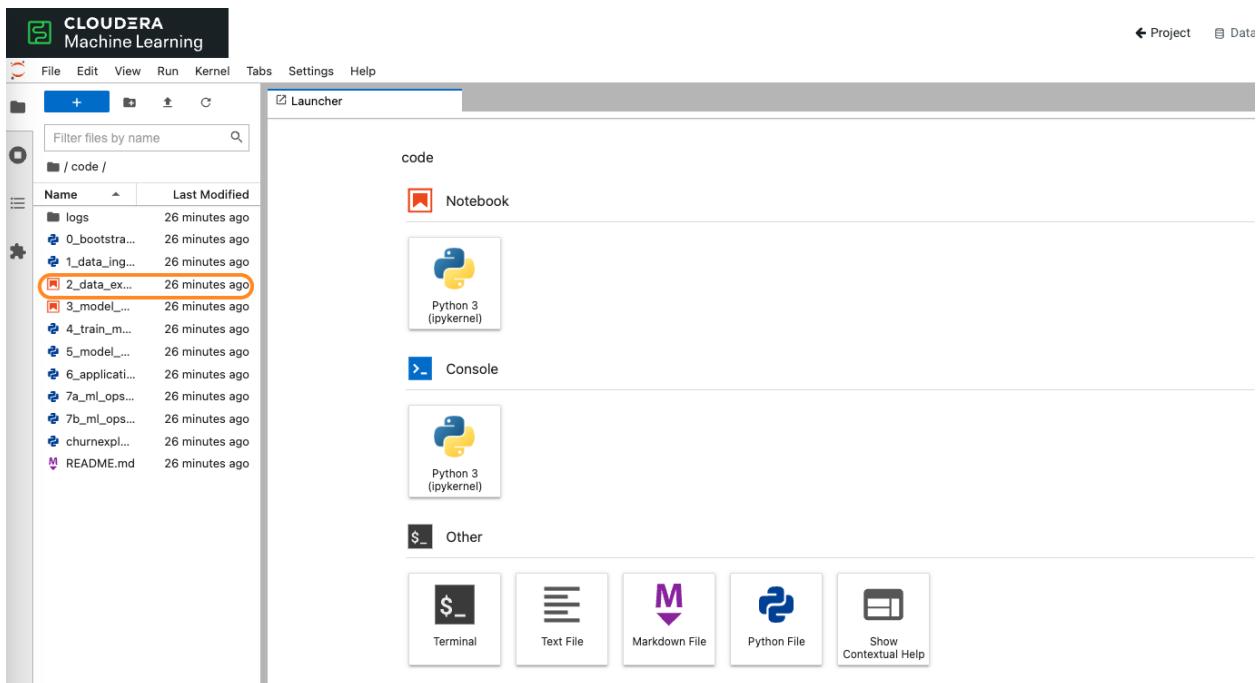
Runtime Image  
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-jupyterlab-python3.7-standard:2022.11.2-b2

---

Resource Profile **Resource Profile**  
2 vCPU / 4 GiB Memory

---

**Start Session** 



Then click on the `code/2_data_exploration.ipynb`.

And it will take you into the notebook

The screenshot shows a Jupyter Notebook titled "Telco Data Exploration". The code in cell [1] imports SparkSession and creates a DataFrame from a local file. Cell [2] runs a SQL query to select all columns from the "telco\_churn" table and converts it to a Pandas DataFrame. The resulting DataFrame is displayed in cell [2]. The data shows various customer details like gender, tenure, and service usage. A note at the bottom says "7043 rows x 21 columns". Below the DataFrame, there's a section titled "Basic DataFrame operations" with a link to the documentation.

```

In [1]: # Load the data
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("Telco Data Set")\
    .master("local[*]") \
    .getOrCreate()

In [2]: telco_data_raw = spark.sql("SELECT * FROM `default`.`telco_churn`")
telco_data_raw.toPandas()

Out[2]:
customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  InternetService  OnlineSecurity  ...  DeviceProtection  TechSupport
0  7590-VHVEG  Female        0   Yes       No  1.0      No  No phone service  DSL  No  ...  No
1  5575-GNVDIE  Male         0   No       No  34.0     Yes  No  DSL  Yes  ...  Yes
2  3668-QPYBK  Male         0   No       No  2.0      Yes  No  DSL  Yes  ...  No
3  7795-CFOCW  Male         0   No       No  45.0     No  No phone service  DSL  Yes  ...  Yes
4  9237-HQITU  Female        0   No       No  2.0      Yes  No  Fiber optic  No  ...  No
...
7038  6840-RESVB  Male         0   Yes      Yes  24.0     Yes  Yes  DSL  Yes  ...  Yes
7039  2234-XADUH  Female        0   Yes      Yes  72.0     Yes  Yes  Fiber optic  No  ...  Yes
7040  4801-JZAZL  Female        0   Yes      Yes  11.0     No  No phone service  DSL  Yes  ...  No
7041  8361-LTMKD  Male         1   Yes      No  4.0      Yes  Yes  Fiber optic  No  ...  No
7042  3186-AJIEK  Male         0   No       No  66.0     Yes  No  Fiber optic  Yes  ...  Yes
7043 rows x 21 columns

Basic DataFrame operations
Dataframes essentially allow you to express sql-like statements. We can filter, count, and so on. Documentation - (http://spark.apache.org/docs/latest/sql-programming-guide.html#dataframe-operations)

In [3]: "number of lines in dataset : {:#d}".format(telco_data_raw.count())
Out[3]: 'number of lines in dataset : 7043'

