

# CML Workshop Student Guide

---

<b>Introduction</b>	<b>2</b>
<b>Labs Summary</b>	<b>3</b>
<b>Lab 1 - Initial setup</b>	<b>4</b>
Getting Connected	4
Setting Up Workload Password	4
Create CML Project	7
Workload Password	8
Create a New Project	8
Project Environment Variables	11
<b>Lab 2 - Data Ingest</b>	<b>12</b>
Run the Bootstrap Script	12
Execute Data Ingest Script	15
<b>Lab 3 - Data Exploration</b>	<b>18</b>
Execute the data exploration script	18
<b>Lab 4 - Model Training and Experiment</b>	<b>23</b>
Build the model	23
Model Training and Experiments	23
Compare Experiments	25
<b>Lab 5 - Model Deploy/Serve</b>	<b>26</b>
Build a Model	26
Deploy a Model	29
Test a Model	29
<b>Lab 6 - Application Deployment</b>	<b>31</b>
Prerequisites	31
Application Deployment	31
<b>Demo - Applied ML Prototypes (AMPs)</b>	<b>36</b>
Demo- Model Lineage Tracking	37

# 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.

The screenshot displays two main components of the FF06 Hands On Lab:

- REPORT**: A dashboard titled "Interpretability" which includes:
  - A heatmap titled "Churn Prediction" showing churn probability across various dimensions like Device, Internet, Streami, TV, Online, Tech, and Partner.
  - A flowchart illustrating the machine learning pipeline: Training Data → Training → Model.
  - A section titled "Model with Global Interpretability" comparing "Local Interpretability" (red) and "Global Interpretability" (blue).
  - A note: "Model with Local Interpretability" is highlighted in red.
  - A section titled "The Power of Interpreting" with a note: "Interpretability can prevent overfitting."
  - A note: "Model with Local Interpretability" is highlighted in blue.
  - A section titled "J2 Enhancing Trust" with a note: "Data scientists have a well-established process to make sure their models are of a good, valuable. They train a model with perhaps lots of their training data, then measure its performance on the test data. By assessing the model's performance on the test data, they can ensure that a powerful model with a lot of flexibility will adequately generalize the training data."
  - A note: "Model with Local Interpretability" is highlighted in red.
  - A note: "The Power of Interpreting" is highlighted in blue.
- REFRACTOR**: An application interface showing a table of customer data with columns including ID, Churn Prob., Phone Service, Internet Service, Multi-line, Streami, Streami TV, Online Socia..., Online Backup, Tech Supp..., Device Prot..., Con-tract, Pay-ments, Paperless, Month-ly, Total Charge, Tenure, and Partner.

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

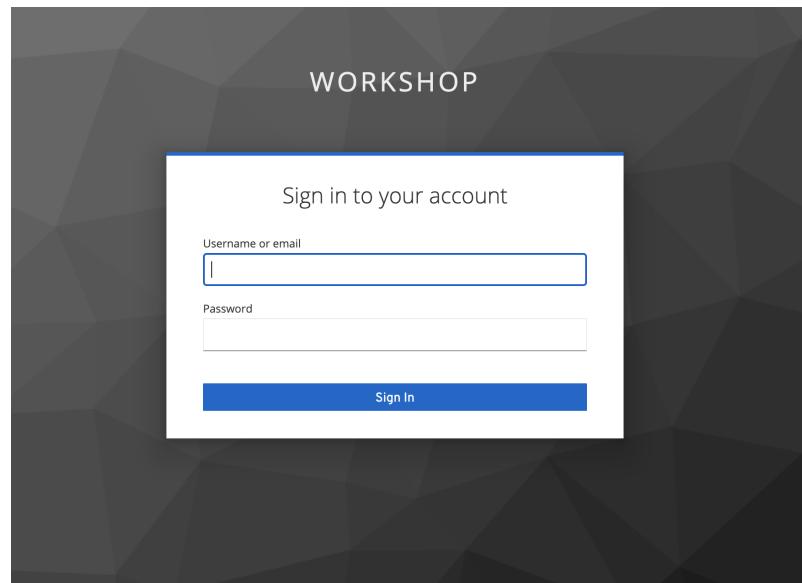
## Lab 1 - Initial setup

### Getting Connected

<b>CDP Console URL</b>	<a href="https://54.169.212.131:8443/auth/realms/uob-workshop/protocol/saml/clients/samlclient">https://54.169.212.131:8443/auth/realms/uob-workshop/protocol/saml/clients/samlclient</a>
<b>User</b>	Refer Google sheet
<b>Password</b>	Refer Google sheet
<b>Project Github URL</b>	<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

