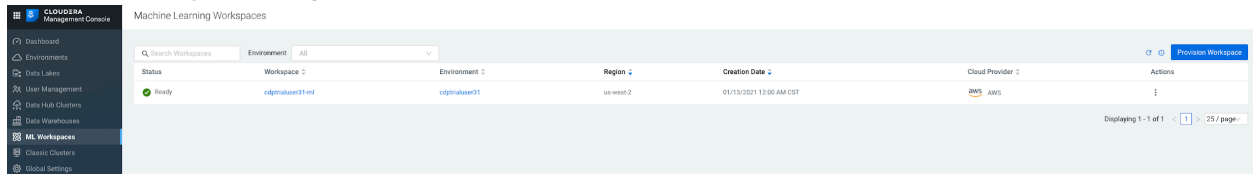
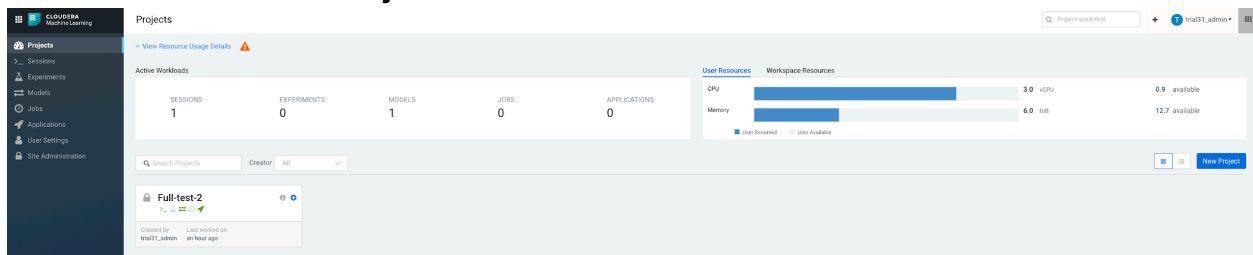


A Workspace is a small cluster that runs on a kubernetes service to provide teams of data scientists to develop, test, train, and ultimately deploy machine learning models. **Click into the Workspace by clicking the Workspace name.**



You can visualize all of the Projects and Resources are part of the Projects page. Next we will create a Project where we will develop and deploy models along with other CML features. **Click on “New Project”**



When creating a new project give a Name, Visibility, and initial configuration.

Project Name: Telco_churn

Visibility: Private

Initial Setup: Git -> <https://github.com/andy-hansen/cml.git>

Create a New Project

Project Name

Project Visibility

☒ **Private** - Only added collaborators can view the project.

☐ **Public** - All authenticated users can view this project.

