

CLOUDERA

MACHINE LEARNING HANDS-ON

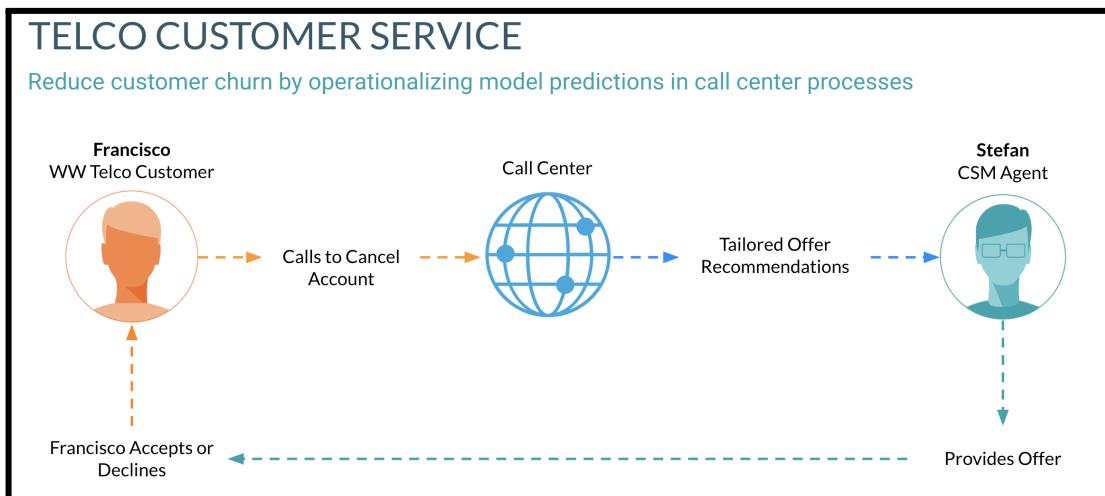
STUDENT GUIDE

Table of Contents

Introduction	2
Lab summary	3
Lab 0 - Initial setup	4
Pre-requisites	4
Getting Connected	4
Create Project	4
Lab 1 - Data Ingest	8
Bootstrap Script	9
Step 1 : Create New Session	9
Step 2 : Execute Bootstrap script	11
Execute Data Ingest Script	13
Lab 2 - Data Exploration	17
Step 1 : Create New Session	17
Step 2 : Execute Data Exploration script	19
Lab 3 - Model Build/Train Experiment	23
Model Building	23
Step 1 : Create Session	23
Step 2 : Execute model building script	25
Model Training	26
Step 1 : Create a session	26
Step 2 : Run Experiments	29
Step 3 : Verify Experiment metrics	31
Lab 4 - Model Deploy/Serve	33
Lab 5 - Application Deploy	42

Introduction

In this workshop, we will introduce you to the World Wide Telco organization. They have many ML use cases, but we'll focus on a specific one for today. In this use case they are trying to reduce customer churn (i.e. cancellations) and most of the cancellations are coming in from their call center. In these hands-on labs, a customer, "Francisco" will call in with the intent to cancel. The call center application will - based on the predicted reasoning for Francisco to cancel - provide recommendations to Stefan who is the customer service manager for WW Telco. He'll then provide the offer to Francisco who will choose or not to choose to accept - i.e. not cancel his account.



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.

Lab summary

1. Data Ingest.
2. Data Exploration.
3. Model Build Experiment.
4. Model Deploy.
5. Application Deploy

Lab 0 - Initial setup

Pre-requisites

1. Laptop with a supported OS (Windows 7 not supported) or Macbook.
2. A modern browser - Google Chrome (IE, Firefox, Safari not supported).
3. Firewall that prevents access to GitHub should not be present.

Getting Connected

You will be provided with a set of credentials along with a URL which will give you access to the CDP environment. The details will be shared by the instructor either before the lab starts or during the course of the labs

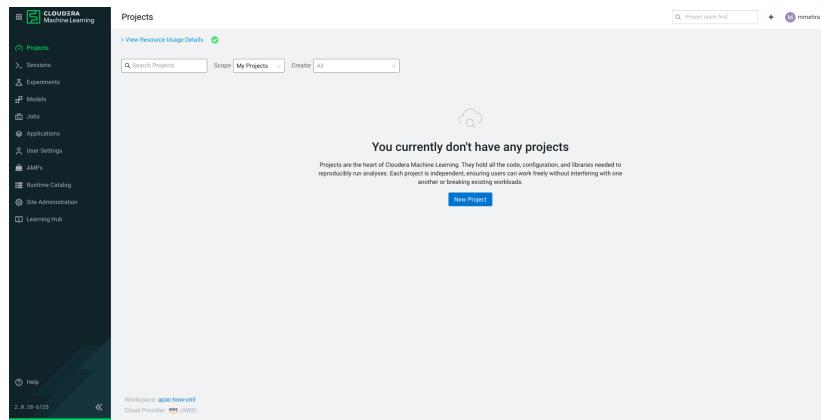
Environment URL	Will be shared by the instructor
Login Email	Will be shared by the instructor
Workload User Name	Will be shared by the instructor
Workload Password	Will be shared by the instructor
Github URL	Will be shared by the instructor

Create 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.

Status	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	workshop-workspace	ptp-demo-env01	ap-south-1	04/13/2022 5:58 PM IST	AWS	⋮

This will launch a ML workspace screen.



Click on “New Project” to start with the creation of our project.

Enter the following details in the New Project Page

Project Name	<Workload_Username>_telco_churn_project
Project Description	Telco churn analytics
Project Visibility	Private
Initial Setup	Select “Git”
Protocol	HTTPS
Git URL of Project	https://github.com/mmehra12/cml_churn_demo
Runtime Setup	Basic
Kernel	Python 3.7

* Project Name

Project Description

Project Visibility

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

Initial Setup

[Blank](#) [Template](#) [AMPs](#) [Local Files](#) [Git](#)

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

HTTPS SSH

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

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 - 2023.05
- PBJ Workbench - Python 3.7 - Standard - 2023.05
- Workbench - Python 3.7 - Standard - 2023.05

Click on **Create Project**

On successful creation you should now see the project on your Project page

Projects

The screenshot shows the CloudBees Machine Learning Project page. At the top, there is a button labeled "View Resource Usage Details" with a green checkmark icon. Below it is a search bar with placeholder text "Search Projects". To the right of the search bar are dropdown menus for "Scope" set to "My Projects" and "Creator" set to "All". A list of projects is displayed, with the first project being "apac100_telco_churn_...". This project has a lock icon, a gear icon, and a plus sign icon. Below the project name, it says "Owned by mmehra" and "Last worked on a minute ago".

Clicking on it will take you to the Project that you just cloned from GitHub and you will be able to manage all the files from GitHub here.

The screenshot shows the details of the "mmehra-telco-churn-projV1" project. On the left, a sidebar menu includes "All Projects", "Overview" (which is selected), "Sessions", "Experiments", "Models", "Jobs", "Applications", "Files", "Collaborators", and "Project Settings". The main area shows the project name "mmehra / mmehra-telco-churn-projV1". It has sections for "Models" (no models yet) and "Jobs" (no jobs yet). The "Files" section displays a file tree and a table of contents. The file tree shows a directory structure with "flask", "images", "models", "raw", and several Python files like "0_bootstrap.py", "1_data_ingest.py", "2_data_exploration.ipynb", "3_model_building.ipynb", "4_train_models.py", "5_model_server_explainer.py", "6_application.py", "7_build_project.py", "cdw-build.sh", "chumexplainer.py", "README.md", and "requirements.txt". The table of contents lists these files with their sizes and last modified times:

Name	Size	Last Modified
flask	-	a minute ago
images	-	a minute ago
models	-	a minute ago
raw	-	a minute ago
0_bootstrap.py	2.08 kB	a minute ago
1_data_ingest.py	2.99 kB	a minute ago
2_data_exploration.ipynb	599.00 kB	a minute ago
3_model_building.ipynb	76.78 kB	a minute ago
4_train_models.py	5.00 kB	a minute ago
5_model_server_explainer.py	1.27 kB	a minute ago
6_application.py	2.62 kB	a minute ago
7_build_project.py	7.27 kB	a minute ago
cdw-build.sh	44 B	a minute ago
chumexplainer.py	6.69 kB	a minute ago
README.md	10.02 kB	a minute ago
requirements.txt	175 B	a minute ago

Lab 1 - Data Ingest

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

STAGES

Data Ingest

Data Exploration

Model Build

Model Deploy

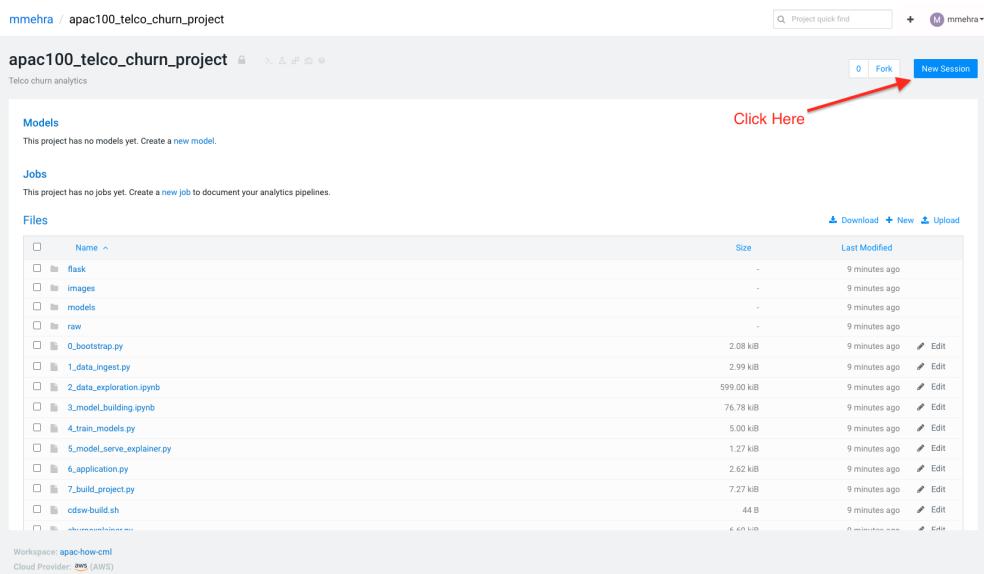
Application Deploy

Bootstrap Script

We need to execute a bootstrap script at the start of the project. It will install the requirements, create the STORAGE environment variable and copy the data from raw/WA_Fn-UseC_-Telco-Customer-Churn-.csv into /datalake/data/churn of the STORAGE location, on AWS it will s3a://[something], on Azure it will be abfs://[something] and on CDSW cluster, it will be hdfs://[something]

Step 1 : Create New Session

To create a new session you can go into your project and click on **New Session**



The screenshot shows the Apache Spark Project UI for the 'apac100_telco_churn_project'. At the top right, there are two buttons: 'Fork' (gray) and 'New Session' (blue). A red arrow points to the 'New Session' button. Below the buttons is a section labeled 'Click Here'. The main area contains sections for 'Models', 'Jobs', and 'Files'. The 'Files' section lists several Python files and a shell script:

Name	Size	Last Modified
0.bootstrap.py	2.08 kB	9 minutes ago
1.data_ingest.py	2.99 kB	9 minutes ago
2.data_exploration.ipynb	599.00 kB	9 minutes ago
3.model_building.ipynb	76.78 kB	9 minutes ago
4.train_models.py	5.00 kB	9 minutes ago
5.model_serve_explainer.py	1.27 kB	9 minutes ago
6.application.py	2.62 kB	9 minutes ago
7.build_project.py	7.27 kB	9 minutes ago
cdsw-build.sh	44 B	9 minutes ago
abm_analytics.m	6.60 kB	9 minutes ago

At the bottom left, it says 'Workspace: apac-how-cml' and 'Cloud Provider: aws (AWS)'.