```

As you notice we are interacting with the data lake, in particular with the database previously created

### Telco Data Exploration

```
In [1]: # Load the data
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("Telco Data Set")\
    .master("local[*]") \
    .getOrCreate()

In [2]: telco_data_raw = spark.sql("SELECT * FROM `default`.`telco_churn`")
telco_data_raw.toPandas()
```

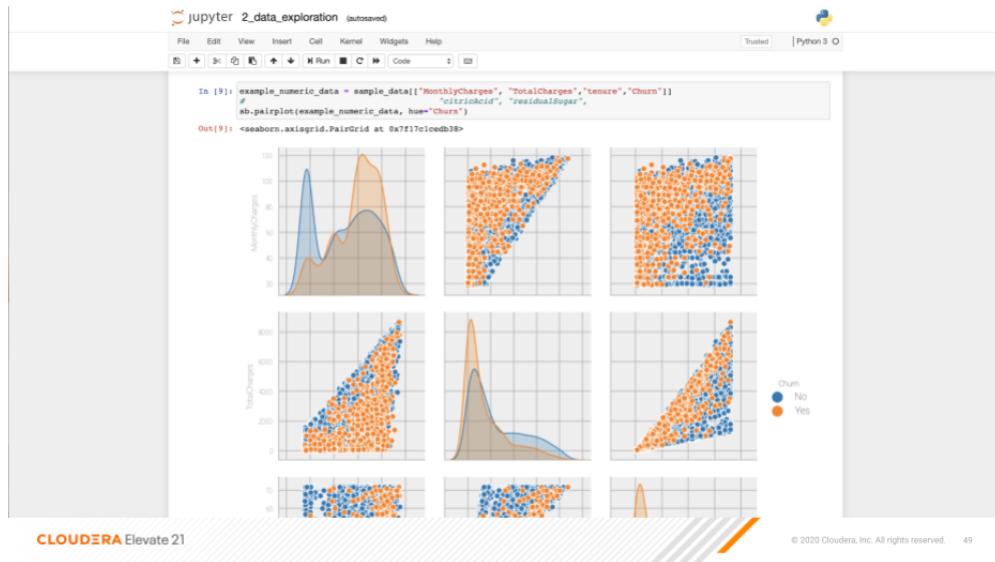
Out[2]:

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Telco Data Exploration
- Code Cells:**
  - In [1]: # Load the data
 from pyspark.sql import SparkSession
 spark = SparkSession\
 .builder\
 .appName("Telco Data Set")\
 .master("local[\*]") \
 .getOrCreate()
  - In [2]: telco\_data\_raw = spark.sql("SELECT \* FROM `default`.`telco\_churn`")
 telco\_data\_raw.toPandas()
- Output Cell:** Out[2]: A Pandas DataFrame titled "Exploration" showing the first 5 rows of the telco\_churn dataset.
- File Explorer:** On the left, it shows a file tree with files like 2\_data\_exploration.ipynb, 3\_model\_building.ipynb, etc.
- Kernel Menu:** A context menu is open over the second code cell, showing options like "Run All Cells", "Restart Kernel and Run All Cells...", and "Run All Above Selected Cell".

You are ready to run the notebook, go to *Cell, Run All*

And you can analyze the plotted graphs:



Now we can go back to *Project*

This concludes this lab.

## Lab 4 - Model Training and Experiment

In this lab, you will build and train the model, using the Experiment feature from CML that allows you to run offline different training sessions, with different parameters configuration, for your model so that you could promote in “Production” that configuration that showed the best results, KPIs.

### Build the model

First we will build the model. We will use a Jupyter Notebook to show the process of selecting and building the model to predict churn. It also shows more details on how the LIME model is created and a bit more on what LIME is actually doing.