Initial Setup

Blank

Template

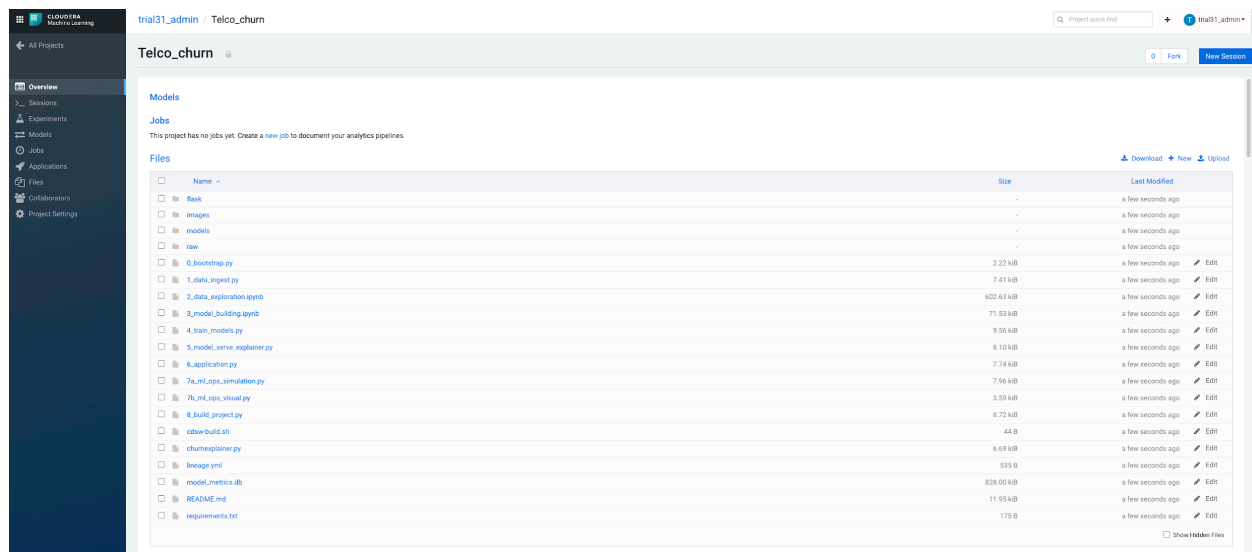
Local

Git

Create Project

Part 2: CML Project Overview

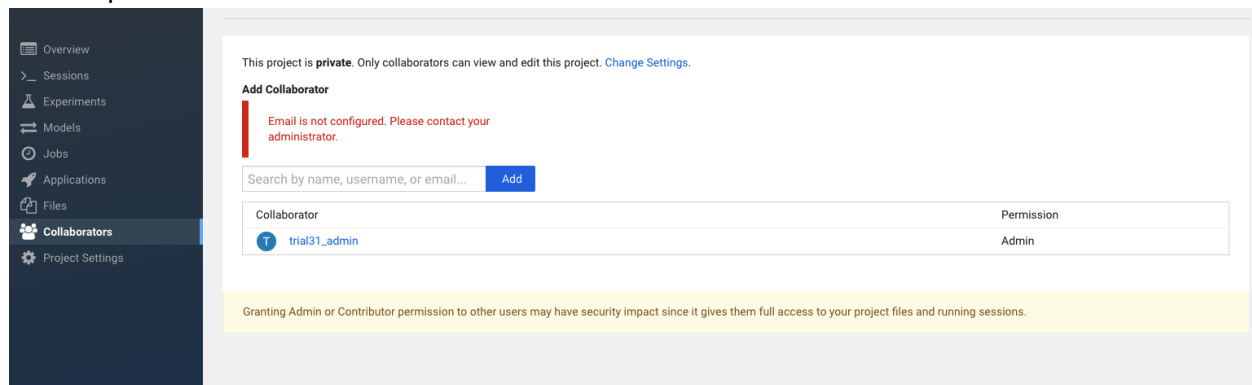
Overview gives you access into all the features of a CML project. We will only have files copied from the Github repo currently. Initially it is good to start on the management components of a project.



Collaborators:

For our demo we aren't adding additional collaborators.

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



Project Settings:

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

We won't be changing any settings for the demo.

CLOUDERA
Machine Learning

← All Projects

Overview

Sessions

Experiments

Models

Jobs

Applications

Files

Collaborators

Project Settings

trial31_admin / Telco_churn / Settings

Q Proj

Project Settings

Options Runtime/Engine Advanced Tunnels Delete Project

Project Name

Telco_churn

Description

Description

Visibility

- ☒ **Private** - Only collaborators can view or edit the project.
- ☐ **Public** - All authenticated users can view this project. Collaborators can also edit the project.

Update Project

CLOUDERA
Machine Learning

← All Projects

Overview

Sessions

Experiments

Models

Jobs

Applications

Files

Collaborators

Project Settings

trial31_admin / Telco_churn / Settings / Runtime/Engine

Q

Project Settings

Options Runtime/Engine Advanced Tunnels Delete Project

Default Engine: ☐ ML Runtime ? ☒ Legacy Engine ?

Engine Image

Select the Docker image that Cloudera Machine Learning should use to run sessions and jobs in this project. If you'd like to use a different image, contact your site administrator.

Default engine image, docker.repository.cloudera.com/cloudera/cdsw/engine:13-cml-2020.10-2

Save Engine

Third-party editors

Cloudera Machine Learning allows you to launch sessions with third-party, web-based editors. To add an editor to the Start New Session menu, first launch a session with the built-in Workbench editor and install the third-party editor of your choice. Then, come back to this page and provide a name for the editor and the command to start the editor server. Ensure that you start the server on the port specified by the `CDSW_APP_PORT` environment variable.

+ New Editor

CLUSTERA
Machine Learning

← All Projects

Overview

Sessions

Experiments

Models

Jobs

Applications

Files

Collaborators

Project Settings

trial31_admin / Telco_churn / Settings / Engine

Q Proj

Project Settings

Options Runtime/Engine **Advanced** Tunnels Delete Project

Environment Variables

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

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

Name	Value	Actions
STORAGE	s3a://prod-cdptrialuser31-trycdp-com	Delete
<input type="text"/>	<input type="text"/>	Add