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

Session Name	prep_data_ingest
Runtime Editor	Workbench
Enable Spark	Yes - Spark version 2.4.8
Resource Profile	2 vCPU / 4 GiB

Click on **START SESSION**

Start A New Session

Session Name

Runtime

Editor <input type="radio"/>	Kernel <input type="radio"/>	Edition <input type="radio"/>	Version
Workbench <input type="button" value="▼"/>	Python 3.7	Standard	2023.05

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

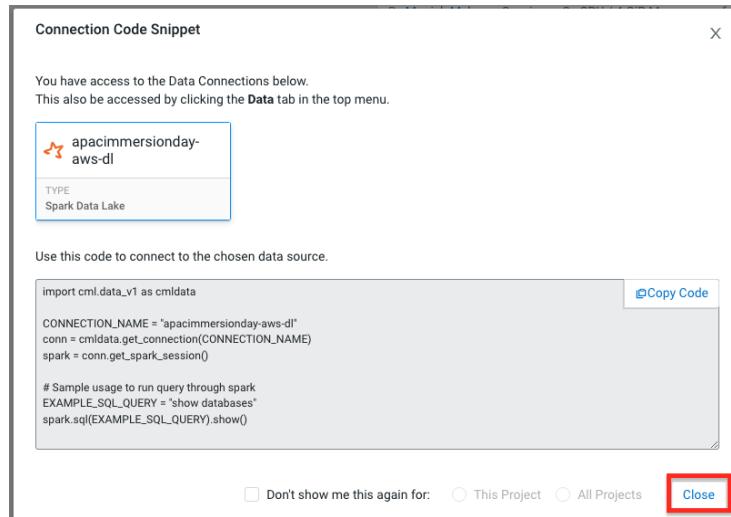
Enable Spark

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2023.05.2-b7

Resource Profile

IMPORTANT : Please do not use the higher resource configurations.

On successful creation of the session you will get a Dialog box with a code snippet to connect to this session from an application. For now we can click on Close

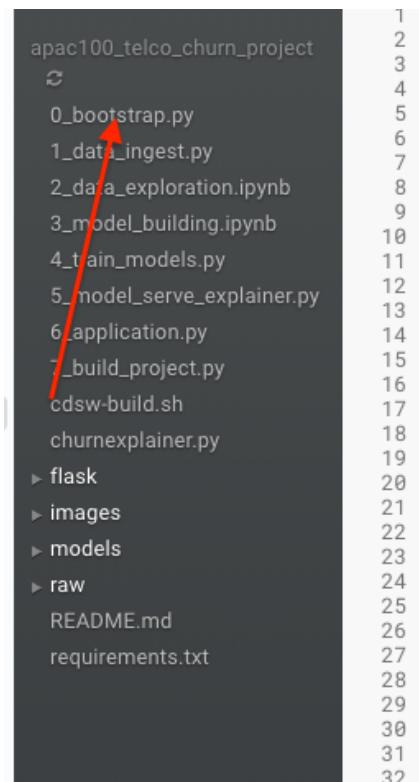


Step 2 : Execute Bootstrap script

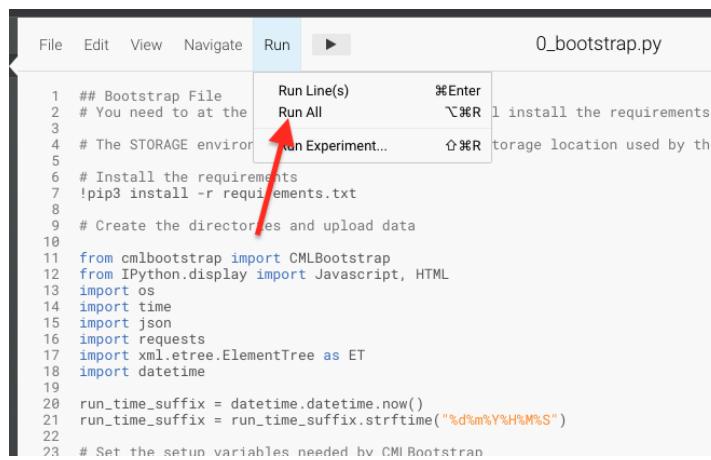
Once the session is ready you should get a similar message

The screenshot shows a Jupyter Notebook interface. At the top, it says "data-ingest_se01" with a "Running" status indicator. Below that, it says "By Manick Mehra – Session – 1 vCPU / 2 GiB Memory – a few seconds ago". There are tabs for "Session" and "Logs", with "Session" being the active tab. At the top right are buttons for "Collapse", "Share", and "Export PDF". A yellow box highlights the text "Use ?command to get help on a particular command.".

Select the `0_bootstrap.py` on the left file browser



Select Run -> Run All



```
File Edit View Navigate Run ► 0_bootstrap.py
1 ## Bootstrap File
2 # You need to at the
3
4 # The STORAGE environ
5
6 # Install the requirements
7 !pip3 install -r requirements.txt
8
9 # Create the directories and upload data
10
11 from cmlbootstrap import CMLBootstrap
12 from IPython.display import Javascript, HTML
13 import os
14 import time
15 import json
16 import requests
17 import xml.etree.ElementTree as ET
18 import datetime
19
20 run_time_suffix = datetime.datetime.now()
21 run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S")
22
23 # Set the setup variables needed by CML Bootstrap
```

As this will install all the dependencies and the first execution will take a bit tad longer as it needs to download all the binaries. You will start to see the execution logs on the right side of the screen.

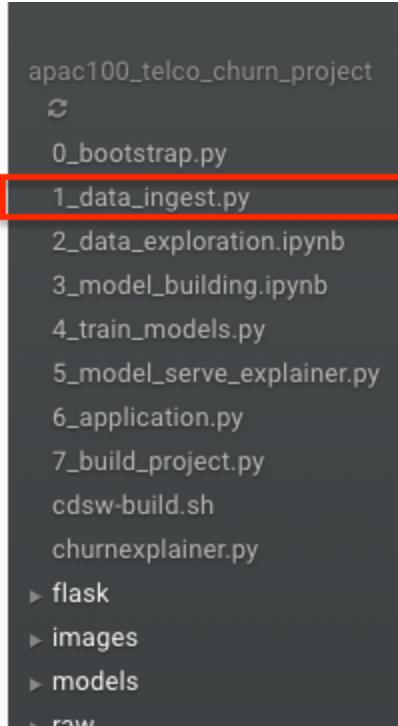
```
Collecting dill==0.3.1.1 (from -r requirements.txt (line 2))
  Downloading dill-0.3.1.1.tar.gz (151 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 152.0/152.0 kB 2.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting lime==0.1.1.36 (from -r requirements.txt (line 3))
  Downloading lime-0.1.1.36.tar.gz (275 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 275.9/275.9 kB 4.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting scikit-learn==0.21.3 (from -r requirements.txt (line 4))
  Downloading scikit_learn-0.21.3-cp37-cp37m-manylinux1_x86_64.whl (6.7 MB)
    ━━━━━━━━━━━━━━━━ 6.7/6.7 MB 15.8 MB/s eta 0:00:00
Collecting xlrd==1.2.0 (from -r requirements.txt (line 5))
  Downloading xlrd-1.2.0-py2.py3-none-any.whl (103 kB)
    ━━━━━━━━━━━━━━ 103.3/103.3 kB 2.6 MB/s eta 0:00:00
Collecting pandas==0.25.1 (from -r requirements.txt (line 6))
  Downloading pandas-0.25.1-cp37-cp37m-manylinux1_x86_64.whl (10.4 MB)
    ━━━━━━━━ 10.4/10.4 MB 45.3 MB/s eta 0:00:00
Collecting numpy==1.17.2 (from -r requirements.txt (line 7))
  Downloading numpy-1.17.2-cp37-cp37m-manylinux1_x86_64.whl (20.3 MB)
    ━━━━━━ 20.3/20.3 MB 40.5 MB/s eta 0:00:00
Collecting flask==2.1.0 (from -r requirements.txt (line 8))
  Downloading Flask-2.1.0-py3-none-any.whl (95 kB)
    ━━━━ 95.2/95.2 kB 2.1 MB/s eta 0:00:00
```

This execution will take a couple of minutes. The last command to be executed is this and post this the bootstrap step is completed, and you can move to the next step.

```
!hdfs dfs -copyFromLocal /home/cdsw/raw/WA_Fn-UseC_-Telco-Customer-Churn-.csv $STORAGE/datalake/data/churn/WA_
22/04/19 10:35:57 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-syste
m.properties,hadoop-metrics2.properties
22/04/19 10:35:57 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/04/19 10:35:57 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/04/19 10:35:57 INFO s3a.IDBDelegationTokenBinding: Maybe renewing Knox Token when deploying unbonded token
22/04/19 10:35:57 INFO s3a.IDBDelegationTokenBinding: There is no Knox Token available, fetching one from IDB
roker...
22/04/19 10:35:57 INFO idbroker.AbstractIDBCClient: Authenticating with IDBroker requires Kerberos
22/04/19 10:35:57 INFO idbroker.AbstractIDBCClient: Kerberos credentials are available, using Kerberos to esta
blish a session. UGI=mmehra/7e588bef-8ef4-45dd-81d3-084ef2499ba1.mlx.cloudera.site@PTP-DEMO.DP5I-5VKQ.CLOUDER
A.SITE (auth:KERBEROS)
Apr 19, 2022 10:35:57 AM org.apache.knox.gateway.shell.KnoxSession createClient
INFO: Using default JAAS configuration
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Bonded to Knox token eyJqa3...SSlPkw
22/04/19 10:35:58 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming S
ECONDS
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Maybe renewing Knox Token when fetching AWS credentials
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Using existing Knox Token: eyJqa3...SSlPkw
22/04/19 10:35:58 INFO idbroker.AbstractIDBCClient: Creating Knox CAB session using Knox DT eyJqa3...SSlPkw
...
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Maybe renewing Knox Token when fetching AWS credentials
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Using existing Knox Token: eyJqa3...SSlPkw
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Maybe renewing Knox Token when fetching AWS credentials
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Using existing Knox Token: eyJqa3...SSlPkw
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Maybe renewing Knox Token when fetching AWS credentials
22/04/19 10:35:58 INFO s3a.IDBDelegationTokenBinding: Using existing Knox Token: eyJqa3...SSlPkw
copyFromLocal: `s3a://ot-ptp-demo/datalake/data/churn/WA_Fn-UseC_-Telco-Customer-Churn-.csv': File exists
22/04/19 10:35:59 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/04/19 10:35:59 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/04/19 10:35:59 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
```

Execute Data Ingest Script

In the same Workbench, open the script “*1_data_ingest.py*”



This script 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.