Open a Jupyter Notebook session (rather than a workbench): python 3.7, 2 vCPU, 4 GiB and open the `3_model_building.ipynb` file.

At the top of the page click **Cells > Run All**.

### Model Training and Experiments

For the training portion of the lab we will use the file `4_train_models.py`

Name	Size	Last Modified
<code>logs</code>	-	an hour ago
<code>0_bootstrap.py</code>	6.92 kB	an hour ago
<code>1_data_ingest.py</code>	12.14 kB	25 minutes ago
<code>2_data_exploration.ipynb</code>	320.71 kB	20 minutes ago
<code>3_model_building.ipynb</code>	84.42 kB	18 minutes ago
<code>4_train_models.py</code>	8.10 kB	an hour ago
<code>5_model_serve_explainer.py</code>	10.00 kB	an hour ago
<code>6_application.py</code>	9.82 kB	an hour ago
<code>7a_ml_ops_simulation.py</code>	11.55 kB	an hour ago
<code>7b_ml_ops_visual.py</code>	5.56 kB	an hour ago
<code>churnexplainer.py</code>	8.64 kB	an hour ago
<code>README.md</code>	9.22 kB	an hour ago

Click on it and familiarize yourself with the code.

To give Data Scientists flexibility to collect, record, and compare experiment runs, CML provides out-of-the-box mlflow Experiments as a framework to achieve this.

Inside a running Workbench session, navigate to code/4\_train\_model.py. This script uses “kernel” and “max\_iter” as the two parameters to manipulate during model training in order to achieve the best result. In our case, we’ll define “best” as the highest “test\_score”.

To compare experiments, click on Experiments and select **Churn Model Tuning** experiment.

The screenshot shows the CML interface with the following steps highlighted:

- Step 1:** The "Experiments" tab is selected in the sidebar.
- Step 2:** The "Churn Model Tuning" experiment is selected in the list of experiments.

The main view displays the experiment details and a table of runs. The table includes columns for Status, Start Time, Run Name, Duration, User, Source, Version, Models, Kernel, Max\_Iter, ct, Pipeline\_sco, Pipeline\_sco, test\_score, estimator\_na, and estimator\_cl. A search bar at the top of the table filters results for "metrics.rmse < 1 and params.model = "true" and tags.miflow.source.type="LOCAL"".

Status	Start Time	Run Name	Duration	User	Source	Version	Models	Kernel	Max_Iter	ct	Pipeline_sco	Pipeline_sco	test_score	estimator_na	estimator_cl
<input checked="" type="checkbox"/>	2023-03-07 04:28:0	72ll-v03...	5.1s	ankitshar...		ada9b5	sklearn	linear	1	ColumnTr...	0.58873...	0.60599...	0.59	Pipeline	sklearn.pi...
<input checked="" type="checkbox"/>	2023-03-07 04:28:1	wmmw-8...	3.2s	ankitshar...		ada9b5	sklearn	linear	10	ColumnTr...	0.47212...	0.47061...	0.47	Pipeline	sklearn.pi...
<input checked="" type="checkbox"/>	2023-03-07 04:28:1	6n5u-58j...	3.8s	ankitshar...		ada9b5	sklearn	linear	100	ColumnTr...	0.65529...	0.65529...	0.66	Pipeline	sklearn.pi...
<input checked="" type="checkbox"/>	2023-03-07 04:28:2	483v-gh...	7.2s	ankitshar...		ada9b5	sklearn	linear	1000	ColumnTr...	0.63594...	0.64277...	0.64	Pipeline	sklearn.pi...
<input checked="" type="checkbox"/>	2023-03-07 04:28:2	7oz3-ocp...	13.9s	ankitshar...		ada9b5	sklearn	linear	10000	ColumnTr...	0.79180...	0.80204...	0.79	Pipeline	sklearn.pi...
<input type="checkbox"/>	2023-03-07 04:28:4	rw8l-wq0...	3.1s	ankitshar...		ada9b5	sklearn	rbf	1	ColumnTr...	0.68828...	0.68638...	0.69	Pipeline	sklearn.pi...
<input type="checkbox"/>	2023-03-07 04:28:4	lb0-lrt-...	3.2s	ankitshar...		ada9b5	sklearn	rbf	10	ColumnTr...	0.67463...	0.66780...	0.67	Pipeline	sklearn.pi...
<input type="checkbox"/>	2023-03-07 04:28:4	141v-yg...	3.9s	ankitshar...		ada9b5	sklearn	rbf	100	ColumnTr...	0.74288...	0.73966...	0.74	Pipeline	sklearn.pi...
<input type="checkbox"/>	2023-03-07 04:28:5	wgvt-dvz...	11.0s	ankitshar...		ada9b5	sklearn	rbf	1000	ColumnTr...	0.77531...	0.81494...	0.78	Pipeline	sklearn.pi...
<input type="checkbox"/>	2023-03-07 04:29:0	lbgy-yolc...	13.5s	ankitshar...		ada9b5	sklearn	rbf	10000	ColumnTr...	0.79180...	0.82233...	0.79	Pipeline	sklearn.pi...