Shared Memory Limit

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

Save Advanced Settings

trial31_admin / Telco_churn / Settings / SSH Tunnels

Q

Project Settings

Options Runtime/Engine Advanced **Tunnels** Delete Project

SSH Tunnels

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

+ New Tunnel

Part 3: CML Sessions and Workbench

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

From the Overview page click on New Session

Telco_churn

Models

Jobs

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

Files

Name	Size	Last Modified
flask	-	a few seconds ago
images	-	a few seconds ago
models	-	a few seconds ago
raw	-	a few seconds ago
0_bootstrap.py	2.22 KiB	a few seconds ago
1_data_ingest.py	7.41 KiB	a few seconds ago
2_data_exploration.ipynb	602.63 KiB	a few seconds ago
3_model_building.ipynb	71.53 KiB	a few seconds ago
4_train_models.py	9.56 KiB	a few seconds ago
5_model_serve_explainer.py	8.10 KiB	a few seconds ago
6_application.py	7.74 KiB	a few seconds ago
7a_ml_ops_simulation.py	7.96 KiB	a few seconds ago
7b_ml_ops_visual.py	3.59 KiB	a few seconds ago
8_build_project.py	8.72 KiB	a few seconds ago
cdsw-build.sh	44 B	a few seconds ago
churnexplainer.py	6.69 KiB	a few seconds ago
lineage.yml	539 B	a few seconds ago
model_metrics.db	828.00 KiB	a few seconds ago
README.md	11.95 KiB	a few seconds ago
requirements.txt	175 B	a few seconds ago

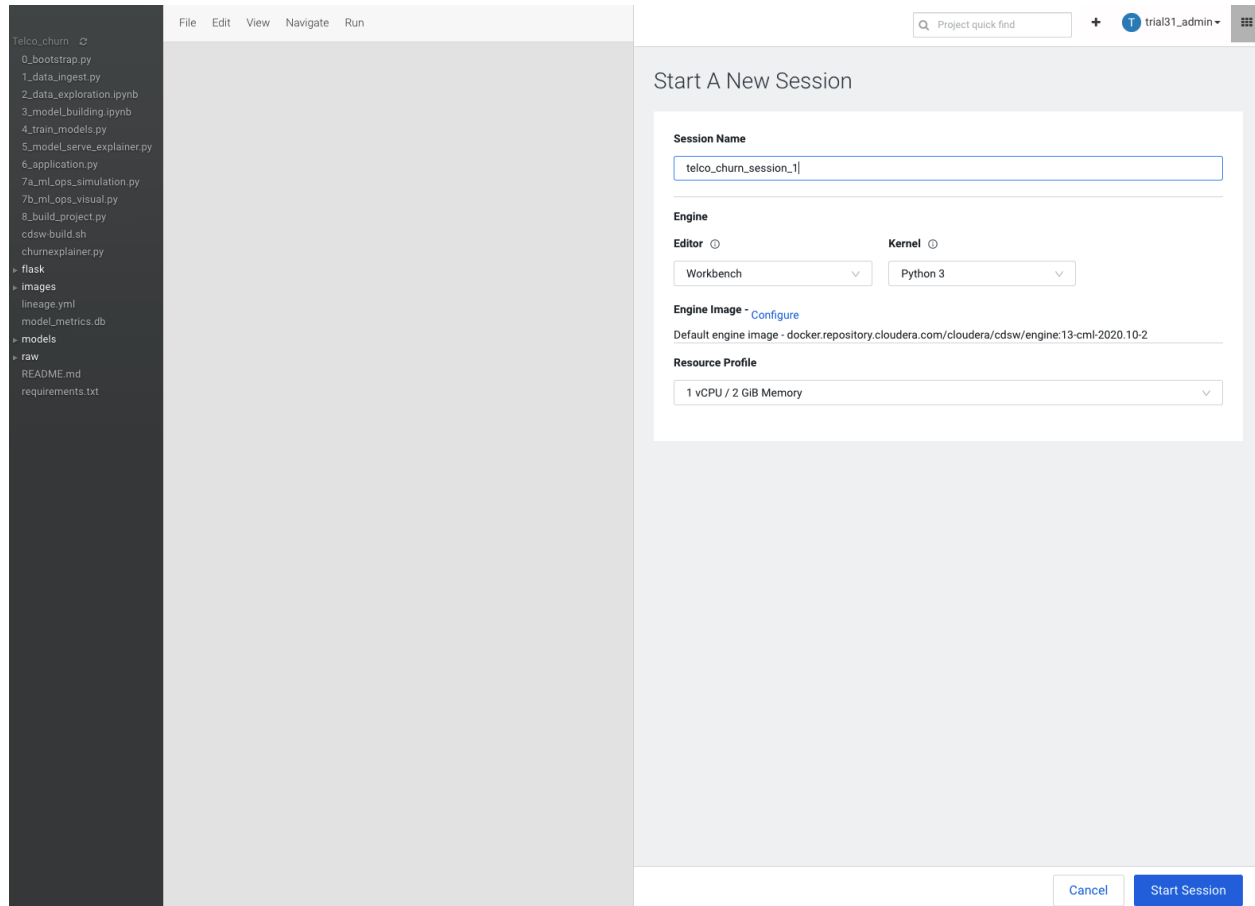
[Download](#) [New](#) [Upload](#)