Click on Run → Run All.

```

File Edit View Navigate Run ►
1_data_ingest.py

1 ## Data Ingest
2 # This script grabs the data from an S3 bucket and writes it into a Spark DataFrame
3 # storage location set in step 0 into a Spark DataFrame
4
5 import os
6 import sys
7 from pyspark.sql import SparkSession
8 from pyspark.sql.types import *
9 spark = SparkSession(
10     .builder\
11     .appName("PythonSQL")\
12     .master("local[*]")\
13     .getOrCreate()
14
15 # Add the following config if you want to run on the k8s cluster and remove 'local[*]'
16 #     .config("spark.yarn.access.hadoopFileSystems","s3a://demo-aws-2//")
17
18 # Since we know the data already, we can add schema upfront. This is good practice as Spark
19 # the schema.
20
21 schema = StructType([
22     [
23         StructField("customerID", StringType(), True),
24         StructField("gender", StringType(), True),
25         StructField("SeniorCitizen", StringType(), True),
26         StructField("Partner", StringType(), True),
27         StructField("Dependents", StringType(), True),
28         StructField("tenure", DoubleType(), True),
29         StructField("PhoneService", StringType(), True),
30         StructField("MultipleLines", StringType(), True),

```

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

```

File Edit View Navigate Run Project Terminal access Clear Interrupt Stop Sessions
ml-02030da5-5dc.partners.dp5i-5vkq.cloudera.site/partner10/telco-churn-10/engines/r7wbvzx5yu5xjgj
1_data_ ingest.py
1 import os
2 import sys
3 from pyspark.sql import SparkSession
4 from pyspark.sql.types import *
5 spark = SparkSession
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 schema = StructType([
15     StructField("customerID", StringType(), True),
16     StructField("gender", StringType(), True),
17     StructField("partner", StringType(), True),
18     StructField("dependents", StringType(), True),
19     StructField("tenure", IntegerType(), True),
20     StructField("phoneservice", StringType(), True),
21     StructField("multiplelines", StringType(), True),
22     StructField("internetservice", StringType(), True),
23     StructField("onlinesecurity", StringType(), True),
24     StructField("onlinesupport", StringType(), True),
25     StructField("deviceprotection", StringType(), True),
26     StructField("techsupport", StringType(), True),
27     StructField("streamingtvtv", StringType(), True),
28     StructField("streamingmovies", StringType(), True),
29     StructField("contract", StringType(), True),
30     StructField("paperlessbilling", StringType(), True),
31     StructField("paymentmethod", StringType(), True),
32     StructField("monthlycharges", DoubleType(), True),
33     StructField("totalcharges", DoubleType(), True),
34     StructField("churn", StringType(), True)
35 ])
36
37 telco_data = spark.read.csv(
38     "file:///data/churn/KA_Fn-UseC--Telco-Customer-Churn-.csv".format(s3.bucket),
39     header=True,
40     schema=schema,
41     sep=",",
42     nullValue="NA"
43 )
44
45 s3.bucket = os.environ["STORAGE"]
46
47 telco_data = spark.read.csv(
48     "file:///data/churn/KA_Fn-UseC--Telco-Customer-Churn-.csv".format(s3.bucket),
49     header=True,
50     schema=schema,
51     sep=",",
52     nullValue="NA"
53 )
54 telco_data.show()
55 telco_data.printSchema()
56 telco_data.coalesce(1).write.csv(
57     "file:///home/codes/raw/telco-data/",
58     mode="overwrite",
59     header=True
60 )
61 spark.sql("show databases").show()
62 spark.sql("show tables in default").show()
63
64 # this code is here to create the table in Hive used by the other parts of the project
65
Line 1.Column 1

```

★ 76 Lines Python Spaces 2

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

Spark Jobs (?)

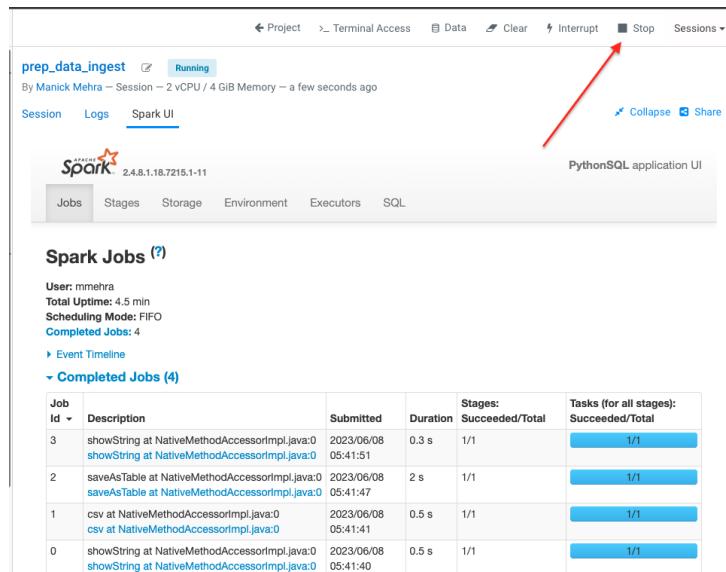
User: partner10
Total Uptime: 3.6 min
Scheduling Mode: FIFO
Completed Jobs: 3

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

Stop the session once you data ingestion completes



The screenshot shows the Databricks PythonSQL application UI. At the top, there's a navigation bar with links for Project, Terminal Access, Data, Clear, Interrupt, Stop, and Sessions. Below the navigation bar, the session details are shown: prep_data_ingest, Running, By Manick Mehra, Session - 2 vCPU / 4 GiB Memory - a few seconds ago. There are tabs for Session, Logs, and Spark UI (which is selected). The Spark UI section displays the following information:

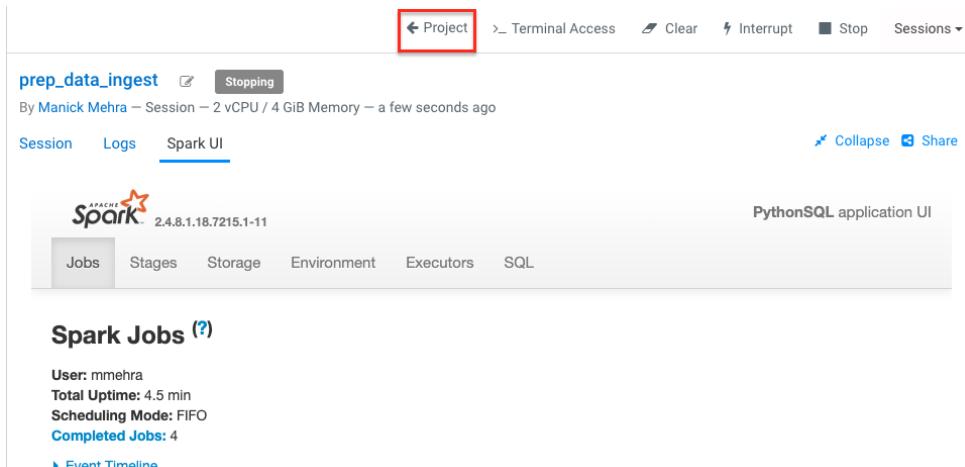
Spark Jobs (2)

User: mmehra
Total Uptime: 4.5 min
Scheduling Mode: FIFO
Completed Jobs: 4

Completed Jobs (4)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	showString at NativeMethodAccessorsImpl.java:0 showString at NativeMethodAccessorsImpl.java:0	2023/06/08 05:41:51	0.3 s	1/1	1/1
2	saveAsTable at NativeMethodAccessorsImpl.java:0 saveAsTable at NativeMethodAccessorsImpl.java:0	2023/06/08 05:41:47	2 s	1/1	1/1
1	csv at NativeMethodAccessorsImpl.java:0 csv at NativeMethodAccessorsImpl.java:0	2023/06/08 05:41:41	0.5 s	1/1	1/1
0	showString at NativeMethodAccessorsImpl.java:0 showString at NativeMethodAccessorsImpl.java:0	2023/06/08 05:41:40	0.5 s	1/1	1/1

Go back to the Project page



The screenshot shows the Databricks PythonSQL application UI. The navigation bar is identical to the previous one, with the 'Project' link highlighted by a red box. The session details and Spark Jobs section are also identical to the previous screenshot.

Lab 2 - 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 that allows you to bring your own editor.

STAGES

Data Ingest

Data Exploration

Model Build

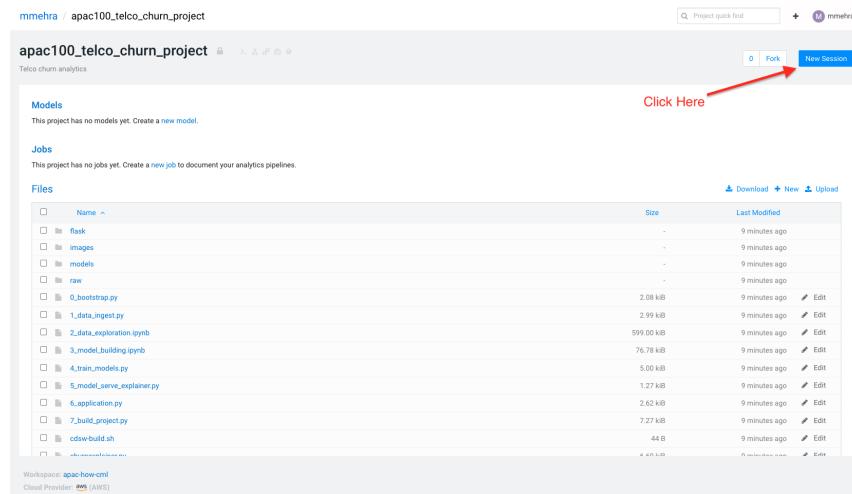
Model Deploy

Application Deploy

CLOUDERA © 2020 Cloudera, Inc. All rights reserved. 45

Step 1 : Create New Session

To create a new session you can go into your project and click on **New Session**



The screenshot shows the Cloudera Manager interface with the following details:

- Project Details:** apac100_telco_churn_project (Telco churn analytics)
- 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:** A list of files in the project directory:

Name	Size	Last Modified	Action
task	-	9 minutes ago	Edit
images	-	9 minutes ago	Edit
models	-	9 minutes ago	Edit
raw	-	9 minutes ago	Edit
0_bootstrap.py	2.08 kB	9 minutes ago	Edit
1_data_ingest.py	2.99 kB	9 minutes ago	Edit
2_data_exploration.ipynb	599.00 kB	9 minutes ago	Edit
3_model_building.py	76.78 kB	9 minutes ago	Edit
4_train_models.py	5.00 kB	9 minutes ago	Edit
5_model_serve_explainer.py	1.27 kB	9 minutes ago	Edit
6_application.py	2.62 kB	9 minutes ago	Edit
7_build_project.py	7.27 kB	9 minutes ago	Edit
cdsw-build.sh	44 B	9 minutes ago	Edit
4.ipynb	4.49 kB	9 minutes ago	Edit
- Project Actions:** Download, New, Upload
- User Information:** mmehera / apac100_telco_churn_project
- Footer:** Workspace: apac-how-cml, Cloud Provider: AWS (AWS)

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

Session Name	data_explore
Runtime Editor	JupyterLab
Enable Spark	Yes - Spark version 2.4.8
Resource Profile	2 vCPU / 4 GiB

Click on **START SESSION**

Start A New Session

Session Name

Runtime

Editor Kernel Edition
JupyterLab Python 3.7 Standard 2023.05

JupyterLab PBJ Workbench Workbench Spark 2.4.8 - CDE 1.18.1 - HOTFIX-4

Resource Profile

Step 2 : Execute Data Exploration script

Double-Click on **2_data_exploration.ipynb** it will take you into the notebook

The screenshot shows a Jupyter Notebook interface with the title "Data Exploration". The content of the cell is a Python script for loading data from a database:

```
[1]: # Load the data
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("Telco Data Set")\
    .master("local[*]") \
    .getOrCreate()
telco_data_raw = spark.sql("SELECT * FROM default.telco_churn")
telco_data_raw.toPandas().head()
```

Below the code, the output shows the first five rows of the data frame:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	... DeviceProtection	TechSupport	StreamingTV	StreamingN
0	7590-VHVEG	Female	0	Yes	No	1.0	No	No phone service	DSL	No	...	No	No	No
1	5575-GNVDE	Male	0	No	No	34.0	Yes	No	DSL	Yes	...	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2.0	Yes	No	DSL	Yes	...	No	No	No
3	7795-CFOCW	Male	0	No	No	45.0	No	No phone service	DSL	Yes	...	Yes	Yes	No
4	9237-HQITU	Female	0	No	No	2.0	Yes	No	Fiber optic	No	...	No	No	No

5 rows x 21 columns

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

The screenshot shows a Jupyter Notebook interface with the title "Telco Data Exploration". The content of the cell is a Python script for loading data from a database:

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

Below the code, the output shows the first five rows of the data frame:

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

Out[2]:

At this point the data scientist realized that they forgot to add a dependency at the time of bootstrap process. They can still do that from here. Let's see how that can be done.