As expected, a higher number of max\_iterations produces better results (higher test\_score). Interestingly, the choice of kernel does not make a difference at higher max\_iter values. We can choose linear as it allows for faster model training.

## Compare Experiments

This feature provides a built-in visualizations in mlflow allow for more detailed comparison of various experiment runs and outcomes.

- Select all runs with “linear” Kernel.
- Click on Compare

**Experiment** BETA

Experiment Name: Churn Model Tuning

Experiment ID: 2g9v-8q5u-ah2q-jzvb

Artifact Location: /home/cdsw/experiments/2g9v-8q5u-ah2q-jzvb

> Notes

Runs (10)

Run ID	Status	Start Time	Run Name	Duration	User	Source	Version	Models	Kernel	Max_iter	ct	Pipeline_sco	Pipeline_sco	test_score	estimator_na	estimator_cl	Tags
2023-03-07 04:28:0	✓	2023-03-07 04:28:0	72ll-v03w-rqlz-0kk3	5.1s	ankitshar...	sklearn	ada9b5	sklearn	linear	1	ColumnTr...	0.58873...	0.60599...	0.59	Pipeline	sklearn.pi...	
2023-03-07 04:28:1	✓	2023-03-07 04:28:1	wmmw-8eu...	3.2s	ankitshar...	sklearn	ada9b5	sklearn	linear	10	ColumnTr...	0.47212...	0.47061...	0.47	Pipeline	sklearn.pi...	
2023-03-07 04:28:1	✓	2023-03-07 04:28:1	6n5u-58j...	3.8s	ankitshar...	sklearn	ada9b5	sklearn	linear	100	ColumnTr...	0.65529...	0.65529...	0.66	Pipeline	sklearn.pi...	
2023-03-07 04:28:2	✓	2023-03-07 04:28:2	483v-gh...	7.2s	ankitshar...	sklearn	ada9b5	sklearn	linear	1000	ColumnTr...	0.63594...	0.64277...	0.64	Pipeline	sklearn.pi...	
2023-03-07 04:28:2	✓	2023-03-07 04:28:2	7oz3-ocp...	13.9s	ankitshar...	sklearn	ada9b5	sklearn	linear	10000	ColumnTr...	0.79180...	0.80204...	0.79	Pipeline	sklearn.pi...	
2023-03-07 04:28:4	✓	2023-03-07 04:28:4	rw8l-wq0...	3.1s	ankitshar...	sklearn	ada9b5	sklearn	rbf	1	ColumnTr...	0.68828...	0.68638...	0.69	Pipeline	sklearn.pi...	
2023-03-07 04:28:4	✓	2023-03-07 04:28:4	l8y0-ljt-...	3.2s	ankitshar...	sklearn	ada9b5	sklearn	rbf	10	ColumnTr...	0.67463...	0.66780...	0.67	Pipeline	sklearn.pi...	
2023-03-07 04:28:4	✓	2023-03-07 04:28:4	t41v-ygv...	3.9s	ankitshar...	sklearn	ada9b5	sklearn	rbf	100	ColumnTr...	0.74288...	0.73986...	0.74	Pipeline	sklearn.pi...	
...	...	...	...	...	...	...	...	...	...	1000	ColumnTr...	0.77654...	0.81004...	0.78	Pipeline	sklearn.pi...	

← Back to all runs

### Run Comparison (5)

Run ID	72ll-v03w-rqlz-0kk3	wmmw-8eu...	6n5u-58j3-jwq1-j929	483v-gh...	7oz3-ocp...
Run Name	72ll-v03w-rqlz-0kk3	wmmw-8eu...	6n5u-58j3-jwq1-j929	483v-gh...	7oz3-ocp...
Start Time	2023-03-07 04:28:09	2023-03-07 04:28:14	2023-03-07 04:28:17	2023-03-07 04:28:21	2023-03-07 04:28:29