☐ Show Hidden Files

Session Name: telco_churn_session_1

Editor: Workbench
Kernel: Python 3
Engine Image: Default
Resource Profile: 1vCPU/2 GiB Memory

Then select **Start Session**



The Workbench is now starting up and deploying a container onto the workspace at this point. Going from left to right you will see the project files, editor pane, and session pane. **Once you see the flashing red line on the bottom of the session pane turn steady green the container has been successfully started.**

The screenshot displays a JupyterLab environment with two main panes. The left pane shows a file explorer with a tree structure of files and folders, including 'telco_churn', '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_rml_ops_simulation.py', '7b_rml_ops_visual.py', '8_build_project.py', 'cdsw-build.sh', 'churnexplainer.py', 'flask', 'images', 'lineage.yml', 'model_metrics.db', 'models', 'raw', 'README.md', and 'requirements.txt'. The '8_build_project.py' file is selected and its contents are displayed in the central editor pane. The script is a Python file named '8_build_project.py' that automates the deployment of a model, installation of requirements, and deployment of the model. It includes comments and code for installing requirements, setting environment variables, creating directories, and running the data ingest file. The right pane shows a terminal session titled 'telco_churn_session_1' with a 'Running' status. The terminal output indicates that the session is running on a Python 3 environment with 1 vCPU and 2 GiB Memory. The terminal also shows the 'Getting Started' section, which provides instructions on how to execute code from the editor, select code, and execute it with 'Command-Enter' on Mac or 'Ctrl-Enter' on Windows. The terminal also shows the 'Use ?command' to get help on a particular command.

```
File Edit View Navigate Run 8_build_project.py

1 # Run this file to auto deploy the model, run a job, and deploy the
2
3 # Install the requirements
4 !pip3 install -r requirements.txt
5 import subprocess
6 import datetime
7 import xml.etree.ElementTree as ET
8 import requests
9 import json
10 import time
11 import os
12 from IPython.display import Javascript, HTML
13 from cmlbootstrap import CMLBootstrap
14
15
16 # Create the directories and upload data
17
18
19 run_time_suffix = datetime.datetime.now()
20 run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S")
21
22
23 HOST = os.getenv("CDSW_API_URL").split(
24     "://")[0] + "://" + os.getenv("CDSW_DOMAIN")
25 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
26     "://")[0] + "://" + os.getenv("CDSW_PROJECT")
27 API_KEY = os.getenv("CDSW_API_KEY")
28 PROJECT_NAME = os.getenv("CDSW_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     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
38         tree = ET.parse("/etc/hadoop/conf/hive-site.xml")
39         root = tree.getroot()
40         for prop in root.findall('property'):
41             if prop.find('name').text == "hive.metastore.warehouse":
42                 storage = prop.find('value').text.split(
43                     "://")[0] + "://" + prop.find('value').text.split(
44                         "://")[1]
45     else:
46         storage = "/user/" + os.getenv("HADOOP_USER_NAME")
47     storage_environment_params = {"STORAGE": storage}
48     storage_environment = cml.create_environment_variable(
49         storage_environment_params)
50     os.environ["STORAGE"] = storage
51
52 # This will run the data ingest file. You need this to create the
53 # csv file.
54 exec(open("1_data_ingest.py").read())
55
56 # Get User Details
57 user_details = cml.get_user({})
58 user_obj = {"id": user_details["id"], "username": USERNAME,
59             "name": user_details["name"],
60             "type": user_details["type"],
61             "html_url": user_details["html_url"],
62             "url": user_details["url"]}
63
64 # Get Project Details
65 project_details = cml.get_project({})
66
67
68
69
70
71
```

telco_churn_session_1 Running

By trial31_admin - Python 3 Session - 1 vCPU / 2 GiB Memory - a few seconds ago

Session Logs Collapse Share Export PDF

Getting Started

This is your Python 3 session. Your editor is on the left and your input prompt is on the bottom.

To install a package type: `!pip3 install [package_name]` at the input prompt.