For our data exploration, if you run the script without making any changes you will see that it will fail at a point because of a missing dependency.

Feature Distributions

We want to examine the distribution of our features, so start with them one at a time.

Seaborn has a standard function called `distplot()` that allows us to easily examine the distribution of a column of a pandas dataframe or a numpy array.

```
: import matplotlib.pyplot as plt
import seaborn as sns

plt.grid(b=None)
plt.title("Tenure Distribution",color='grey')
plt.xticks(color='grey')
plt.yticks(color='grey')

sns.set_style("whitegrid")
sns.despine(left=True,bottom=True)

def plotter():
    sns.distplot(sample_data['tenure'], kde=False, xlabel=False)
plotter()

-----  
ModuleNotFoundError: Traceback (most recent call last)
/tmp/ipykernel_214/1760403484.py in <module>
      1 import matplotlib.pyplot as plt
----> 2 import seaborn as sns
      3
      4 plt.grid(b=None)
      5 plt.title("Tenure Distribution",color='grey')

ModuleNotFoundError: No module named 'seaborn'
```

This can be taken care of by adding the missing dependency before we make use of it.

Add this command at the start of your script and Run All Cells again.

To add a new command block go to the start of the script and select the first block of code,
Click on the + sign at the top of the editor and enter the following command

```
!pip install seaborn
```

Data Exploration

This note book 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 deemed. In step 1, the data is imported into the `default.telco_churn` table in hive. All new data accesses will go via hive.

```
[1]: # Load the data
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("Telco Data Set")\
    .master("local[*]") \
    .getOrCreate()
telco_data_raw = spark.sql("SELECT * FROM default.telco_churn")
telco_data_raw.toPandas().head()

Hive Session ID is e0db9aef-8440-4611-a2c2-993b66c671fc
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	... DeviceProtection	TechSupport	StreamingTV	StreamingMovies
0	7590-VHVEG	Female	0	Yes	No	1.0	No	No phone service	DSL	No	...			
1	5575-GNVDE	Male	0	No	No	34.0	Yes	No	DSL	Yes	...			
2	3668-QPYBK	Male	0	No	No	2.0	Yes	No	DSL	Yes	...			
3	7795-CFOCW	Male	0	No	No	45.0	No	No phone service	DSL	Yes	...			
4	9237-HQITU	Female	0	No	No	2.0	Yes	No	Fiber optic	No	...			

5 rows x 21 columns

```
[1]: !pip install seaborn
```

Some Basic Spark DataFrame Operations

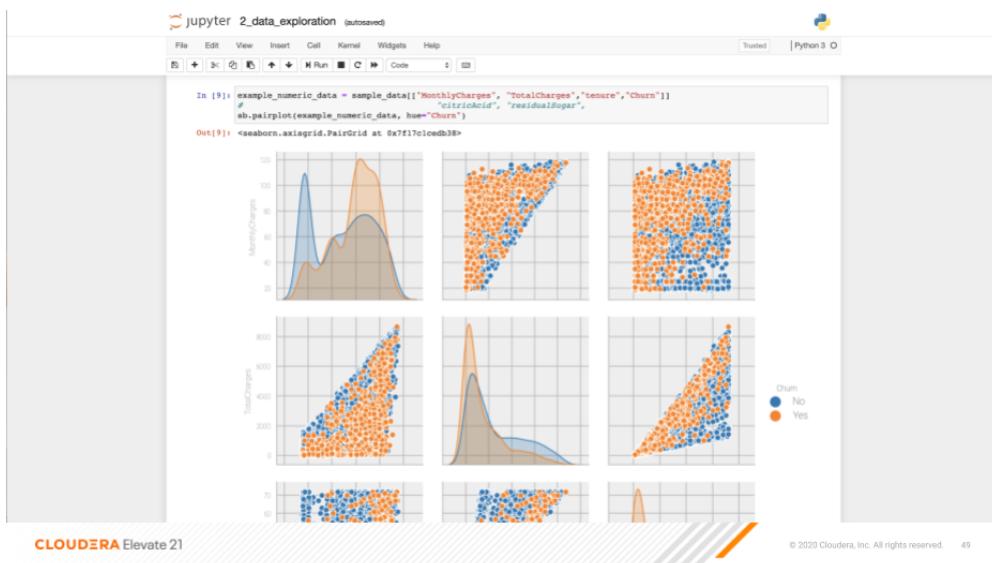
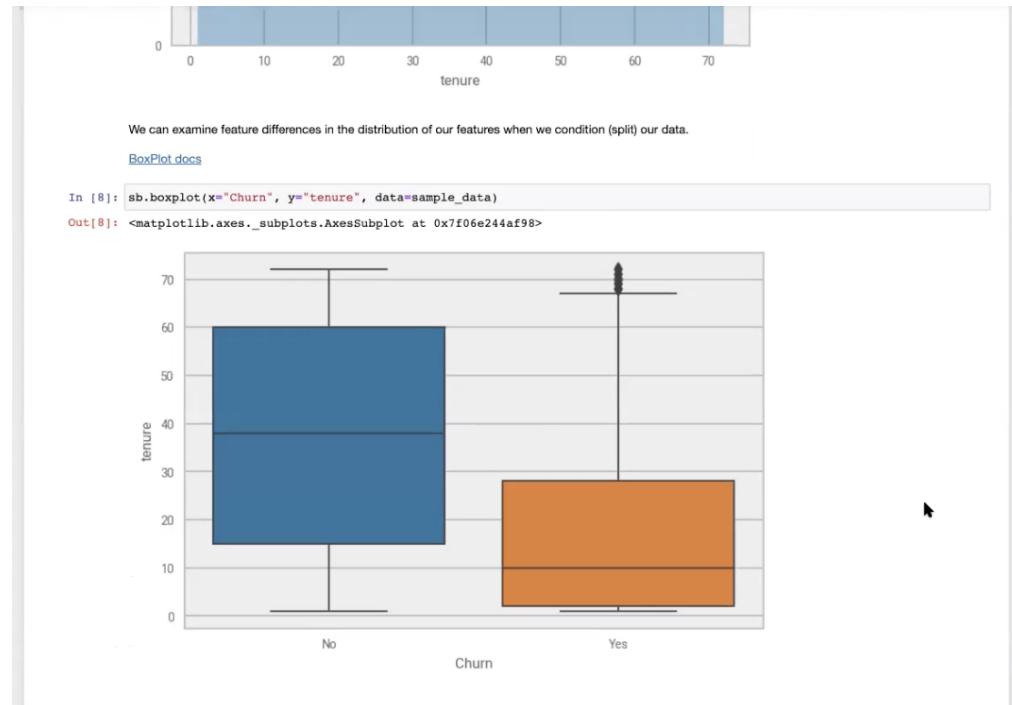
The screenshot shows a Cloudera Machine Learning notebook interface. On the left, there's a sidebar with project files: `1.data_prep.py`, `2_data_exploration.ipynb` (selected), `3.model_building.ipynb`, `4.train_models.py`, `5.model_serve_explainer.py`, `6.application.py`, `7.build_project.py`, `cdsw-build.sh`, `churnexplainer.py`, `README.md`, and `requirements.txt`. The main area has a title bar "Exploration" and a sub-section "Some Basic Spark DataFrame Operations". Below that is a code cell with the following content:

```
....,spark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("Telco Data Set")\
    .master("local[*]") \
    .getOrCreate()
telco_data_raw = spark.sql("SELECT * FROM default.telco_churn")
telco_data_raw.toPandas().head()
```

A context menu is open over the code editor, with the "Run All Cells" option highlighted. Other options in the menu include "Run Selected Cells", "Run Selected Cells and Insert Below", "Run Selected Cells and Don't Advance", "Run Selected Text or Current Line in Console", "Run All Above Selected Cell", "Run Selected Cell and All Below", "Render All Markdown Cells", and "Restart Kernel and Run All Cells...".

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 3 - Model Build/Train Experiment

In this lab, you will build and train the model, using the Experiment feature form 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.

STAGES

Data Ingest

Data Exploration

Model Build Experiment

Model Deploy

Application Deploy

CLOUDERA

© 2020 Cloudera, Inc. All rights reserved. 51

Model Building

Step 1 : Create Session

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.

To create a new session you can go into your project and click on **New Session**

mmehra / apac100_telco_churn_project

apac100_telco_churn_project

Telco churn analytics

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	-	9 minutes ago	Edit
images	-	9 minutes ago	Edit
models	-	9 minutes ago	Edit
raw	-	9 minutes ago	Edit
scripts	-	9 minutes ago	Edit
0_bootstrap.py	2.08 kB	9 minutes ago	Edit
1_data_ingest.py	2.99 kB	9 minutes ago	Edit
2_data_exploration.ipynb	599.00 kB	9 minutes ago	Edit
3_model_building.ipynb	76.78 kB	9 minutes ago	Edit
4_train_models.py	5.01 kB	9 minutes ago	Edit
5_model_service_explainer.py	1.27 kB	9 minutes ago	Edit
6_application.py	2.62 kB	9 minutes ago	Edit
7_build_project.py	7.27 kB	9 minutes ago	Edit
cicd-build.sh	44 kB	9 minutes ago	Edit
...

Workspace: apac-how-cml
Cloud Provider: AWS (AWS)

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

Session Name	model_building
Runtime Editor	JupyterLab
Enable Spark	Yes - Spark version 2.4.8
Resource Profile	2 vCPU / 4 GiB

Click on **START SESSION**

Start A New Session

Session Name

Runtime

Editor Kernel Edition Version

JupyterLab Python 3.7 Standard 2023.05

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

Enable Spark Spark 2.4.8 - CDE 1.18.1 - HOTFIX-4

Runtime Image - docker.repository.cloudera.com/cloudera/cdsu/ml-runtime-jupyterlab-python3.7-standard:2023.05.2-b7

Resource Profile

Step 2 : Execute model building script

Open the `3_model_building.ipynb` file.

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a file tree with the following structure:

- /
- flask
- images
- models
- raw
- 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
- 7_build_project.py
- cdsw-build.sh
- churnexplainer.py
- README.md
- requirements.txt

The file `3_model_building.ipynb` is selected and highlighted in blue. The main pane displays the notebook content:Model Building

This notebook explores the building a churn model and the explainer that is used b

```
[2]: import os, datetime, subprocess, glob, sys
import dill
import pandas as pd
import numpy as np
import cdsw
import matplotlib.pyplot as plt
data_dir = '/home/cdsw'
```

Read in the Data

This section reads in the data into a pandas dataframe from the Hive table created Spark in local mode is fine.

```
[3]: from pyspark.sql import SparkSession
from pyspark.sql.types import *
spark = SparkSession\
    .builder\
    .appName("PythonSQL")\
    .master("local[*]")\
    .getOrCreate()
spark_df = spark.sql("SELECT * FROM default.telco_churn")
spark_df.printSchema()
df = spark_df.toPandas()
root
    |   customersID string (nullable = true)
```

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

Once the script finishes executing you can scroll to the bottom of the screen and check the part of the code where the built model is getting saved in pickle format.

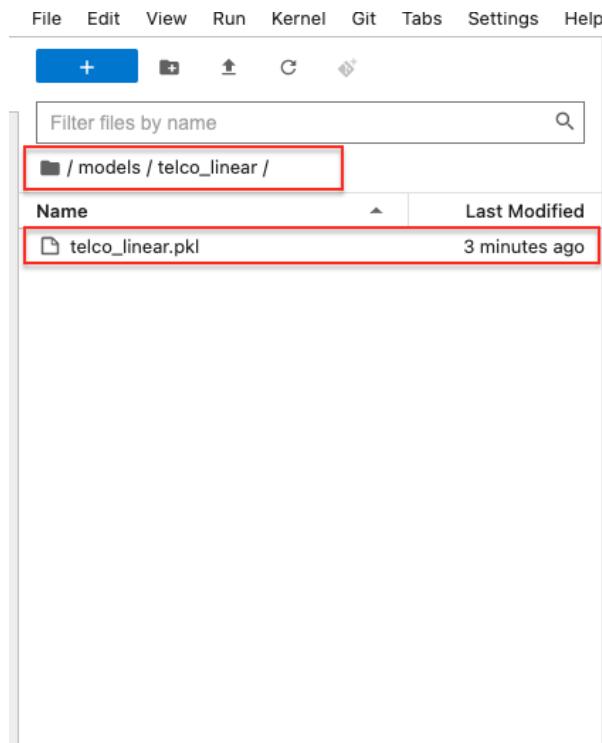
ExplainedModel class

The ExplainedModel class is a handy wrapper that combines the CategoricalEncoder, the Pipeline and the Lime Explainer to make it easier to make and predict new features and is used in other parts of this project.