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

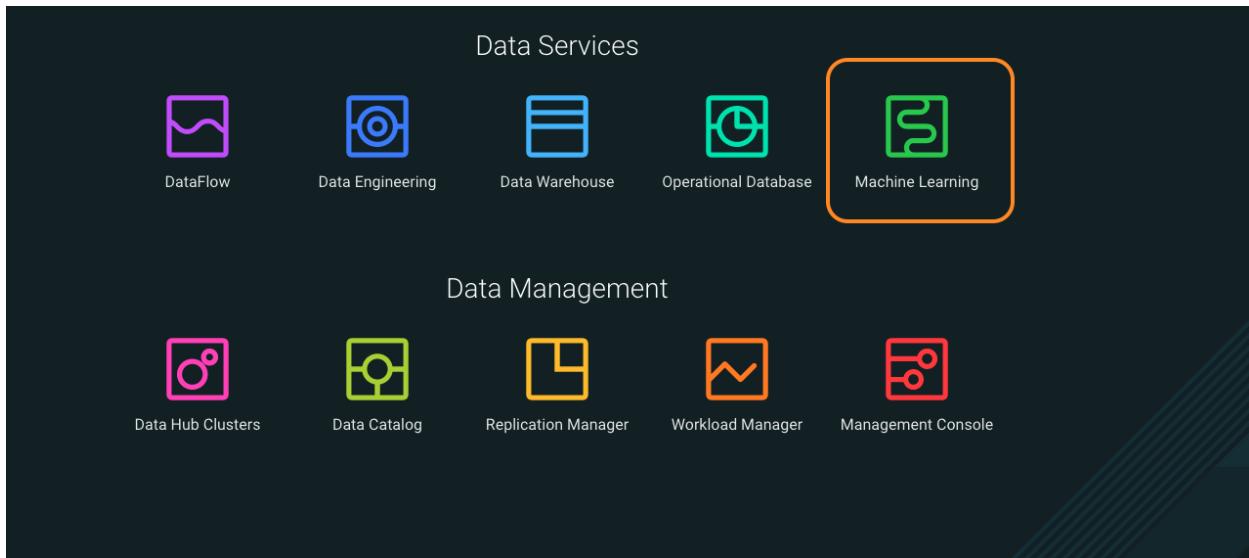


- You should land on the CDP Console as shown below.



## 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 workspace. Click on Workspace to proceed to executing the hands-on labs.



A screenshot of the Cloudera Management Console Environments list screen. It shows a table with columns: Status, Name, Cloud Provider, Region, Data Lake, CDP Runtime Version, and Time Created. One row is selected: Available, csa-workshop, aws, Asia Pacific (Mumbai), Running, 7.2.16, 2/22/2023, 11:18:50 AM GMT+5:30. A red arrow points from the "ML Workspaces" link in the sidebar to the "Environments" link in the header. The sidebar also lists Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters, Data Warehouses, ML Workspaces (which is highlighted with an orange border), and Classic Clusters.

This will launch a ML workspace screen.

A screenshot of the Cloudera Machine Learning Workspaces screen. It shows a table with columns: Status, Workspace, Environment, Region, Creation Date, Cloud Provider, and Actions. One row is selected: Ready, csa-cml-workshop, csa-workshop, ap-south-1, 03/03/2023 7:49 PM IST, AWS. A red box highlights the "Workspace" and "Environment" columns for the selected row. The sidebar on the left includes Workspaces (highlighted with an orange border) and Workspace Backups.

## Workload Password

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

## Create a New Project

Create a new Project based on Github.

**Project Name :** <username>\_telco\_churn\_projV1

**Initial Setup :** Git

**Git URL :** [https://github.com/ogakulov/CML\\_AMP\\_Churn\\_Prediction\\_mlflow](https://github.com/ogakulov/CML_AMP_Churn_Prediction_mlflow)

\* Project Name **Project Name**  
muser00\_telco\_churn\_projV1 

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** 

**Git Project**

Git URL of Project ⓘ  
**Git URL**  
`https://github.com/ogakulov/CML_AMP_Churn_Prediction_mlflow` 

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:  
JupyterLab - Python 3.7 - Standard - 2022.11  
PBJ Workbench - Python 3.7 - Standard - 2022.11  
Workbench - Python 3.7 - Standard - 2022.11

**Create Project** 

The screenshot shows the Cloudera Machine Learning interface. On the left is a dark sidebar with various navigation options: Projects (selected), Sessions, Experiments, Models, Jobs, Applications, User Settings, AMPs, Runtime Catalog, Site Administration, and Learning Hub. The main area is titled 'Projects' and shows a search bar with 'Search Projects' and dropdowns for 'Scope: My Projects' and 'Creator: All'. A single project card is visible, titled 'muser00\_telco\_churn\_...', which was created by 'ankitsharma' 36 minutes ago. There are also 'Edit' and 'Delete' buttons next to the project name.

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

The screenshot shows the 'Files' view for the project 'muser00\_telco\_churn\_projV1'. The sidebar on the left is identical to the previous screenshot. The main area shows a list of files and directories under 'muser00\_telco\_churn\_projV1'. The files listed include: churn\_prediction.egg-info, code, flask, images, logs, models, raw, src, cdswebuild.sh, cml\_catalog.yaml, LICENSE.txt, lineage.yaml, model\_metrics.db, README.md, requirements.txt, and setup.py. Each item has a 'Name' column, a 'Size' column (e.g., 45 B, 1.01 kB, 9.94 kB, etc.), and a 'Last Modified' column (all entries are 'a few seconds ago'). There are also 'Edit' and 'Delete' icons for each file entry. At the top right of the file list, there are buttons for 'Download', 'New', and 'Upload'.

## Project Environment Variables

The screenshot shows the 'Project Settings' page for a project. On the left is a sidebar with links like 'All Projects', 'Overview', 'Sessions', 'Data', 'Experiments', 'Models', 'Jobs', 'Applications', 'Files', 'Collaborators', and 'Project Settings'. A red arrow points to the 'Project Settings' link. The main area is titled 'Project Settings' and has tabs for 'Options', 'Runtime/Engine', 'Advanced' (which is selected), 'SSH Tunnels', 'Data Connections', 'Prototype', and 'Delete Project'. Under 'Advanced', there's a section for 'Environment Variables' with a note: 'Set project environment variables that can be accessed from your scripts.' It says '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.' There are five environment variable entries:

Variable Name	Value	Actions
DATA_LOCATION	data/churn_prototype	- +
HIVE_DATABASE	default	- +
HIVE_TABLE	churn_prototype	- +
CDSW_APP_POLLING_ENDPOINT	/	- +
PROJECT_OWNER	ankitsharma	- +

Add below variable in Project Settings

Variable Name	Value
DATA_LOCATION	my-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.