Parameters

Parameter	72ll-v03w-rqlz-0kk3	wmmw-8eu...	6n5u-58j3-jwq1-j929	483v-gh...	7oz3-ocp...
Kernel	linear	linear	linear	linear	linear
Max_iter	1	10	100	1000	10000
ct	ColumnTransformer(n_jobs=None, remainder='passthrough', sparse_threshold=0.3, transformer_weights=None, transformers=[('ohe', OneHotEncoder(categorical_features='all'))])				
ct_n_jobs	None	None	None	None	None
ct_ohe	OneHotEncoder(categorical_features='all', categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features='all', categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features='all', categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features='all', categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features='all', categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)

## Lab 5 - Model Deploy/Serve

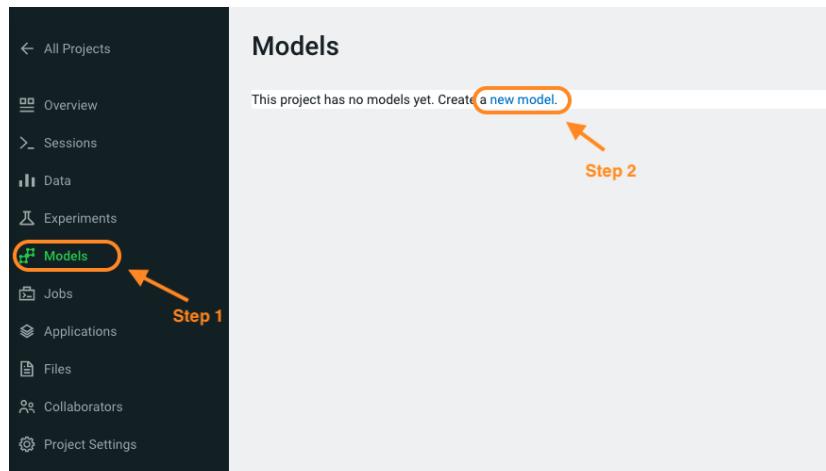
Once a model is trained its predictions and insights must be put to use so they can add value to the organization. Generally this means using the model on new, unseen data in a production environment that offers key ML Ops capabilities.

In this lab, you will deploy/serve the model that you have trained in the Lab 3 as a REST endpoint. The model can be invoked as-needed, in real-time or batch fashion, by external services that need to score the prediction implemented by the model.

### Build a Model

Below are the steps to deploy a near-real-time scoring model:

- Click on Models in the side panel
- Create a new model.



- Name your model *Churn Model API Endpoint*. Any other name will cause issues with downstream scripts.
- Uncheck Enable Authentication.
- Under File select *code/5\_model\_serve\_explainer.py*
- Under Function enter *explain*
- For Example Input enter the following JSON
- You do not need to Enable Spark for model serving in this case
- Deploy the model by clicking *Deploy model*

```
{  
    "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"  
}
```

Name \*

Description \*

Enable Authentication

ⓘ Enforces model API requests to be authenticated with an API key. ⓘ

### Build

File \*

Function \*

Example Input ⓘ

```
{  
    "StreamingTV": "No",  
    "MonthlyCharges": 70.35,  
    "PhoneService": "No",  
    "PaperlessBilling": "No",  
    "Partner": "No",  
}
```

Example Output ⓘ

```
{ "result": "value" }
```

### Runtime

Editor ⓘ	Kernel ⓘ	Edition ⓘ	Version
JupyterLab	Python 3.7	Standard	2022.11

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

Enable Spark ⓘ

ⓘ Spark 2.4.8 - CDE 1.17 - HOTFIX-1

## Deploy a Model

The status will go thru the life-cycle of *Pending -> Building -> Deploying -> Deployed*

Model	Source	Status	Replicas	CPU	Memory	Created By	Deployed By	Last Deployed	Actions
Churn Model API Endpoint	code/5...	Building	0 / 1	0	0 GiB	ankitsharma	ankitsharma	Mar 7, 2023, 05:24 PM	<button>Stop</button>

## Test a Model

- Once Model is deployed, click on Test

Description Churn Model API Endpoint

Sample Code

Shell Python R

```
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-2281c21a-70a.csa-work.d
p51-5vkq.cloudera.site/model -d '{"accessKey":"m70ss2gubs77eitce6wqo7hrd1z22r90", "request":{"Str
eamTV": "No", "MonthlyCharges": 70.35, "PhoneService": "No", "PaperlessBilling": "No", "Partner": "N
o", "OnlineBackup": "No", "gender": "Female", "Contract": "Month-to-month", "TotalCharges": 1397.475, "St
reamingMovies": "No", "DeviceProtection": "No", "PaymentMethod": "Bank transfer (automatic)", "tenur
e": 29, "Dependents": "No", "OnlineSecurity": "No", "MultipleLines": "No", "InternetService": "DSL", "Seni
orCitizen": "No", "TechSupport": "No"}}
or
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-2281c21a-70a.csa-work.d
p51-5vkq.cloudera.site/model?accessKey=m70ss2gubs77eitce6wqo7hrd1z22r90 -d '{"request":{"Strami
ngTV": "No", "MonthlyCharges": 70.35, "PhoneService": "No", "PaperlessBilling": "No", "Partner": "No", "On
lineBackup": "No", "gender": "Female", "Contract": "Month-to-month", "TotalCharges": 1397.475, "Streaming
Movies": "No", "DeviceProtection": "No", "PaymentMethod": "Bank transfer (automatic)", "tenure": 29, "Dep
endents": "No", "OnlineSecurity": "No", "MultipleLines": "No", "InternetService": "DSL", "SeniorCitiz
en": "No", "TechSupport": "No"}'}
```