```
[11]: from churnexplainer import ExplainedModel
explainedmodel = ExplainedModel(data=data, labels=labels, model_name='telco_linear',
                                 categoricalencoder=ce, pipeline=pipe,
                                 explainer=explainer, data_dir=data_dir)
explainedmodel.save()
```

```
[12]: spark.stop()
```

You can now see this pickle file created in your project directory as shown below.



Model Training

Step 1 : Create a session

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

Click on it and familiarize yourself with the code. This can be done by going into your project, clicking on **Files** in the left pane, and viewing the file ***4_train_models.py***

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with navigation links: All Projects, Overview, Sessions, Data, Experiments, Models, Jobs, Applications, Files (which is selected), Collaborators, and Project Settings. Below the sidebar, there are links for Help, version 2.0.38-b125, and Cloud Provider: AWS (AWS). The main area is titled 'mmehra / apac100_telco_churn_project / Files' and shows the file '4_train_models.py'. The code in the file is as follows:

```

## Model Training
# This script is used to train the Explained Model. It can be run in a session, or as job or as an experiment.

import os, datetime, subprocess, glob
import dill
import pandas as pd
import numpy as np
import cdsw

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegressionCV
from sklearn.pipeline import TransformerMixin
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer
from lime.lime_tabular import LimeTabularExplainer
from churnexplainer import ExplainedModel, CategoricalEncoder

data_dir = '/home/cdsw'

idcol = 'customerID'
labelcol = 'Churn'
cols = ((('gender', True),
          ('SeniorCitizen', True),
          ('Partner', True),
          ('Dependents', True),
          ('tenure', False),
          ('PhoneService', True),
          ('MultipleLines', True),
          ('InternetService', True),
          ('OnlineSecurity', True),
          ('OnlineBackup', True),
          ('DeviceProtection', True),
          ('TechSupport', True),
          ('StreamingTV', True))
         )

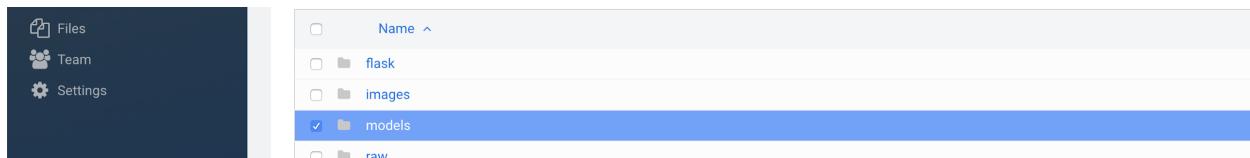
```

At the bottom of the code editor, it says 'Workspace: apac-how-cml' and 'Cloud Provider: aws (AWS)'.

The code also keeps track of the metrics associated to a particular train configuration:

```
"cdsw/cdsw-telco-churn-project-model-path")
mlflow.log_metric("train_score", round(train_score,2))
mlflow.log_metric("test_score", round(test_score,2))
```

The real model that is being trained can be seen here. Go to the `models` folder:



The screenshot shows the AI Platform's left sidebar with various project management options like Overview, Sessions, Experiments, Models, Jobs, Applications, Files, Team, and Settings. The 'Files' option is selected. The main area displays a file tree under 'Files / models', showing a single folder named 'telco_linear'.

To create a new session you can go into your project, select **Sessions** in the left pane, and click on **New Session**

The screenshot shows the AI Platform's project dashboard for 'apac100_telco_churn_project'. It includes sections for Models, Jobs, and Files. The 'Files' section shows a list of files and folders. A red arrow points to the 'New Session' button in the top right corner of the dashboard.

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

Session Name	experiment_runs
Runtime Editor	Workbench
Enable Spark	Yes - Spark version 2.4.8
Resource Profile	2 vCPU / 4 GiB

Click on **START SESSION**

Start A New Session

Session Name

Runtime

Editor	Kernel	Edition	Version
Workbench	Python 3.7	Standard	2023.05

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

Enable Spark Spark 2.4.8 - CDE 1.18.1 - HOTFIX-4

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2023.05.2-b7

Resource Profile

[Cancel](#) [Start Session](#)

Step 2 : Run Experiments

Select the file ***4_train_models.py***

Select ***Run -> Run All***

Once this runs successfully Go back to the **Project Page**

The screenshot shows a Jupyter Notebook interface with two main panes. The left pane contains the Python script `4_train_models.py` with syntax highlighting for code blocks. The right pane shows the results of the execution, including a table of metrics for a logistic regression model and a snippet of LIME explainer code.

```

1 ## Model Training
2 # This script is used to train the Explained Model. It can be run in a session, or
3 # as an experiment. In this example we will run it as an experiment.
4 import os, datetime, subprocess, glob
5 import pickle
6 import pandas as pd
7 import numpy as np
8 import cdsw
9 import mlflow
10
11 from sklearn.model_selection import train_test_split
12 from sklearn.metrics import classification_report
13 from sklearn.preprocessing import OneHotEncoder, StandardScaler
14 from sklearn.compose import ColumnTransformer
15 from sklearn.linear_model import LogisticRegressionCV
16 from sklearn.pipeline import TransformerMixin
17 from sklearn.preprocessing import LabelEncoder
18 from sklearn.compose import ColumnTransformer
19
20 from lime.lime_tabular import LimeTabularExplainer
21
22 from churnexplainer import ExplainedModel, CategoricalEncoder
23
24 mlflow.set_experiment("Telco Churn Experiments")
25 mlflow.start_run()
26
27 data_dir = '/home/cdsw'
28
29 idcol = 'customerID'
30 labelcol = 'Churn'
31 cols = ((gender, True),
32          (SeniorCitizen, True),
33          (Partner, True),
34          (Dependents, True),
35          (tenure, False),
36          (PhoneService, True),
37          (MultipleLines, True),
38          (InternetService, True),
39          (OnlineSecurity, True),
40          (OnlineBackup, True),
41          (DeviceProtection, True),
42          (TechSupport, True),
43          (StreamingTV, True),
44          (StreamingMovies, True),
45          (Contract, True),
46          (PaperlessBilling, True),

```

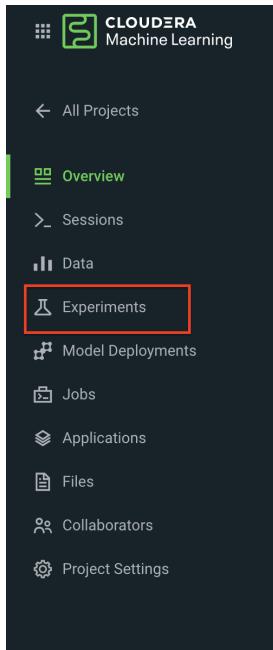
Session	Logs	Spark UI		
True	0.62	0.52	0.56	458
	accuracy		0.79	1758
	macro avg	0.73	0.70	0.71
	weighted avg	0.78	0.79	0.79

```

> data[labels.name + ' probability'] = pipe.predict_proba(X)[:, 1]
Create LIME Explainer
> feature_names = list(ce.columns_)
> categorical_features = list(ce.cat_columns_ix_.values())
> categorical_names = {i: ce.classes_[c] for c, i in ce.cat_columns_ix_.items()}
> class_names = [No + labels.name, labels.name]
> explainer = LimeTabularExplainer(ce.transform(data),
                                    feature_names=feature_names,
                                    class_names=class_names,
                                    categorical_features=categorical_features,
                                    categorical_names=categorical_names)
Create and save the combined Logistic Regression and LIME Explained Model.
> explainedmodel = ExplainedModel(data=data, labels=labels, model_name='telco_linear',
                                    categoricalencoder=ce, pipeline=pipe,
                                    explainer=explainer, data_dir=data_dir)
> explainedmodel.save()
If running as as experiment, this will track the metrics and add the model trained in this training run to the experiment history.
cdsw.track_metric("train/core", round(train.core, 2)) cdsw.track_metric("test/core", round(test.core, 2)) cdsw.track_metric("model/path", explainedmodel.model_path) cdsw.track_file(explainedmodel.model_path)

```

Select **Experiments** from the left tab



You will see that the experiment we ran from the session shows up here as it has the same name that we specified in the script.