**Editor:** Workbench

**Engine Kernel:** Python 3

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

**Enable Spark :** Yes ( Default Version )

muser00\_telco\_churn\_projV1

0 Fork New Session

Models

This project has no models yet. Create a [new model](#).

Jobs

This project has no jobs yet. Create a [new job](#) to document your analytics pipelines.

Files

Name	Size	Last Modified
flask	-	37 minutes ago
images	-	37 minutes ago
models	-	37 minutes ago
raw	-	37 minutes ago

### Start A New Session

**Session Name**

**Runtime**

**Editor**

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

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

**Enable Spark** (checkbox)    **Spark Version**: Spark 2.4.8 - CDE 1.17 - HOTFIX-1

**Runtime Image**

- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2022.11.1-b2

**Resource Profile**

**Resource Profile**

2 vCPU / 4 GiB Memory

Click on **Start Session**

**Connection Code Snippet**

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

csa-workshop-datalake	CSA-WORKSHOP-VW	csa-workshop-impala
TYPE Spark Data Lake	TYPE Hive Virtual Warehouse	TYPE Impala Virtual Warehouse

Use this code to connect to the chosen data source.

```

import cml.data_v1 as cmldata
cmldata.set_project('CSA WORKSHOP')

# Set connection name
CONNECTION_NAME = 'csa-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**

X

**Close the Code Snippet**

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

The screenshot shows the Cloudera Manager interface. On the left, there's a file browser for a project named 'muser00\_telco\_churn\_projV1'. Inside the 'code' directory, the '0\_bootstrap.py' file is selected. On the right, there's a code editor window titled 'code/0\_bootstrap.py' with the following content:

```

0_bootstrap.py

muser00_telco_churn_projV1
  - project-metadata.yaml
  - cdsw-build.sh
  - churn-prediction.egg-info
    - cml_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.py
      - 6_application.py
      - 7a_ml_ops_simulation.py
      - 7b_ml_ops_visual.py
      - churnexplainer.py
    - logs
      - README.md
  - flask
  - images
    - LICENSE.txt
    - image.yml
  - logs
    - model_metrics.db
  - models
  - raw
    - README.md
  - requirements.txt
  - setup.py
  - src

```

```

# #########################################################################
# # 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
# # individually copyrighted by its original author(s) and/or
# # collective copyright holders. Your license to use the
# # collective work as provided for in your written agreement with
# # Cloudera is granted under the open source license identified above.
# #
# # 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 access nor to use this code.
# #
# # Absent a written agreement with Cloudera, Inc. ("Cloudera") to the
# # contrary, A) CLOUDERA PROVIDES THIS CODE AS IS WITHOUT WARRANTY 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
# # FOR ANY DAMAGES, LOSSES, EXPENSES OR COSTS ARISING FROM YOUR EXERCISE
# # OF ANY RIGHTS GRANTED TO YOU FOR THE CODE; AND (D)WITH RESPECT TO YOUR EXERCISE
# # OF ANY RIGHTS GRANTED TO YOU FOR THE CODE, CLOUDERA IS NOT LIABLE FOR ANY
# # 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 set this at the start of the project. It will install the requirements,
# # set the storage environment variables and copy the data from
# # raw/VA_Fn-Used-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 s3a://[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
# Create the directories and upload data
from cmlbootstrap import CMLBootstrap
from IPython.display import Javascript, HTML
import os
import time
import json
import requests
import xml.etree.ElementTree as ET
import datetime
run_time_suffix = datetime.datetime.now()
run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S")
# Set the setup variables needed by CMLBootstrap
HOST = os.getenv("CDSW_API_URL").split(
    ":")[0] + ":" + os.getenv("CDSW_DOMAIN")
USERNAME = os.getenv("CDSW_PROJECT_URL").split(
    "/")[-1] # args.username # "vdibia"

```

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

The screenshot shows the Cloudera Manager interface with the code editor open. A context menu is displayed over the code area, with the 'Run All' option highlighted by a red arrow. To the right of the editor, there's a sidebar with session information and a 'Logs' tab.

- Dependencies are starting to get downloaded.

```
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 # THE STORAGE environment variable is the Cloud Storage location used by the Datalake t
4
5 # Install the requirements
6 !pip install -r requirements.txt
7
8 # Create the directories and upload data
9
10 from cmlbootstrap import CMLBootstrap
11 from IPython.display import Javascript, HTML
12 import os
13 import time
14 import json
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("%d%b%Y%H%M%S")
21
22 # Set the setup variables needed by CMLBootstrap
23 HOST = os.getenv("COSM_API_URL").split(
24     "/")[-1] + os.getenv("COSM_PROJECT_NAME")
25 USERNAME = os.getenv("COSM_PROJECT_URL").split(
26     "/")[-1] # args.username # "vdbias"
27 API_KEY = os.getenv("COSM_API_KEY")
28 PROJECT_NAME = os.getenv("COSM_PROJECT")
29
30 # Instantiate API wrapper
31 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
32
33 # Set the STORAGE environment variable
34 try:
35     storage=os.environ["STORAGE"]
36 except:
37     tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
38     root = tree.getroot()
39
40     for prop in root.findall('property'):
41         if prop.root.findall('property'):
```

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.

File Edit View Navigate Run code/1\_data\_ingest.py