Test Model

Input

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

**Test** **Reset**

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.

If you want to call the model from external services, the sample codes for invoking this REST endpoint are provided in Shell, Python and R.

[Description](#)

Churn Model API Endpoint

[Sample Code](#)[Shell](#)[Python](#)[R](#)

```
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-2281c21a-70a.csa-work.d  
p5i-5vkq.cloudera.site/model -d '{"accessKey":"m70ss2gubs77eitce6wqo7hrd1z22r9o","request":{"Str  
eamingTV":"No","MonthlyCharges":70.35,"PhoneService":"No","PaperlessBilling":"No","Partner":"N  
o","OnlineBackup":"No","gender":"Female","Contract":"Month-to-month","TotalCharges":1397.475,"St  
reamingMovies":"No","DeviceProtection":"No","PaymentMethod":"Bank transfer (automatic)","tenur  
e":29,"Dependents":"No","OnlineSecurity":"No","MultipleLines":"No","InternetService":"DSL","Seni  
orCitizen":"No","TechSupport":"No"}}  
or  
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-2281c21a-70a.csa-work.d  
p5i-5vkq.cloudera.site/model?accessKey=m70ss2gubs77eitce6wqo7hrd1z22r9o -d '{"request":{"Str  
eamingTV":"No","MonthlyCharges":70.35,"PhoneService":"No","PaperlessBilling":"No","Partner":"No","On  
lineBackup":"No","gender":"Female","Contract":"Month-to-month","TotalCharges":1397.475,"Streamin  
gMovies":"No","DeviceProtection":"No","PaymentMethod":"Bank transfer (automatic)","tenure":29,"D  
ependents":"No","OnlineSecurity":"No","MultipleLines":"No","InternetService":"DSL","SeniorCitize  
n":"No","TechSupport":"No"}}'
```

This concludes this lab.

# Lab 6 - Application Deployment

In this lab, you will create an application that embeds the model deployed in the previous lab, allowing business users, end-users that are not Data Scientists to interact and to get insight about the context of these analyses.

## Prerequisites

- Copy accesskey *Models -> Settings -> Access Key*
- Update accesskey into file. *Files -> flask -> single\_view.html (line : 61 )*

The screenshot shows the Cloudera Manager interface with the 'flask' project selected. On the left, there's a sidebar with various project management options like 'Overview', 'Sessions', 'Data', 'Experiments', 'Models', 'Jobs', 'Applications', and 'Files'. The 'Files' option is currently selected and highlighted with an orange border. The main area displays a file tree under 'flask'. The 'single\_view.html' file is explicitly highlighted with an orange border around its listing in the tree. Other files visible include 'ajax-loader.gif', 'churn\_vis.css', 'churn\_vis.js', 'env\_vars.png', and 'table\_view.html'. A table at the bottom provides details for each file, such as size and last modified time.

The screenshot shows the code editor for the 'single\_view.html' file. The code is a standard HTML document with some CSS and JavaScript imports. At line 61, there is a line of JavaScript code that defines an 'accessKey' constant. This line is highlighted with an orange border. The code also includes several other lines of comments and imports, such as 'd3.v5.min.js', 'lodash.min.js', and 'churn\_vis.css'.