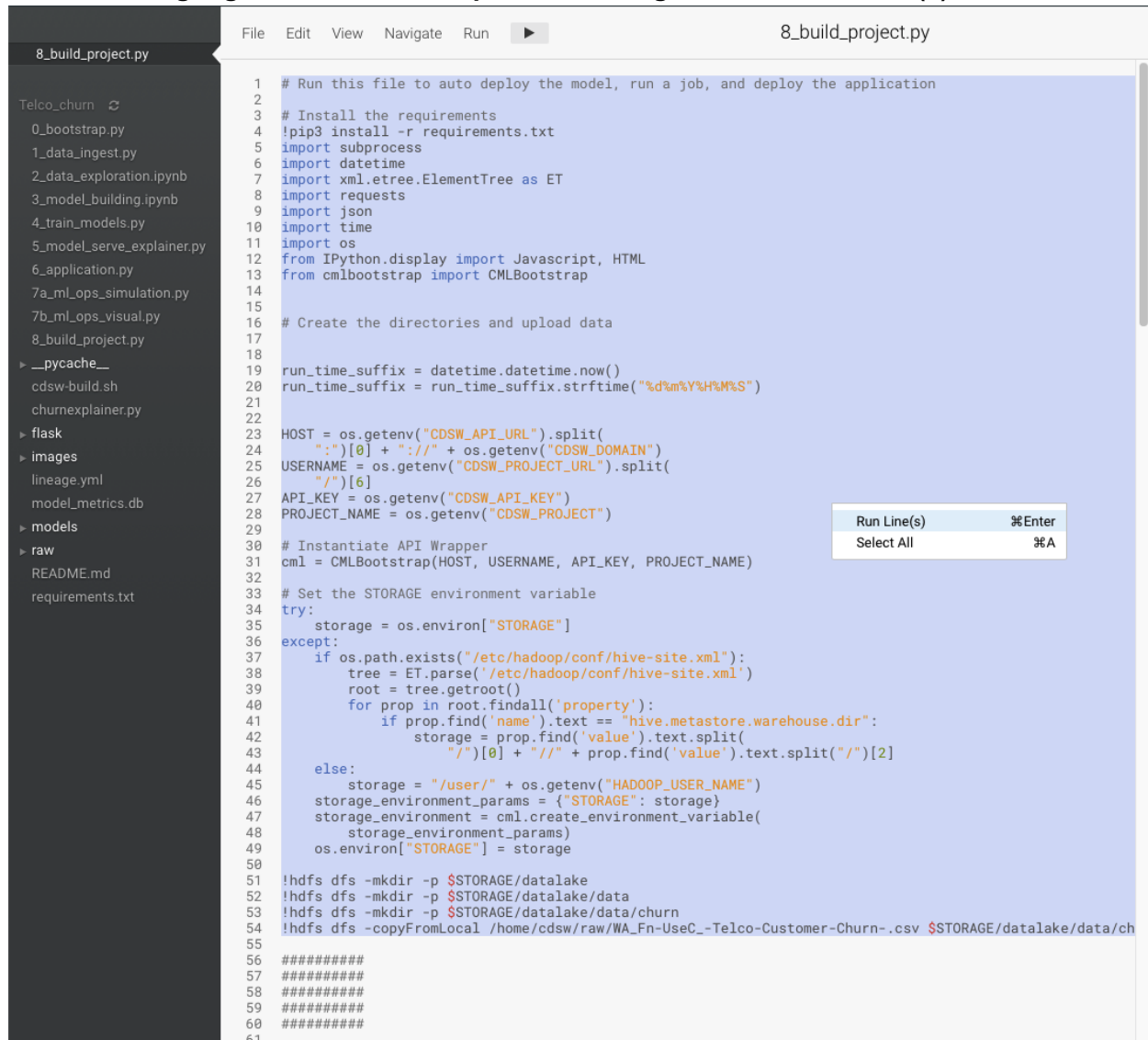
To execute code from the editor, select the code and execute it with `Command-Enter` on Mac or `Ctrl-Enter` on Windows. You can also enter code at the prompt below.

Use `?command` to get help on a particular command.

Open up the script 8_build_project.py from the left pane. In the editor pane we are going to select and run the script in several parts. Throughout the script you will see breaks in the code defined by Part 1, Part 2, Part 3, etc.

Part 1: Will install any required packages to be used. As an example, flask is installed as part of the project. A variable is also set as part of the project. Last but not least we are loading a file from the project as a test dataset and moving that to a s3 location.