```

1.0 bootstrap.py
1.1 muser00_technical_churn_projV1
1.2 project-metadata.yaml
1.3 cdw-build.sh
1.4 churn_prediction.egg-info
1.5 cm_catalog.yaml
1.6 code
1.7 0_bootstrap.py
1.8 1_data_ingest.py
1.9 2_data_exploration.py
1.10 3_model_building.ipynb
1.11 4_train_models.py
1.12 5_model_serve_explainer
1.13 6_application.py
1.14 7a_ml_ops_simulation.py
1.15 7b_ml_ops_visual.py
1.16 churnexplainer.py
1.17 logs README.md
1.18 flask
1.19 images LICENSE.txt
1.20 lineage.yml
1.21 logs README.md
1.22 model_metrics.db
1.23 raw README.md
1.24 requirements.txt
1.25 setup.py
1.26 src README.md
1.27
1.28
1.29
1.30
1.31
1.32
1.33
1.34
1.35
1.36
1.37
1.38
1.39
1.40
1.41
1.42
1.43
1.44
1.45
1.46
1.47
1.48
1.49
1.50
1.51
1.52
1.53
1.54
1.55
1.56
1.57

```

Run Line(s) Enter .py', so the following code is set up session'. Run All Run All

muser00\_session\_1 Running By muser00 – Session – 2 vCPU / 4 GiB Memory – a few seconds ago Session Logs

Click on Run → Run All.

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

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

```

1.0 bootstrap.py
1.1 muser00_technical_churn_projV1
1.2 project-metadata.yaml
1.3 cdw-build.sh
1.4 churn_prediction.egg-info
1.5 cm_catalog.yaml
1.6 code
1.7 0_bootstrap.py
1.8 1_data_ingest.py
1.9 2_data_exploration.py
1.10 3_model_building.ipynb
1.11 4_train_models.py
1.12 5_model_serve_explainer
1.13 6_application.py
1.14 7a_ml_ops_simulation.py
1.15 7b_ml_ops_visual.py
1.16 churnexplainer.py
1.17 logs README.md
1.18 flask
1.19 images LICENSE.txt
1.20 lineage.yml
1.21 logs README.md
1.22 model_metrics.db
1.23 raw README.md
1.24 requirements.txt
1.25 setup.py
1.26 src README.md
1.27
1.28
1.29
1.30
1.31
1.32
1.33
1.34
1.35
1.36
1.37
1.38
1.39
1.40
1.41
1.42
1.43
1.44
1.45
1.46
1.47
1.48
1.49
1.50
1.51
1.52
1.53
1.54
1.55
1.56
1.57

```

Untitled Session Running By partner10 – Python 3 Session – 1 vCPU / 2 GiB Memory – just now Session Logs Spark UI header=True
> spark.sql("show databases").show()
Hive Session ID: b2a7e63fe-ca2a-406b-9961-6b66858c1cbe
+-----+
| database |
+-----+
| default |
| information\_schema |
| sys |
+-----+
> spark.sql("show tables in default").show()
+-----+
| database | tableName | isTemporary |
+-----+
| default | telco\_churn | false |
+-----+
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|
+-----+
| [7598-WHVED] | Female | No | No | 1.0 | No|No phone service| Yes | DSL | No | Electronic check| 29.85 | 29.85 |
| [No] | No | No | No | 34.0 | Yes | No | No | DSL | Yes | No | |
| [S575-GNVE] | Male | Yes | No | No | No | One year | No | No | Mailed check | Yes | 56.95 | 189.5 |
| [6668-QPYBK] | Male | No | No | No | No | 2.0 | Yes | No | DSL | Yes | \$3.85 | 108.15 |
| [7795-CFOCM] | Male | Yes | No | No | 45.0 | No|No phone service| Yes | DSL | Yes | No | |
+-----+

# this code is here to create the table in Hive used be the other parts of the project Line 1, Column 1 ★ 76 Lines Python Spaces 2

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

Untitled Session Running  
By partner10 — Python 3 Session — 1 vCPU / 2 GiB Memory — just now

Session Logs [Spark UI](#) x Collapse Share

AUTHENTICATED SPARK 2.4.0.7.1.0.0-714 Jobs Stages Storage Environment Executors SQL PythonSQL application UI

### Spark Jobs (?)

User: partner10  
Total Uptime: 3.6 min  
Scheduling Mode: FIFO  
Completed Jobs: 3

▶ Event Timeline

▼ Completed Jobs (3)

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 1 - 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

Running Sessions:  
muser00\_session\_1 started 84 minutes ago,

Session Name  
muser00\_session\_2

**Runtime**

Editor (i) **Editor**  Kernel (i) Python 3.7 Edition (i) Standard Version 2022.11

JupyterLab

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

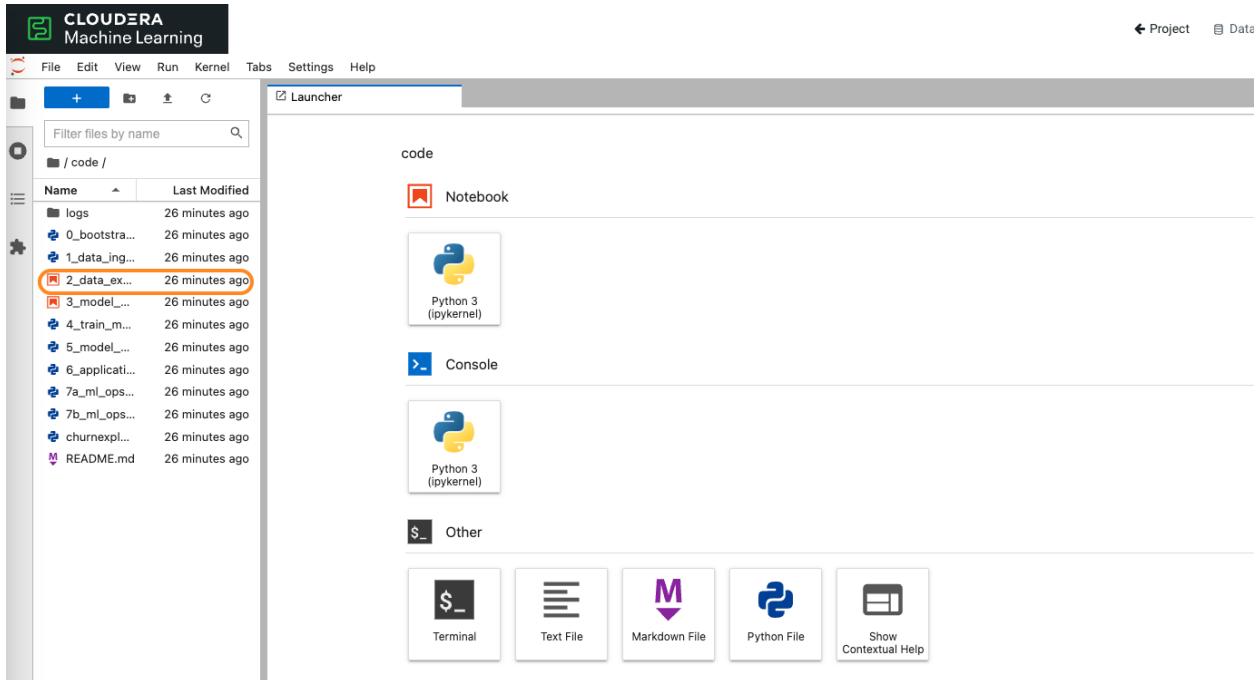
Enable Spark (i) 

**Runtime Image** - docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-jupyterlab-python3.7-standard:2022.11.1-b2

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 cell (In [1]) contains PySpark SQL code to load data from a database:

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

The next cell (In [2]) runs a query and displays the resulting DataFrame:

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

The output cell (Out[2]) shows the first 10 rows of the DataFrame:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	... DeviceProtection	Tech...
0	7590-VHVEG	Female	0	Yes	No	1.0	No	No phone service	DSL	No	...	No
1	5575-GNVDE	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	7705-CFOCIW	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]:

```
File Edit View Run Kernel Tabs Settings Help
Run Selected Cells ⌘ Enter
Run Selected Cells and Insert Below ⌘ Enter
Run Selected Cells and Don't Advance ⌘ Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
File / Project Data Terminal Access Logs Stop Sessions
2_data_exploration.ipynb
Exploration
This notebook does some basic data exploration of the telco churn data. This is just to get a sense of what the data looks like and if there are any specific patterns that can be discerned.
The data is imported into the default.telco_churn table in hive. All new data accesses will go via hive.
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().head()
```

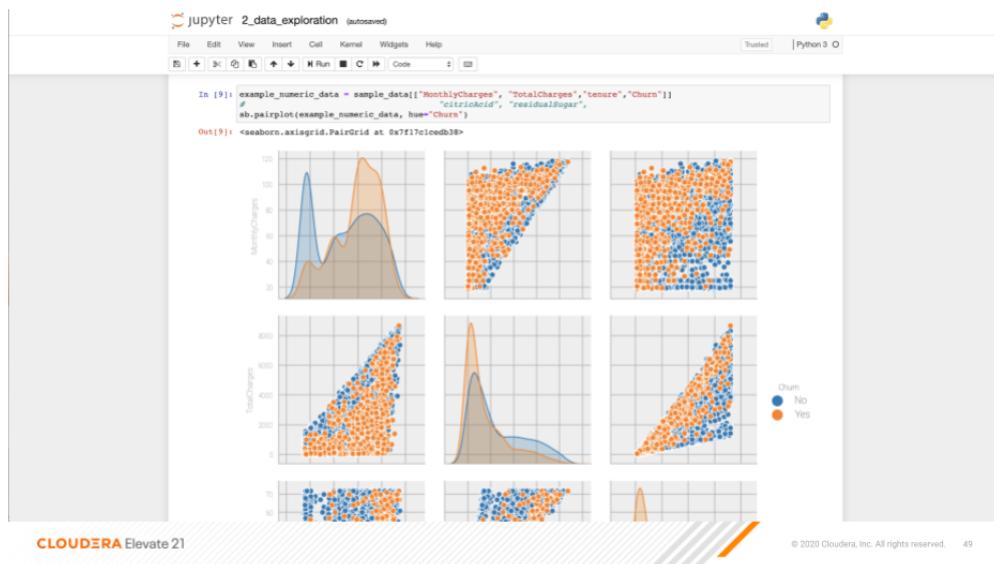
customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	TechSupport	StreamingTV	StreamingMovies
0	VHVG-	Female	0	Yes	No	1.0	No	No phone service	DSL	No	...	No	No
1	SZTC-	Male	0	No	No	34.0	Yes	No	DSL	Yes	...	Yes	No
2	3668-QPYBK	Male	0	No	No	2.0	Yes	No	DSL	Yes	...	No	No
3	7795-CFOCW	Male	0	No	No	45.0	No	No phone service	DSL	Yes	...	Yes	No
4	9337-HQHTU	Female	0	No	No	2.0	Yes	No	Fiber optic	No	...	No	No

5 rows x 21 columns

Some Basic Spark DataFrame Operations

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

Run Name	Start Time	Duration	User	Source	Version	Models	Parameters	Metrics	Tags
72ll-v03...	2023-03-07 04:28:0	5.1s	ankitshar...	ada9b5	sklearn	linear	1	Pipeline_sco: 0.58873..., Pipeline_sco: 0.60599..., test_score: 0.59	Pipeline, sklearn.pi...
wmmw-8...	2023-03-07 04:28:1	3.2s	ankitshar...	ada9b5	sklearn	linear	10	Pipeline_sco: 0.47212..., Pipeline_sco: 0.47061..., test_score: 0.47	Pipeline, sklearn.pi...
6n5u-58...	2023-03-07 04:28:1	3.8s	ankitshar...	ada9b5	sklearn	linear	100	Pipeline_sco: 0.65529..., Pipeline_sco: 0.65529..., test_score: 0.66	Pipeline, sklearn.pi...
483v-gh...	2023-03-07 04:28:1	7.2s	ankitshar...	ada9b5	sklearn	linear	1000	Pipeline_sco: 0.63594..., Pipeline_sco: 0.64277..., test_score: 0.64	Pipeline, sklearn.pi...
7oz3-ocp...	2023-03-07 04:28:2	13.9s	ankitshar...	ada9b5	sklearn	linear	10000	Pipeline_sco: 0.79180..., Pipeline_sco: 0.80204..., test_score: 0.79	Pipeline, sklearn.pi...
rw8l-wq0...	2023-03-07 04:28:4	3.1s	ankitshar...	ada9b5	sklearn	rbf	1	Pipeline_sco: 0.68828..., Pipeline_sco: 0.68638..., test_score: 0.69	Pipeline, sklearn.pi...
lb0y-irjt...	2023-03-07 04:28:4	3.2s	ankitshar...	ada9b5	sklearn	rbf	10	Pipeline_sco: 0.67463..., Pipeline_sco: 0.66780..., test_score: 0.67	Pipeline, sklearn.pi...
t41v-ygv...	2023-03-07 04:28:4	3.9s	ankitshar...	ada9b5	sklearn	rbf	100	Pipeline_sco: 0.74288..., Pipeline_sco: 0.73986..., test_score: 0.74	Pipeline, sklearn.pi...
wgvt-dvz...	2023-03-07 04:28:5	11.0s	ankitshar...	ada9b5	sklearn	rbf	1000	Pipeline_sco: 0.77531..., Pipeline_sco: 0.81494..., test_score: 0.78	Pipeline, sklearn.pi...
lbgy-yolc...	2023-03-07 04:29:0	13.5s	ankitshar...	ada9b5	sklearn	rbf	10000	Pipeline_sco: 0.79180..., Pipeline_sco: 0.82233..., test_score: 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

Runs (10)

	Status	Start Time	Run Name	Duration	User	Source	Version	Model	Kernel	Max_iter	ct	Pipeline_sco	Pipeline_sco	test_score	estimator_na	estimator_ci
<input checked="" type="checkbox"/>	✓	2023-03-07 04:28:0	72ll-w03...	5.1s	ankitshar...	sklearn	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...	sklearn	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-58...	3.8s	ankitshar...	sklearn	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...	sklearn	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...	sklearn	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...	sklearn	ada9b5	sklearn	rbf	1	ColumnTr...	0.68828...	0.68638...	0.69	Pipeline	sklearn.pi...