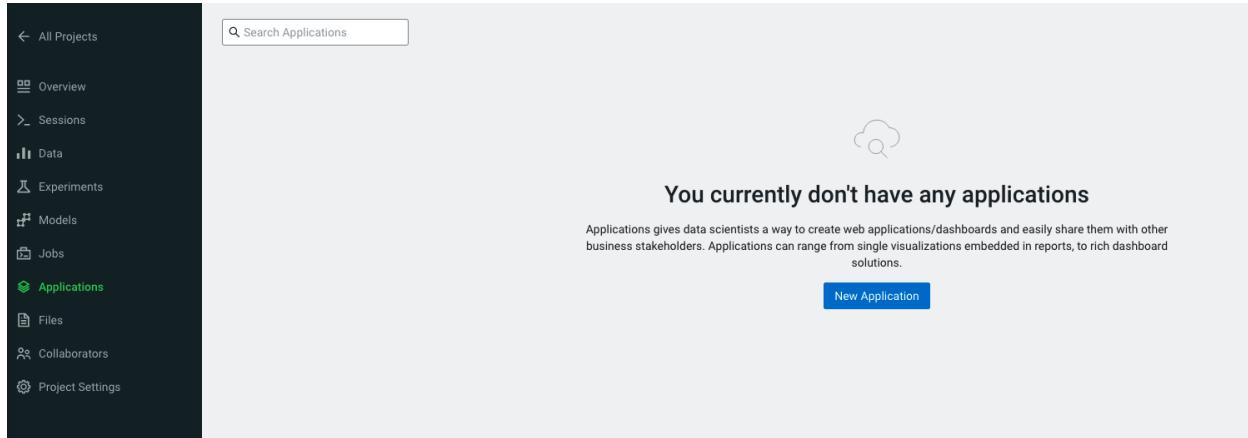
```

single_view.html
File Edit View Navigate Run flask/single_view.html
w001 .project-metadata.yaml
cdsw-build.sh
churn_prediction.egg-info
cml_catalog.yaml
code
flask
images
LICENSE.txt
lineage.yml
logs
model_metrics.db
models
raw
README.md
requirements.txt
setup.py
src
23 * have any rights to access nor to use this code.
24 *
25 * Absent a written agreement with Cloudera, Inc. ("Cloudera") to the
26 * contrary, (A) CLOUDERA PROVIDES THIS CODE TO YOU WITHOUT WARRANTIES OF ANY
27 * KIND; (B) CLOUDERA DISCLAIMS ANY AND ALL EXPRESS AND IMPLIED
28 * WARRANTIES WITH RESPECT TO THIS CODE, INCLUDING BUT NOT LIMITED TO
29 * IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY AND
30 * FITNESS FOR A PARTICULAR PURPOSE; (C) CLOUDERA IS NOT LIABLE TO YOU,
31 * AND WILL NOT DEFEND, INDEMNIFY, NOR HOLD YOU HARMLESS FOR ANY CLAIMS
32 * ARISING FROM OR RELATED TO THE CODE; AND (D) WITH RESPECT TO YOUR EXERCISE
33 * OF ANY RIGHTS GRANTED TO YOU FOR THE CODE, CLOUDERA IS NOT LIABLE FOR ANY
34 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, PUNITIVE OR
35 * CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, DAMAGES
36 * RELATED TO LOST REVENUE, LOST PROFITS, LOSS OF INCOME, LOSS OF
37 * BUSINESS ADVANTAGE OR UNAVAILABILITY, OR LOSS OR CORRUPTION OF
38 * DATA.
39 *
40 * ****
41 -->
42
43 <!DOCTYPE html>
44
45 <head>
46   <meta charset="utf-8">
47   <script src="https://d3js.org/d3.v5.min.js"></script>
48   <script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.11/lodash.min.js"></script>
49   <link rel="stylesheet" type="text/css" href="churn_vis.css">
50 </head>
51
52 <body>
53   <h1>Single Prediction View</h1>
54   <div style='clear:both;' class="churn_div">
55     <div style='float:left;padding-left:20px;'>Churn Probability</div>
56     <div id='pred_value'></div>
57   </div>
58   <div id='features' style='clear:both;'></div>
59   <script>
60     const accessKey = "mwbhuvqo4islnwdfjaaqtxgi8c66fdne";
61     in_url = new URL(window.location.href)
62     out_url = new URL(window.location.origin + window.location.pathname)
63
64

```

## Application Deployment

Go to *Applications* and create a new Application.



Name	Customer Churn Explainer-<uesrname>
Subdomain	churn-<uesrname>
Script	code/06_application.py
Editor	Workbench
Enable Spark	Yes

**General**

**Name**  
Customer Churn Explainer

**Subdomain \***  
churn-fubon20

**Description**  
fubon 20 Customer Churn Explainer

**Script \***  
code/6\_application.py

---

**Runtime**

**Editor** ⓘ Workbench

**Kernel** ⓘ Python 3.7

**Edition** ⓘ Standard

**Version** 2022.11

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

Enable Spark ⓘ Spark 2.4.8 - CDE 1.17 - HOTFIX-1 

**Runtime Image**  
- docker.repository.cloudera.com/cloudera/cdsweb/ml-runtime-workbench-python3.7-standard:2022.11.2-b2

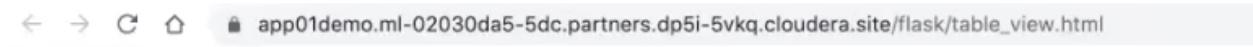
- Click on Create Application

Application startup can take up to 2 minutes, and once the application is ready you'll see a card similar to this:

The screenshot shows a card for a project named 'fubon99\_tel...'. It includes a logo, the title 'Customer Churn Explainer' with a blue checkmark icon, and a status message 'Running since a few seconds ago'. Below this, there are fields for 'Project' (fubon99\_tel...), 'Created by' (fubon99), and 'Last Updated' (04/24/2023 8:09 PM). There are also 'i' and '⋮' icons in the top right corner.