Select and highlight the contents of part 1, then Right click -> Run Line(s)



```
1 # Run this file to auto deploy the model, run a job, and deploy the application
2
3 # Install the requirements
4 !pip3 install -r requirements.txt
5 import subprocess
6 import datetime
7 import xml.etree.ElementTree as ET
8 import requests
9 import json
10 import time
11 import os
12 from IPython.display import Javascript, HTML
13 from cmlbootstrap import CMLBootstrap
14
15
16 # Create the directories and upload data
17
18
19 run_time_suffix = datetime.datetime.now()
20 run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S")
21
22
23 HOST = os.getenv("CDSW_API_URL").split(
24     ":[0] + " + "://" + os.getenv("CDSW_DOMAIN")
25 )
26 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
27     ":[0] + " + "://" + os.getenv("CDSW_DOMAIN")
28 )
29 API_KEY = os.getenv("CDSW_API_KEY")
30 PROJECT_NAME = os.getenv("CDSW_PROJECT")
31
32 # Instantiate API Wrapper
33 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
34
35 # Set the STORAGE environment variable
36 try:
37     storage = os.environ["STORAGE"]
38 except:
39     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
40         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
41         root = tree.getroot()
42         for prop in root.findall('property'):
43             if prop.find('name').text == "hive.metastore.warehouse.dir":
44                 storage = prop.find('value').text.split(
45                     ":[0] + " + "://" + prop.find('value').text.split("/") [2]
46                 )
47             else:
48                 storage = "/user/" + os.getenv("HADOOP_USER_NAME")
49                 storage_environment_params = {"STORAGE": storage}
50                 storage_environment = cml.create_environment_variable(
51                     storage_environment_params
52                 )
53                 os.environ["STORAGE"] = storage
54
55 !hdfs dfs -mkdir -p $STORAGE/datalake
56 !hdfs dfs -mkdir -p $STORAGE/datalake/data
57 !hdfs dfs -mkdir -p $STORAGE/datalake/data/churn
58 !hdfs dfs -copyFromLocal /home/cds/raw/WA_Fn-UseC_-Telco-Customer-Churn.csv $STORAGE/datalake/data/ch
59
60 #####
61 #####
62 #####
63 #####
64 #####
65 #####
```

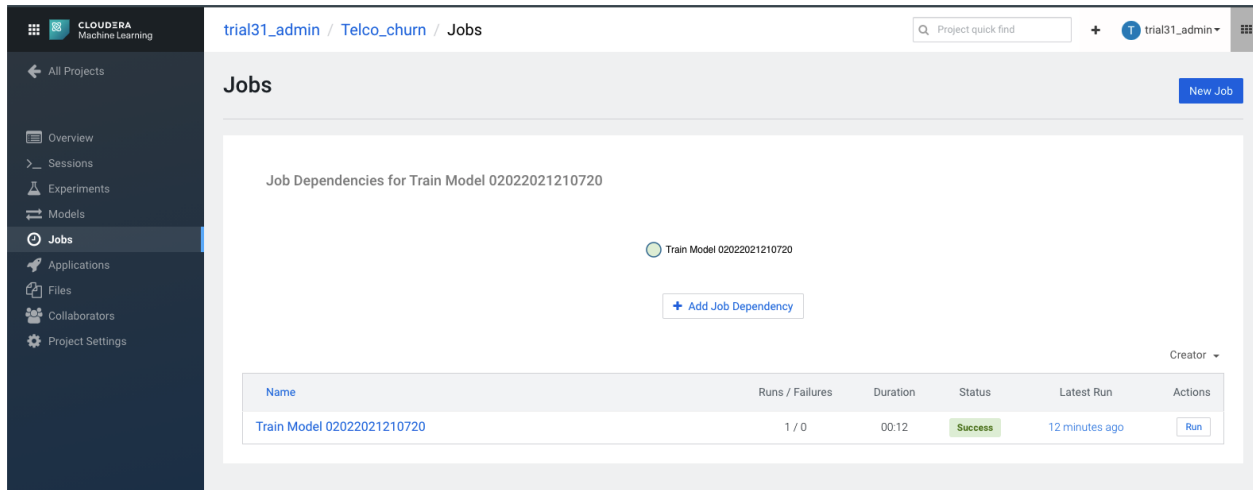
Part 2: The telco churn dataset is ingested from s3 and a hive table is created using Spark.

Select and highlight the contents of part 2, then Right click -> Run Line(s)

Part 3: Create a CML Job and start the job. A job automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

Select and highlight the contents of part 3, then Right click -> Run Line(s)

Once the Job is started we will look at what was created. **Click the Project button** (top right corner of screen). **Click on Jobs** to explore the job that was created and details can be found by **Clicking on the Job Name**.