<input type="checkbox"/>	✓	2023-03-07 04:28:4	l8y0-lrj-t...	3.2s	ankitshar...	sklearn	ada9b5	sklearn	rbf	10	ColumnTr...	0.67463...	0.66780...	0.67	Pipeline	sklearn.pi...
<input type="checkbox"/>	✓	2023-03-07 04:28:4	t41v-ygv...	3.9s	ankitshar...	sklearn	ada9b5	sklearn	rbf	100	ColumnTr...	0.74288...	0.73966...	0.74	Pipeline	sklearn.pi...
<input type="checkbox"/>	...	...	...	...	ankitshar...	sklearn	ada9b5	sklearn	rbf	1000	ColumnTr...	0.77521...	0.71484...	0.79	Pipeline	sklearn.pi...

← Back to all runs

**Run Comparison (5)**

Run ID	72ll-v03w-rqlz-0kk3	wmmw-8euz-jofd-dpog	6n5u-58j3-jwq1-j929	483v-ghnh-kh6c-lrcs	7oz3-ocp9-std4-npwa
Run Name	72ll-v03w-rqlz-0kk3	wmmw-8euz-jofd-dpog	6n5u-58j3-jwq1-j929	483v-ghnh-kh6c-lrcs	7oz3-ocp9-std4-npwa
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
<b>Parameters</b>					
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=[0], handle_unknown='error', n_values=None, sparse=True))])	ColumnTransformer(n_jobs=None, remainder='passthrough', sparse_threshold=0.3, transformer_weights=None, transformers=[('ohe', OneHotEncoder(categorical_features=[0], handle_unknown='error', n_values=None, sparse=True))])	ColumnTransformer(n_jobs=None, remainder='passthrough', sparse_threshold=0.3, transformer_weights=None, transformers=[('ohe', OneHotEncoder(categorical_features=[0], handle_unknown='error', n_values=None, sparse=True))])	ColumnTransformer(n_jobs=None, remainder='passthrough', sparse_threshold=0.3, transformer_weights=None, transformers=[('ohe', OneHotEncoder(categorical_features=[0], handle_unknown='error', n_values=None, sparse=True))])	ColumnTransformer(n_jobs=None, remainder='passthrough', sparse_threshold=0.3, transformer_weights=None, transformers=[('ohe', OneHotEncoder(categorical_features=[0], handle_unknown='error', n_values=None, sparse=True))])
ct_n_jobs	None	None	None	None	None
ct_ohe	OneHotEncoder(categorical_features=[0], categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features=[0], categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features=[0], categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features=[0], categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)	OneHotEncoder(categorical_features=[0], categories='auto', drop=None, dtype=<class 'numpy.float64'>, handle_unknown='error', n_values=None, sparse=True)

## Lab 5 - Model Deploy/Serve

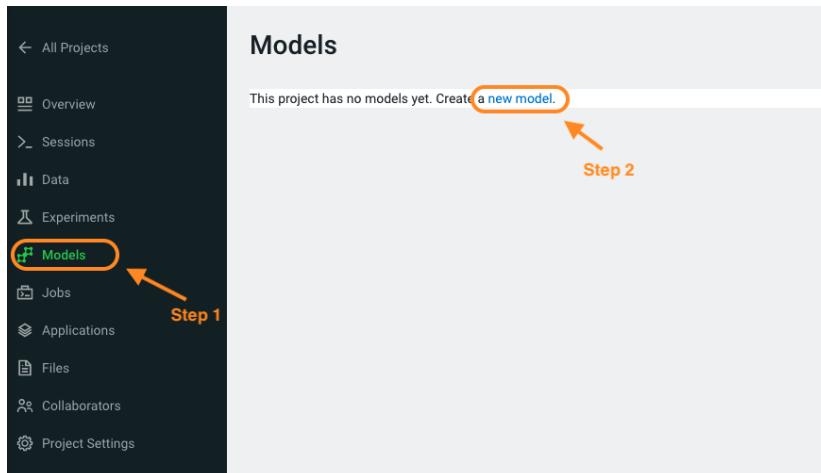
Once a model is trained its predictions and insights must be put to use so they can add value to the organization. Generally this means using the model on new, unseen data in a production environment that offers key ML Ops capabilities.

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\_server\_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

i 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
p5i-5vkq.cloud-era.site/model -d '{"accessKey": "m70ss2gubs77eitce6wqo7hrd1z22r9o", "request": {"Str
reamingTV": "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.cloud-era.site/model?accessKey=m70ss2gubs77eitce6wqo7hrd1z22r9o -d '{"request": {"Str
reamingTV": "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  
Sample Code

Churn Model API Endpoint