```

5 import dill
6 import pandas as pd
7 import numpy as np
8 import cdsw
9 import mlflow
0
1 from sklearn.model_selection import train_test_split
2 from sklearn.metrics import classification_report
3 from sklearn.preprocessing import OneHotEncoder, StandardScaler
4 from sklearn.pipeline import Pipeline
5 from sklearn.linear_model import LogisticRegressionCV
6 from sklearn.pipeline import TransformerMixin
7 from sklearn.preprocessing import LabelEncoder
8 from sklearn.compose import ColumnTransformer
9
0 from lime.lime_tabular import LimeTabularExplainer
1
2 from churnexplainer import ExplainedModel, CategoricalEncoder
3
4 mlflow.set_experiment("Telco Churn Experiments")
5 mlflow.start_run()
6
7 data_dir = '/home/cdsw'
8
9 idcol = 'customerID'
0 labelcol = 'Churn'
1 cols = (('gender', True),
2          ('SeniorCitizen', True),
3          ('Partner', True),
4

```

mmehra / telco / Experiments

Search Experiments

BETA New Experiment

Name	Creator	Created At	Last Updated
Telco Churn Experiments	Manick Mehra	06/08/2023 9:14 PM	-

Displaying 1 - 1 of 1 < 1 > 25 / page

Step 3 : Verify Experiment metrics

Click on the Experiment you just ran and look for the train score and test score

mmehra / telco / Experiments / Telco Churn Experiments

Experiment (BETA)

Experiment Name: Telco Churn Experiments

Experiment ID: io7o-7lgy-ocbz-95b1

Artifact Location: /home/cdsw/experiments/io7o-7lgy-ocbz-95b1

> Notes

Runs (1)

Metrics

	Status	Start Time	Run Name	Duration	User	Source	Version	Models	test_score	train_score
<input type="checkbox"/>	<input checked="" type="checkbox"/>	2023-06-08 09:14:10	q0b2-qis2-alti-vj...	22.7s	mmehra	ipython3	1fd4e5	sklearn	0.79	0.81

We can run multiple experiments and based on the best score we can decide which model to go with. The model is also one of the columns in the Experiment details.

	Status	↓ Start Time	Run Name	Duration	User	Source	Version	Models	Metrics	
									test_score	train_score
<input type="checkbox"/>	✓	2023-06-08 09:14:10	q0b2-qis2-altl-vj...	22.7s	mmehra	ipython3	1fd4e5	 sklearn	0.79	0.81

This concludes the Lab

Lab 4 - Model Deploy/Serve

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.

STAGES

Data Ingest

Data Exploration

Model Build Experiment

Model Deploy

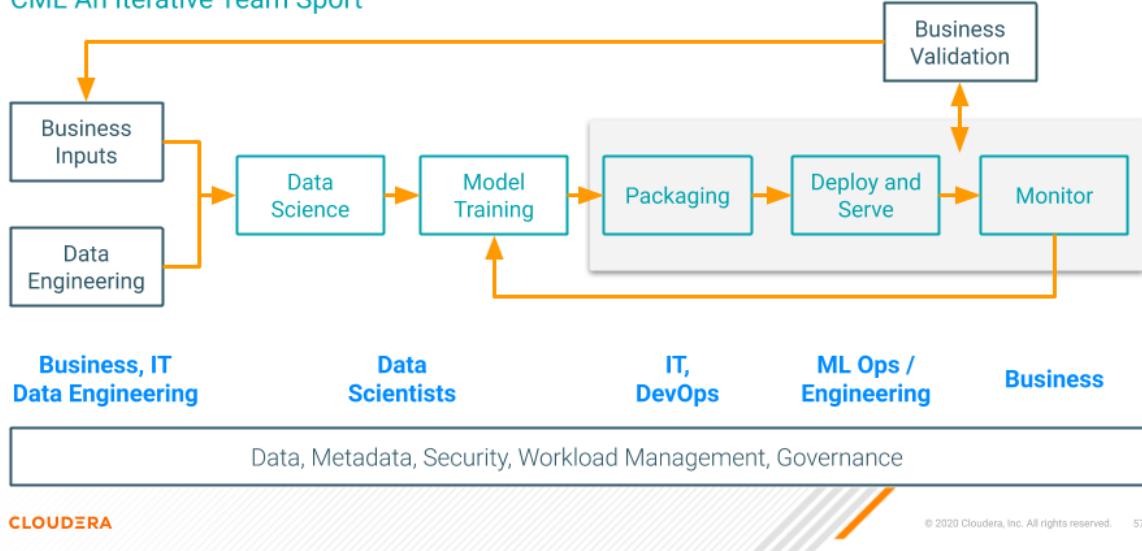
Application Deploy

The Cloudera logo, which consists of the word "CLOUDERA" in a bold, orange, sans-serif font.

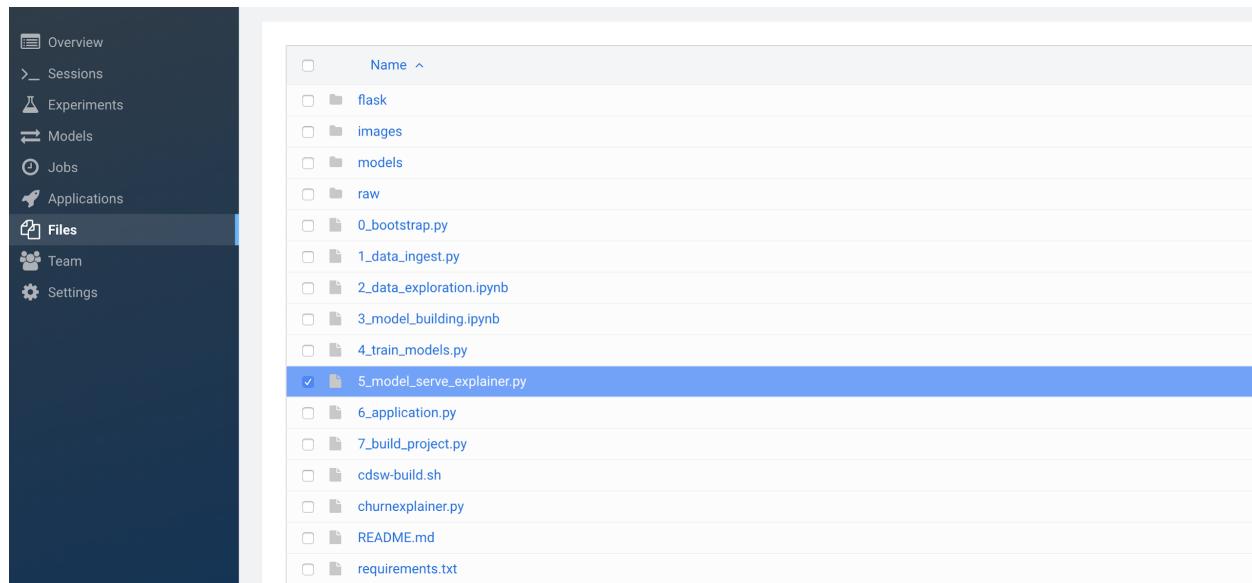
© 2020 Cloudera, Inc. All rights reserved. 56

MACHINE LEARNING IN PRODUCTION

CML An Iterative Team Sport



Click on **Files** in the left tab, and go to the file **5_model_server_explainer.py**:



This is the script for serving the model, and the line below is loading the pickle model we have generated in the Lab 3

```
em = ExplainedModel(os.getenv('CHURN_MODEL_NAME', 'test_model'))
```

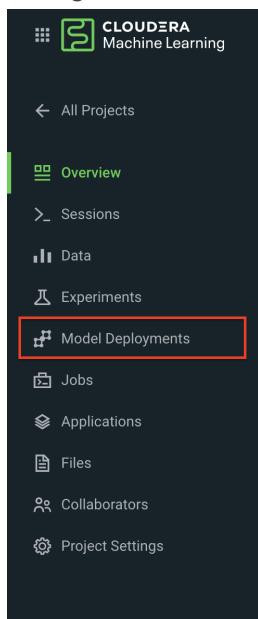
and the function is the one that takes the input arguments, passes them to the model for scoring and gives back the result of the score.

```
def explain(args):
    #data = dict(ChainMap(request.args, em.default_data))
    data = dict(ChainMap(args, em.default_data))
    data = em.cast_dct(data)
    probability, explanation = em.explain_dct(data)
    return jsonify({'data': dict(data),
                    'probability': probability,
                    'explanation': explanation})
```

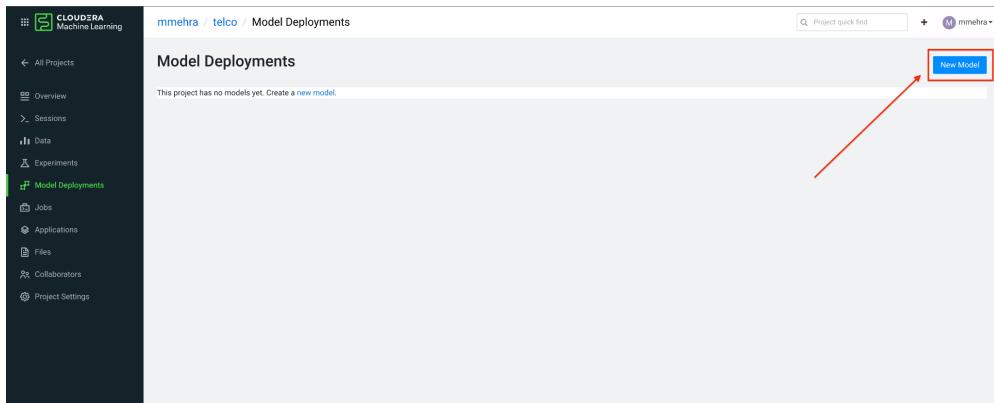
Copy the entire string as shown below because we are going to use it in the future and a sample JSON input parameters for the model.

```
{"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"}
```

Now go to **Model Deployments**



Click **New Model**



Enter the following details

Deployment Template	Deploy Model from Code
Name	<workload_username>_telco_churn_mod
Description	Deploying the telco churn model
Enable Authentication	False (Disable)
File	5_model_serve_explainer.py
Function	explain
Example Input	<pre>{"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"}</pre>
Runtime Editor	Workbench
Enable Spark	TRUE - Spark 2.4.8
Resource Profile	2 vCPU / 4 GiB

Deploy a Model

Deployment Template

- Deploy model from code
- Deploy registered model

General

Name *

apac100_telco_churn_model

Description *

Deploying the telco churn model

 Enable Authentication

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

Build

File *

5_model_serve_explainer.py

Function *

explain

Example Input i

```
{"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 (automati c)", "tenure": 29, "Dependents": "No", "OnlineSecurity": "No", "MultipleLines": "No", "InternetService": "DSL", "SeniorCitizen": "No", "TechSupport": "No"}
```

Example Output i

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

Runtime

Editor i

Workbench

Kernel i

Python 3.7

Edition i

Standard

Version

2023.05

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

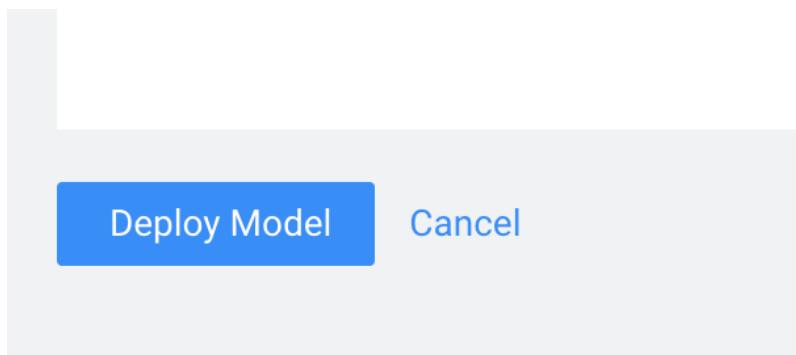
Spark 2.4.8 - CDE 1.18.1 - HOTFIX-4

Runtime Image - docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2023.05.2-b7

Comment

Deployment

Resource Profile



At this point you can deploy the model by clicking **Deploy model**

Model	Status	Replicas	CPU	Memory	Created By	Deployed By	Last Deployed	Actions
mmehra-mod-workshop01	Building	0 / 1	0	0 GiB	mmehra	mmehra	Apr 19, 2022, 07:33 PM	<button>Stop</button>

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

Model	Status	Replicas	CPU
Explain	Building	0 / 1	0

Building -> *Deploying*

Models

Model	Project	Status	Replicas	CPU
Explain	churn	Deploying	0 / 1	0



And finally *Deployed*

Models

Model	Project	Status	Replicas	CPU
Explain	churn	Deployed	1 / 1	1



Now you can click on the model name and test it ...

Explain

Overview Deployments Builds Monitoring Settings

Description
Sample Code

Explain Model Deployment

Shell | Python | R