The screenshot shows the Cloudera Machine Learning interface. The left sidebar contains navigation links: All Projects, Overview, Sessions, Experiments, Models, Jobs (selected), Applications, Files, Collaborators, and Project Settings. The main area is titled 'Jobs' and shows 'Job Dependencies for Train Model 02022021210720'. It features a circular dependency icon, a '+ Add Job Dependency' button, and a table with job details.

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Model 02022021210720	1 / 0	00:12	Success	12 minutes ago	Run

Part 4: Using CML, you can create any function within a script and deploy it to a REST API. In a machine learning project, this will typically be a predict function that will accept an input and return a prediction based on the model's parameters.

Select and highlight the contents of part 4, then Right click -> Run Line(s)

When getting to deploying the Model this can take a little time, and is a good spot to stretch the legs and refill your coffee.

```
179 with open('lineage.yml', 'w') as lineage:
180     lineage.write(yaml_text)
181
182 #####
183 #####
184 #####
185 #####
186 #####
187
188 # Create Model
189 example_model_input = {'StreamingTV': 'No', 'MonthlyCharges': 70.3,
190                        'StreamingMovies': 'No', 'DeviceProtection':
191
192 create_model_params = {
193     'projectId': project_id,
194     'name': 'Model Explorer ' + run_time_suffix,
195     'description': 'Explain a given model prediction',
196     'visibility': 'private',
197     'enableAuth': False,
198     'targetFilePath': '5_model_serve_explainer.py',
199     'targetFunctionName': 'explain',
200     'engineImageId': default_engine_image_id,
201     'kernel': 'python3',
202     'examples': [
203         {
204             'request': example_model_input,
205             'response': {}
206         }
207     ],
208     'cpuMillicores': 1000,
209     'memoryMb': 2048,
210     'nvidiaGPUs': 0,
211     'replicationPolicy': {'type': 'fixed', 'numReplicas': 1},
212     'environment': {}
213 }
214
215 new_model_details = cml.create_model(create_model_params)
216 access_key = new_model_details['accessKey'] # todo check for bad
217 model_id = new_model_details['id']
218
219 print("New model created with access key", access_key)
220
221 # Disable model_authentication
222 cml.set_model_auth({'id': model_id, 'enableAuth': False})
223
224 # Wait for the model to deploy.
225 is_deployed = False
226 while is_deployed == False:
227     new_model_details['id'], 'latestModelDeployment': True,
228     if model['latestModelDeployment']['status'] == 'deployed':
229         print("Model is deployed")
230         break
231     else:
232         print("Deploying Model.....")
233         time.sleep(10)
234
235 #####
236 #####
237 ##### Good time to take a break
238 #####
239 #####
240
241 # Change the line in the flask/single_view.html file.
242 subprocess.call(['sed', '-i', 's/const/accessKey.*/const accessKey = ' + access_key + ';/', '/home/cdsw/flask/single_view
243
244 # Change the model_id value in the 7a_model_operations.py and 7b_m
245 subprocess.call(['sed', '-i', 's/model_id = */model_id = ' + model_id + ';/', '/home/cdsw/7a_ml_ops_simulation
246 subprocess.call(['sed', '-i', 's/model_id = */model_id = ' + model_id + ';/', '/home/cdsw/7a_ml_ops_simulation
247
248
249
250
```

```
telco_churn_session_1
By trial31_admin - Python Session - 1 vCPU / 2 GiB Memory - 28 minutes ago

Session Logs Spark UI
print( New model created with access key', access_key)
New model created with access key ml8mrkticueiyejlz1z121fxndyjkdw
Disable model_authentication
> cml.set_model_auth({'id': model_id, 'enableAuth': False})
{'accessKey': 'ml8mrkticueiyejlz1z121fxndyjkdw',
 'authEnabled': False,
 'createdAt': '2021-02-02T21:07:55.221Z',
 'creator': {'id': 3, 'type': 'user', 'username': 'trial31_admin'},
 'creatorId': 3,
 'crn': 'crn:cdp:ml:us-west-1:f209dbac-bd43-4198-b7d1-7ecf68c2a8f4:workspace:8a7b7049-882d-4806-9f1e-999f4f759df7/3156dd9a-c500-462c-ad6a-92a751a1da8d',
 'defaultReplicationPolicy': {'numReplicas': 1, 'type': 'fixed'},
 'defaultResources': {'cpuMillicores': 1000,
 'memoryMb': 2048,
 'nvidiaGPUs': 0},
 'description': 'Explain a given model prediction',
 'htmlUrl': 'https://ml-86b5138c-077.cdp.trial.hs71-0qe7.cloudera.site/trial31_admin/telco_churn/models/4',
 'id': '4',
 'name': 'Model Explorer 020220210720',
 'namespace': 'mlx-user-3',
 'project': {'crn': 'crn:cdp:ml:us-west-1:f209dbac-bd43-4198-b7d1-7ecf68c2a8f4:workspace:8a7b7049-882d-4806-9f1e-999f4f759df7/8e442fec-d009-4ccb-a684-4074df4ba7f1',
 'id': 10,
 'name': 'telco_churn',
 'slug': 'telco_churn'},
 'projectId': 10,
 'projectOwner': {'id': 3, 'type': 'user', 'username': 'trial31_admin'},
 'updatedAt': '2021-02-02T21:07:57.479Z',
 'visibility': 'private'}

Wait for the model to deploy.
> is_deployed = False
> while is_deployed == False:
    model = cml.get_model({'id': str(
        new_model_details['id']), 'latestModelDeployment': True, 'latestModelBuild': True})
    if model['latestModelDeployment']['status'] == 'deployed':
        print("Model is deployed")
        break
    else:
        print("Deploying Model.....")
        time.sleep(10)

Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Deploying Model.....
Model is deployed
```