```
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  
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  
p5i-5vkq.cloudera.site/model?accessKey=m70ss2gubs77eitce6wqo7hrd1z22r9o -d '{"request": {"Str  
eamTV": "No", "MonthlyCharges": 70.35, "PhoneService": "No", "PaperlessBilling": "No", "Partner": "No", "On  
lineBackup": "No", "gender": "Female", "Contract": "Month-to-month", "TotalCharges": 1397.475, "Stre  
amingMovies": "No", "DeviceProtection": "No", "PaymentMethod": "Bank transfer (automatic)", "tenure": 29, "D  
ependents": "No", "OnlineSecurity": "No", "MultipleLines": "No", "InternetService": "DSL", "SeniorCitiz  
e": "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 'Files' tab selected. The left sidebar includes links for All Projects, Overview, Sessions, Data, Experiments, Models, Jobs, Applications, and Files. The 'Files' link is highlighted with a green circle. The main area displays a list of files under the 'flask' project, with 'single\_view.html' being the selected file, indicated by a red circle around its row in the table.

Name	Size	Last Modified	Action
ajax-loader.gif	2.82 kB	3 hours ago	Edit
churn_vis.css	5.12 kB	3 hours ago	Edit
churn_vis.js	2.23 kB	3 hours ago	Edit
env_vars.png	23.48 kB	3 hours ago	Edit
<b>single_view.html</b>	9.07 kB	3 hours ago	<b>Edit</b>
table_view.html	6.40 kB	3 hours ago	Edit

## Application Deployment

Go to *Applications* and create a new Application.

The screenshot shows the Cloudera Manager interface with the 'Applications' tab selected. The left sidebar includes links for All Projects, Overview, Sessions, Data, Experiments, Models, Jobs, Applications, and Files. The 'Applications' link is highlighted with a green circle. The main area displays a message: 'You currently don't have any applications'. Below this message is a brief description of what Applications are used for, followed by a 'New Application' button.