```
curl -H 'Content-Type: application/json' -X POST https://modelservice.m1-02030da5-5dc.partners.dp5i-5vkq.cloudera.site/model -d '{"accessKey":"mmjr7bceqg5Sy5ka7364ygnBmzj86","request":{"StreamingTV":"No","MonthlyCharges":70.35,"PhoneService":"No","PaperlessBilling":"No","Partner":"No","OnlineBackup":"No","gender":"Female","Contract":"Month-to-month","TotalCharges":1397.475,"StreamingMovies":"No","DeviceProtection":"No","PaymentMethod":"Bank transfer (automatic)","tenure":29,"Dependents":"No","OnlineSecurity":"No","MultipleLines":"No","InternetService":"DSL","SeniorCitizen":"No","TechSupport":"No"}}'
```

Test Model

Input

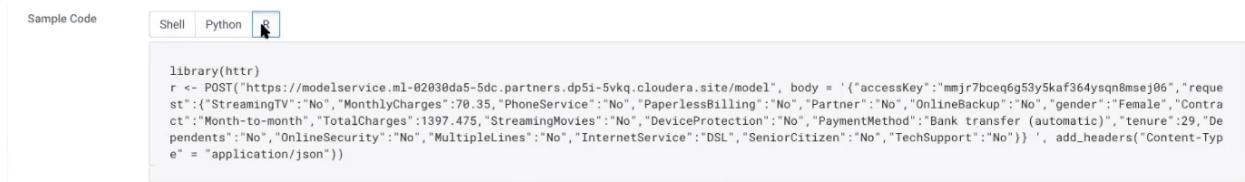
```
{"OnlineSecurity": "No",
"MultipleLines": "No",
"InternetService": "DSL",
"SeniorCitizen": "No",
"TechSupport": "No"
}'
```

Text Reset

... you should get the following:

Status	success
Response	{"data":{"MonthlyCharges":70.35,"MultipleLines":"No","TechSupport":"No","PhoneService":"No","OnlineBackup":"No","Contract":"Month-to-month","PaperlessBilling":"No"}}

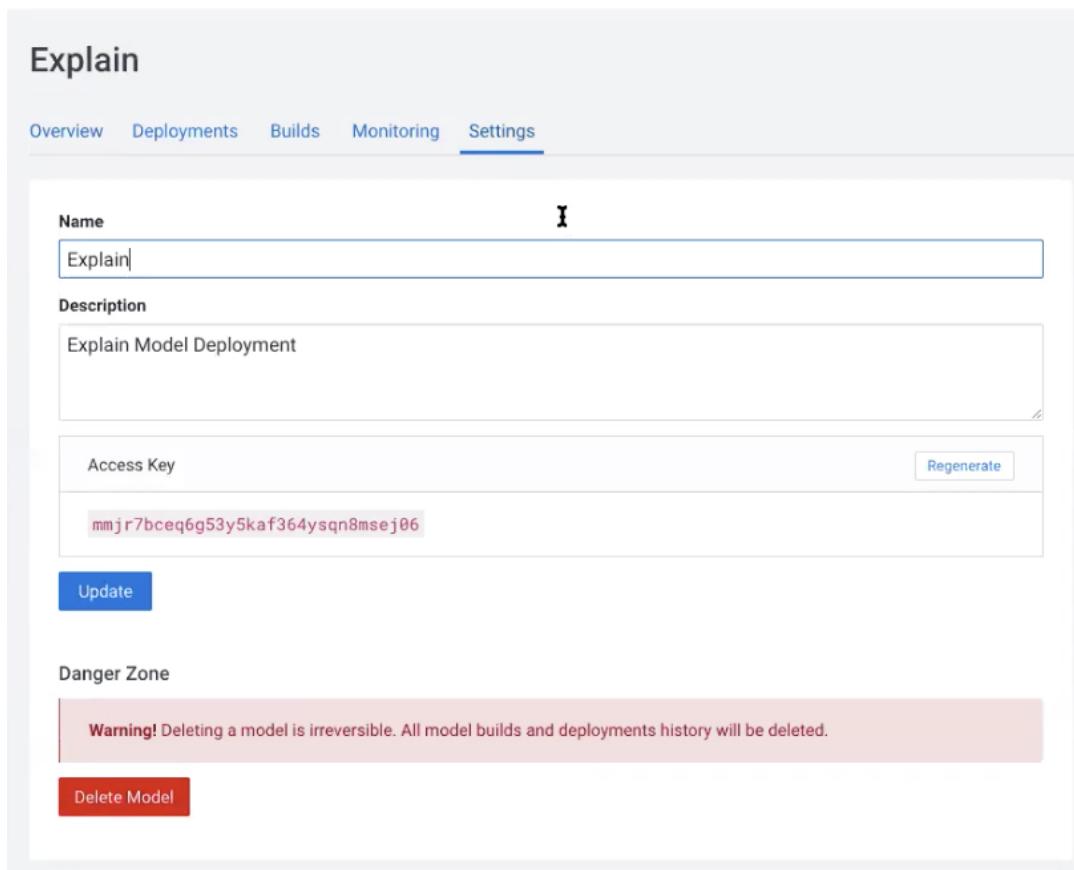
If you want to call the model from external services, the sample codes for invoking this REST endpoint are provided in Shell, Python and R



The screenshot shows a code editor window titled "Sample Code". It has tabs for "Shell", "Python", and "R", with "R" being the active tab. The code in the R tab is as follows:

```
library(httr)
r <- POST("https://modelservice.ml-02030da5-5dc.partners.dp5i-5vkq.cloudera.site/model", body = '{"accessKey":"mmjr7bceq6g53y5kaf364ysqn8msej06","request":{"StreamingTV":"No","MonthlyCharges":70.35,"PhoneService":"No","PaperlessBilling":"No","Partner":"No","OnlineBackup":"No","gender":"Female","Contract":"Month-to-month","TotalCharges":1397.475,"StreamingMovies":"No","DeviceProtection":"No","PaymentMethod":"Bank transfer (automatic)","tenure":29,"Dependents":"No","OnlineSecurity":"No","MultipleLines":"No","InternetService":"DSL","SeniorCitizen":"No","TechSupport":"No"} }', add_headers("Content-Type" = "application/json"))
```

As you can see the sample codes also provide an **accesskey** for invoking this model, so only the services that provide a correct **accesskey** can invoke it. You can manage the access key by going to **Settings**:



The screenshot shows the "Settings" tab of a model configuration interface. The "Name" field is set to "Explain". The "Description" field contains the text "Explain Model Deployment". The "Access Key" field displays the value "mmjr7bceq6g53y5kaf364ysqn8msej06", which is highlighted with a light blue background. Below the access key is a "Regenerate" button. A blue "Update" button is located at the bottom left. In the "Danger Zone" section, there is a red warning message: "Warning! Deleting a model is irreversible. All model builds and deployments history will be deleted." A red "Delete Model" button is located at the bottom left of this section.

Copy the Access Key for the next lab

Access Key

[Regenerate](#)

mmj r7bceq6g53y5kaf364ysqn8msej06

[Update](#)

The model can be monitored going to the **Monitoring** tab

christopherroyles / churn / Models / Explain / Monitoring

Explain

[Overview](#) [Deployments](#) [Builds](#) [Monitoring](#) [Settings](#)

Replica	Status	Received	Proce
explain-5-5-79cf99d9f9-4mll8	Ready	1	1 (10

Streams: stdout stderr

2020-04-22 04:24:41.634 Model ready	INFO	Model.Runtime	Finish Model initialization
2020-04-22 04:24:41.634 2020-04-22 15:24:41,634 32	INFO	Model.Runtime	Start Model initialization
2020-04-22 04:24:38.720 2020-04-22 15:24:38,720 32	INFO	Model.Runtime	Start Python model runtime in 32
2020-04-22 04:24:38.481 2020-04-22 15:24:38,480 32	INFO	Model.Runtime	

This concludes this lab.

Lab 5 - Application Deploy

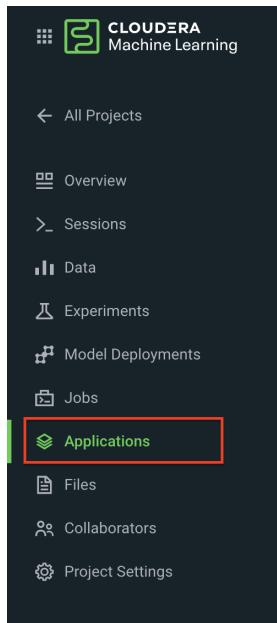
In this lab, you will create an application that embeds the model deployed in the previous lab, allowing business users, end-users that are not Data Scientists to interact and to get insight about the context of these analyses.

STAGES

- Data Ingest
- Data Exploration
- Model Build Experiment
- Model Deploy
- Application Deploy**

CLOUDERA © 2020 Cloudera, Inc. All rights reserved. 62

In the left tab, go to **Applications**



And as you can see we do not have any applications available yet. Go back to Files, here you can see that we provide the code on an application that is a Flask application as front-end

And the back-end is provided by the `6_application.py` code

And now update the access key in the `single_view.html` (this file is under the `flask` folder)

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with various project management options like Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, and Files. The 'Files' option is selected and highlighted with a green bar. The main area shows a file tree under 'mmehra / telco / Files'. Inside the 'flask' directory, several files are listed: ajax-loader.gif, churn_vis.css, churn_vis.js, env_vars.png, single_view.html (which is highlighted with a red box), and table_view.html.

Click on `single_view.html` and you can see the accesskey that we need to substitute to the one we copied in the previous lab

The screenshot shows the Cloudera Machine Learning interface. On the left, there's a sidebar with navigation links: All Projects, Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, Files (which is selected), Collaborators, and Project Settings. The main area shows a file tree under 'mmehra / telco / Files'. A specific file, 'single_view.html', is highlighted. The code editor on the right contains the content of 'single_view.html'.

```

<!DOCTYPE html>
<head>
<meta charset="utf-8">
<script src="https://d3js.org/d3.v5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.11/lodash.min.js"></script>
<link rel="stylesheet" type="text/css" href="churn_vis.css">
</head>
<body>
<h1>Single Prediction View</h1>
<div style='clear:both;' class="churn_div">
<div style='float:left;padding-left:20px;'>Churn Probability</div>
<div id='pred_value'></div>
</div>
<div id='features' style='clear:both;'></div>
<script>
const accessKey = "m321vuo2tffjgd0behaaw47aaxd3f1vi";
in_url = new URL(window.location.href);
out_url = new URL(window.location.origin + window.location.pathname)

params = {}
for (let p of in_url.searchParams.entries()) {
    params[p[0]] = p[1]
}
var features_numeric = d3.json('/stats').then(json => {
    return json;
});
var features_categorical = d3.json('/categories').then(json => {
    return json;
});

```

In order to edit this file, open it in a workbench (click Open in Session)

The screenshot shows a Jupyter Notebook interface with the 'single_view.html' file open. The left pane shows a file tree with various files like '1.data_ingest.py', 'churn', '2.build_project.py', '3.data_ingest.py', '2.data_exploration.ipynb', '3.train_models.py', '4.model_serve_explainer.py', '5.application.py', 'cdsw-build.sh', 'churnexplainer', 'flask', 'models', 'raw', 'README.md', 'requirements.txt', and 'utils'. The right pane shows the code editor with the same content as the previous screenshot. The bottom right corner has a 'Launch Session' button.

You do not need to launch a session. Just substitute the access key with yours and select **File > Save**.

```

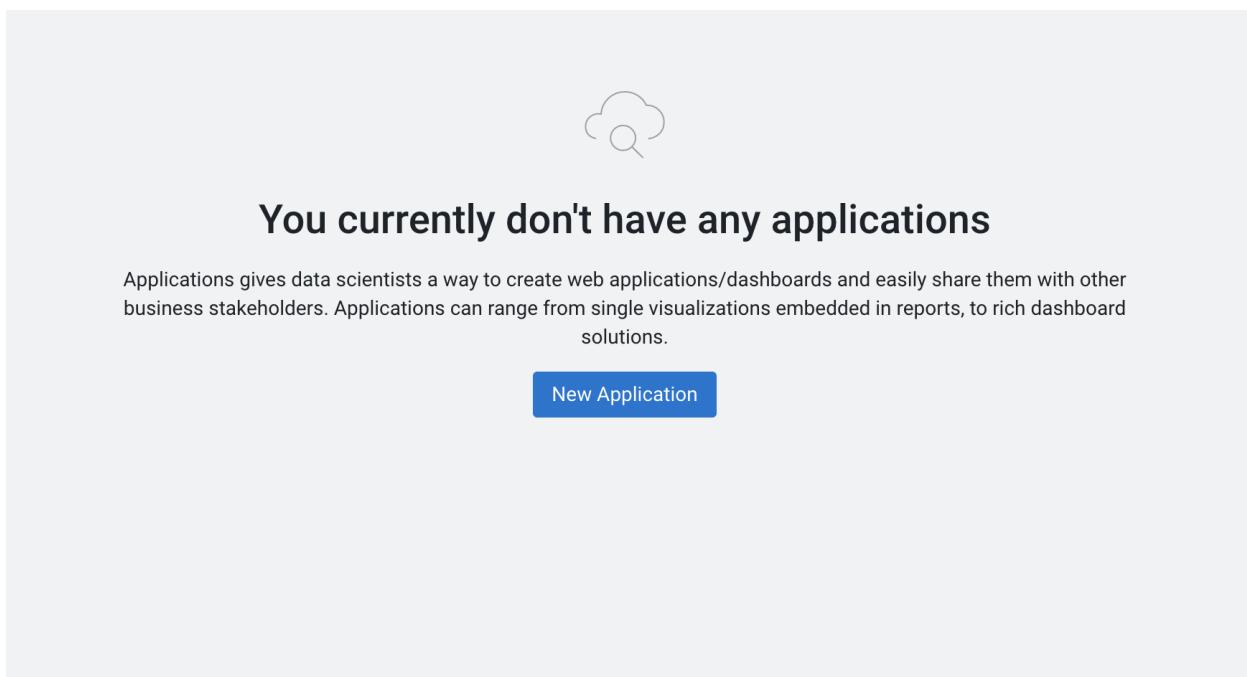
        <div style='float:left;padding-left:20px'>Churn Probability</div>
        <div id='pred_value'></div>