Click now in your newly created application

You can see the subdomain we have specified before as a prefix of your application url.



## Refractor

Loading Sample Data...

Once the application is loaded

Refractor		Customer Churn Explainer																							
ID	Probability	gender	SeniorCitizen	Partner	Dependents	Tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges					
3869	0.735	Male	No	No	22	Yes	Yes	Fiber	No	Yes	No	No	Yes	Month	Yes	Elect	100.6	2415.							
2410	0.407	Femal	Yes	Yes	40	Yes	Yes	Fiber	No	No	No	Yes	Yes	Month	Yes	Credi	94.55	3640.							
2166	0.383	Femal	No	No	2	Yes	No	DSL	No	No	Yes	No	No	Month	No	Elect	60.85	111.4							
3802	0.149	Femal	No	Yes	3	Yes	No	No	No in	No in	No in	No in	No in	No in	No in	Month	No	Mail	19.3	54.7					
2408	0.148	Male	No	Yes	7	No	No ph	DSL	Yes	No	Yes	Yes	No	Yes	Yes	Two y	No	Credi	49.65	305.5					
2020	0.147	Femal	No	Yes	72	Yes	Yes	Fiber	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Two y	Yes	Bank	116.8	8477.					
499	0.137	Male	No	No	34	Yes	Yes	Fiber	Yes	Yes	Yes	Yes	Yes	Yes	Yes	One y	No	Credi	116.2	3899.					
1648	0.132	Male	No	Yes	No	58	Yes	No	No	No in	No in	No in	No in	No in	No in	Two y	No	Mail	20.75	1185.					
3198	0.122	Femal	No	Yes	No	72	Yes	Yes	DSL	Yes	Yes	Yes	Yes	Yes	Yes	Two y	No	Bank	88.05	6520.					
6879	0.110	Male	No	No	53	Yes	No	DSL	No	No	Yes	No	No	Yes	Yes	One y	Yes	Elect	61.1	3357.					

Click on one of the item in the Probability column

>5i-5vkq.cloudera.site/flask/table\_v

id	Probability	gender	SeniorCitizen
1846	0.517	Femal	No
3872	0.436	Male	No
1242	0.402	Femal	No
1203	0.301	Femal	No
709	0.217	Male	No
6260	0.144	Femal	No
3569	0.113	Male	No
5661	0.111	Femal	No
3730	0.104	Femal	No
3382	0.030	Femal	No

To get the detailed view

## Single Prediction View

Churn Probability **0.718**

MonthlyCharges	35.1	0.32	mean 64.80 min 18.25 max 118.75	<input type="text"/>	<input type="button" value="Submit"/>
MultipleLines	No phone service	0	No No phone service Yes		
TechSupport	No	0	No No internet service Yes		
PhoneService	No	-0.04	No Yes		
OnlineBackup	No	0	No No internet service Yes		
Contract	Month-to-month	0.13	Month-to-month One year Two year		
PaperlessBilling	Yes	0	No Yes		
InternetService	DSL	-0.17	DSL Fiber optic No		
PaymentMethod	Electronic check	0.04	Bank transfer (automatic) Credit card (automatic) Electronic check Mailed check		
StreamingMovies	Yes	0.09	No No internet service Yes		
TotalCharges	68.75	-0.12	mean 2283.30 min 18.80 max 8684.80	<input type="text"/>	<input type="button" value="Submit"/>
tenure	2	0.29	mean 32.42 min 1.00 max 72.00	<input type="text"/>	<input type="button" value="Submit"/>
gender	Female	0	Female Male		
SeniorCitizen	No	-0.04	No Yes		
StreamingTV	No	0	No No internet service Yes		
DeviceProtection	No	0	No No internet service Yes		
OnlineSecurity	No	0.04	No No internet service Yes		
Partner	Yes	0	No Yes		
Dependents	No	0	No Yes		

If you change some of the values, that will also change the churn probability by calling the model we have deployed in the previous lab.

**This concludes the lab and workshop**