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

**General**

Name  
Customer Churn Explainer

Subdomain \*  
churn-muser00

Description  
Description

Script \*  
code/6\_application.py

---

**Runtime**

Editor ⓘ Kernel ⓘ  
Workbench Python 3.7

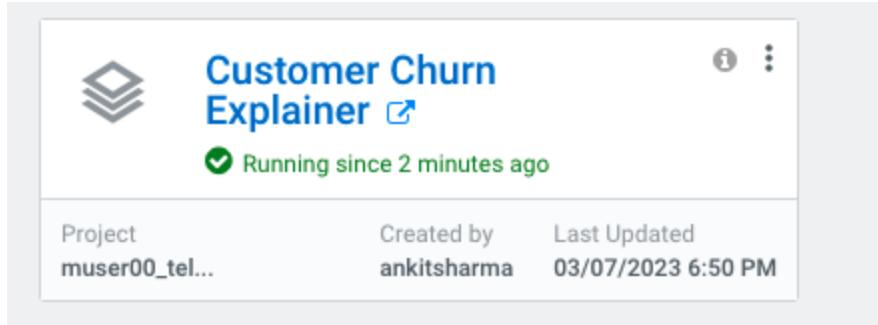
Edition ⓘ Version  
Standard 2022.11

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

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

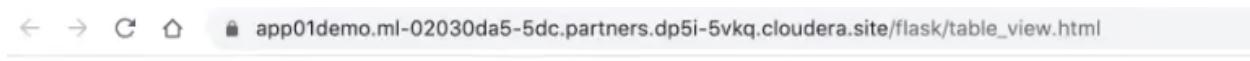
- Click on Create Application

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



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																								
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	No	Yes	Yes	Month	Yes	Elect	100.6	2415.				
2410	0.407	Femal	Yes	Yes	40	Yes	Yes	Fiber	No	No	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	No	Yes	Month	No	Elect	60.85	111.4				
3802	0.149	Femal	No	Yes	Yes	3	Yes	No	No	No in	No in	No in	No in	No in	No in	Month	No	Maile	19.3	54.7				
2408	0.148	Male	No	Yes	Yes	7	No	No ph	DSL	Yes	No	Yes	Yes	No	Yes	Two y	No	Credi	49.65	305.5				
2020	0.147	Femal	No	Yes	Yes	72	Yes	Yes	Fiber	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	Maile	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	No	No	One y	Yes	Elect	61.1	3357.				

Click on one of the item in the Probability column

[http://51-5vkq.cloudera.site/flask/table\\_v](http://51-5vkq.cloudera.site/flask/table_v)

id	Probability	gender	SeniorCitizen
1846	0.17	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.

# Demo - Applied ML Prototypes (AMPs)

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 displays the CML interface with a dark theme. On the left, a sidebar lists navigation options: Projects, Sessions, Experiments, Models, Jobs, Applications, AMPs (highlighted in green), Runtime Catalog, Site Administration, and Learning Hub. The main content area is titled "Applied ML Prototypes". It features a search bar and filters for "Source" and "Tags". A grid of nine prototypes is shown:

- Churn Modeling with scikit-learn**: A dashboard showing a confusion matrix and a ROC curve with 78.3% accuracy.
- Deep Learning for Image Analysis**: Categories: COMPUTER VISION, IMAGE ANALYSIS.
- Deep Learning for Anomaly Detection**: Categories: ANOMALY DETECTION, TENSORFLOW.
- Question Answering with Wikipedia**: A dashboard showing a search interface and results for "What is the capital city of United Kingdom?".
- WIKIPEDIA The Free Encyclopedia**: A link to the Wikipedia homepage.
- Deep Learning for Question Answering**: Categories: AUTOMATED QUESTION ANSWERING, EXTRACTIVE QUES.
- Explaining Models with LIME and SHAP**: A dashboard showing feature importance plots for a model.
- Active Learning**: Categories: ACTIVE LEARNING, LEARNING WITH LIMITED LABELED DATA.
- Stream**: A preview of a real-time data processing application.

Configure the Project.

## Churn Modeling with scikit-learn

X

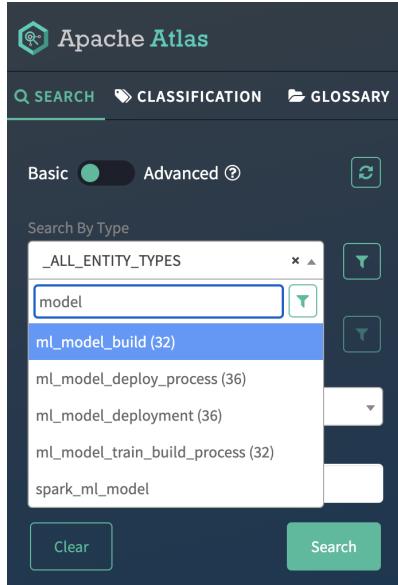
This project demonstrates how to build a logistic regression classification model to predict the probability that a group of customers will churn from a fictitious telecommunications company. In addition, the model is interpreted using a technique called Local Interpretable Model-agnostic Explanations (LIME). Both the logistic regression and LIME models are deployed using CML's real-time model deployment capability and interact with a basic Flask-based web application.

[CHURN PREDICTION](#)[LOGISTIC REGRESSION](#)[EXPLAINABILITY](#)[LIME](#)[View on Github](#)[Cancel](#)[Configure Project](#)

## Demo- 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.

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.



In the output, you will see all models that your colleagues deployed in this workshop. Notice that each model is assigned a unique ID at the end. That ID corresponds to the Model Build from CML. Identify your model using the Build Id noted down when you deployed your model. select the model you created.

The screenshot shows the Apache Atlas search results page for entities of type 'ml\_model\_build'. The search bar at the top has 'ml\_model\_build (32)' entered. The results table below shows three entries:

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.

**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	<pre>{   parent_name: "Churn Modeling with scikit-learn - p...   and fusco".</pre>

**User-defined property**

**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