    </div>
    <div id='features' style='clear:both;'></div>
    <script>
        const accessKey = "msj7n0gwhl155j1ouy8yv15cpeqa9reI";
        in_url = new URL(window.location.href)
        out_url = new URL(window.location.origin + window.location.pathname)

        params = {}
        for (let p of in_url.searchParams.entries()) {
```

This is going to be used to call our deployed model when the end-user of this application will interact with it.

Now go back to Application, and click the New Application button.



Provide the following details.

Name	<username>_telco_churn_app
Subdomain	<username>telco
Description	Application that leverages the built model
Script	6_application.py

Runtime - Editor	Workbench
Enable Spark	TRUE - Spark 2.4.8
Resource Profile	2 vCPU / 4 GiB

Create an Application

General

Name
apac100_telco_churn_app

Subdomain *
apac100telco

Description
Application that leverages the built model

Script *
6_application.py

Runtime

Editor ⓘ	Kernel ⓘ
Workbench	Python 3.7

Edition ⓘ Version
Standard 2023.05

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

Enable Spark ⓘ Spark 2.4.8 - CDE 1.18.1 - HOTFIX-4 ⌂

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-standard:2023.05.2-b7

Resource Profile
2 vCPU / 4 GiB Memory

Environment Variables
CDSW_APP_POLLING_ENDPOINT / - +

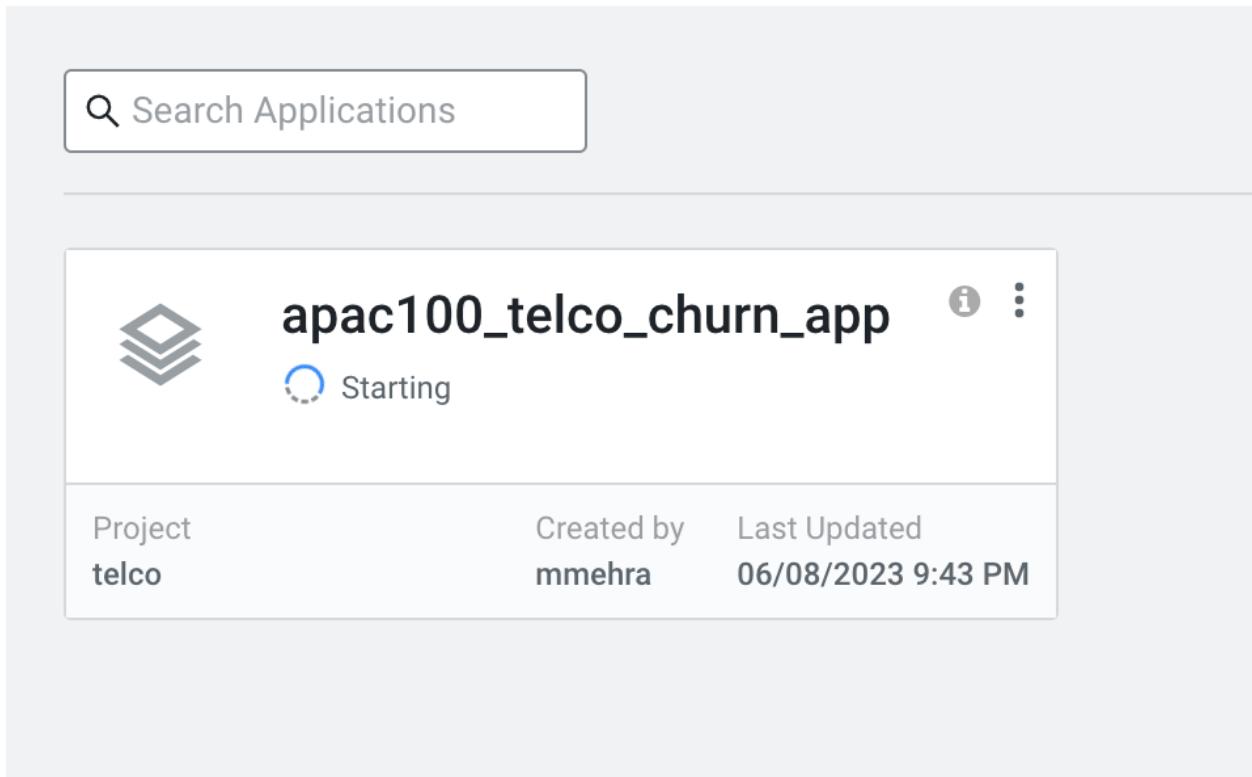
Environmental variables will override the [project environment](#).

Create Application

And then click **Create Application**

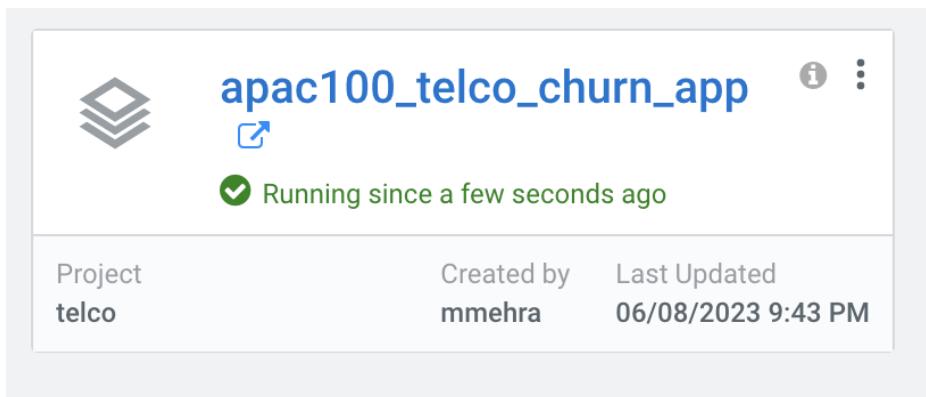
Then you should see the status **Starting** state

Applications



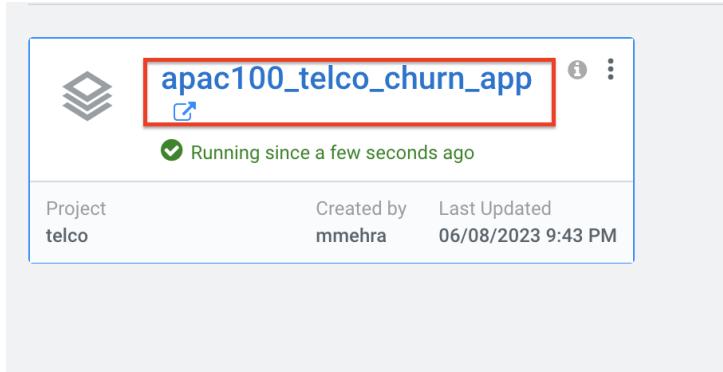
The screenshot shows the details of a newly created application named "apac100_telco_churn_app". The application is currently in a "Starting" state, indicated by a blue circular icon with a white arrow. It was created by "mmehra" on "06/08/2023 9:43 PM". The application is associated with the project "telco". A search bar at the top left is visible.

After a while it will change to **Running**

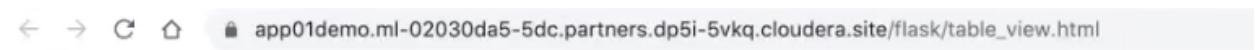


The screenshot shows the same application details after some time has passed. The status has changed to "Running since a few seconds ago", indicated by a green checkmark icon and a blue circular icon with a white checkmark. The other details remain the same: project "telco", created by "mmehra" on "06/08/2023 9:43 PM".

Click now in your newly created application



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



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	
	1846	0.717	Female	No	Yes	No	2	No	No ph	DSL	Yes	No	No	No	No	Yes	Month	Yes	Elect	55.1	68.75	
	3872	0.636	Male	No	No	No	1	Yes	No	DSL	No	No	No	No	No	No	Month	No	Elect	45.75	45.75	
	1242	0.402	Female	No	Yes	Yes	45	Yes	Yes	Fiber	Yes	Yes	No	No	Yes	Yes	Month	Yes	Elect	108.1	479.0	
	1203	0.301	Female	No	No	Yes	3	Yes	No	DSL	No	No	No	Yes	Yes	Yes	Month	Yes	Credi	69.4	571.4	
	709	0.217	Male	No	No	No	1	Yes	No	No	No	No in	No in	No in	No in	No in	Month	No	Male	19.55	19.55	
	6200	0.144	Female	No	No	No	35	No	No ph	DSL	No	No	Yes	Yes	Yes	No	No	Month	Yes	Male	35.45	117.5
	3569	0.113	Male	No	No	No	59	No	No ph	DSL	No	No	No	Yes	Yes	Yes	Month	Yes	Elect	51.7	300.5	
	5661	0.111	Female	No	Yes	Yes	17	Yes	Yes	No	No in	No in	No in	No in	No in	No in	Month	No	Male	24.1	409.9	
	3730	0.104	Female	No	No	No	8	Yes	No	No	No	No in	No in	No in	No in	No in	Oney	Yes	Bank	20.2	140.9	
	3382	0.030	Female	No	Yes	No	65	Yes	No	DSL	Yes	No	Yes	No	No	No	Month	Yes	Male	55.15	367.3	

Click on one of the item in the Probability column

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.

Single Prediction View

Churn Probability	0.946			
MonthlyCharges	35.1	0.31	mean 64.80 min 18.25 max 118.75	<input type="button"/> Submit
MultipleLines	No phone service	0	No No phone service Yes	
TechSupport	No	0.04	No No internet service Yes	
PhoneService	No	0	No Yes	
OnlineBackup	No	0	No No internet service Yes	
Contract	Month-to-month	0.11	Month-to-month One year Two year	
PaperlessBilling	Yes	0	No Yes	
InternetService	Fiber optic	1.19	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="button"/> Submit
tenure	2	0.30	mean 32.42 min 1.00 max 72.00	<input type="button"/> Submit
gender	Female	0	Female Male	
SeniorCitizen	No	0	No Yes	
StreamingTV	No	-0.03	No No internet service Yes	
DeviceProtection	No	0	No No internet service Yes	
OnlineSecurity	No	0.05	No No internet service Yes	
Partner	Yes	0	No Yes	
Dependents	No	0	No Yes	

Everytime you click and change a value, the application will call our model. To check this go back to your model deployed, click Monitoring bd you should see that the Receive value is increased

The screenshot shows the Cloudera Machine Learning web interface. The top navigation bar includes the Cloudera logo and the path: christopherroyles / churn / Models / Explain / Monitoring. On the left, a sidebar menu lists: All Projects, Overview, Sessions, Experiments, **Models**, Jobs, Applications, Files, Team, and Settings. The main content area is titled "Explain" and has tabs for Overview, Deployments, Builds, **Monitoring**, and Settings. Under the Monitoring tab, there is a table with one row labeled "explain-5-5-79ef99d9f9-4mll8". The table columns are Replica, Status, Received, and Processed (with a value of 7 (100 %)). Below the table, a section titled "Streams:" shows log entries with checkboxes for stdout and stderr. The logs include:

Time	Message
2020-04-22 04:24:41.634	Model ready
2020-04-22 04:24:41.634	2020-04-22 15:24:41,634 32 INFO Model.Runtime Finish Model initialization
2020-04-22 04:24:38.720	2020-04-22 15:24:38,720 32 INFO Model.Runtime Start Model initialization
2020-04-22 04:24:38.481	2020-04-22 15:24:38,480 32 INFO Model.Runtime Start Python model runtime in 32

This concludes the lab.