Part 5: Applications give data scientists a way to create ML web applications/dashboards and easily share them with other business stakeholders. Applications can range from single visualizations embedded in reports, to rich dashboard solutions such as Tableau. They can be interactive or non-interactive.

Applications stand alongside other existing forms of workloads in CML (sessions, jobs, experiments, models). Like all other workloads, applications must be created within the scope of a project. Each application is launched within its own isolated engine. Additionally, like models, engines launched for applications do not time out automatically. They will run as long as the web application needs to be accessible by any users and must be stopped manually when needed.

Select and highlight the contents of part 5, then Right click -> Run Line(s)

When deploying the CML Flask Application you will be provided with a URL to follow at the end of your session pane output. **Click to Open Application UI**

```

1 # Change the line in the flask/single_view.html file.
2 subprocess.call(["sed", "-i", 's/const\\accessKey.*/const accessKey/' +
3 access_key + '"/', "/home/cdsw/flask/single_view.html"])
4
5 # Change the model_id value in the 7a_model_operations.py and 7b_model_operations.py
6 subprocess.call(["sed", "-i", 's/model_id = .*/model_id = "' + model_id + '"/',
7 "/home/cdsw/7a_ml_ops_simulation.py"])
8 subprocess.call(["sed", "-i", 's/model_id = .*/model_id = "' + model_id + '"/',
9 "/home/cdsw/7b_ml_ops_visual.py"])
10
11 # Create Application
12 create_application_params = {
13     "name": "Explainer App",
14     "subdomain": run_time_suffix[:],
15     "description": "Explainer web application",
16     "type": "manual",
17     "script": "6_application.py", "environment": {},
18     "kernel": "python3", "cpu": 1, "memory": 2,
19     "nvidia_gpu": 0
20 }
21
22 new_application_details = cml.create_application(create_application_params)
23 application_url = new_application_details["url"]
24 application_id = new_application_details["id"]
25
26 # print("Application may need a few minutes to finish deploying. Open link below in about a minute..")
27 print("Application created, deploying at ", application_url)
28
29 # Wait for the application to deploy.
30 is_deployed = False
31 while is_deployed == False:
32     # Wait for the application to deploy.
33     app = cml.get_application(str(application_id), {})
34     if app["status"] == "running":
35         print("Application is deployed")
36         break
37     else:
38         print("Deploying Application.....")
39         time.sleep(10)
40
41 HTML("<a href='{0}'>Open Application UI</a>".format(application_url))
42
43 #####
44 #####
45 #####
46 #####
47 #####
48 #####
49
50 # This will run the model operations section that makes calls to the model
51 # metrics and track metric aggregations
52 exec(open("7a_ml_ops_simulation.py").read())
53
54 model_id + '"/', "/home/cdsw/7a_ml_ops_simulation.py"])
55
56 > subprocess.call(["sed", "-i", 's/model_id = .*/model_id = "' + model_id + '"/',
57 "/home/cdsw/7b_ml_ops_visual.py"])
58
59 # Create Application
60 create_application_params = {
61     "name": "Explainer App",
62     "subdomain": run_time_suffix[:],
63     "description": "Explainer web application",
64     "type": "manual",
65     "script": "6_application.py", "environment": {},
66     "kernel": "python3", "cpu": 1, "memory": 2,
67     "nvidia_gpu": 0
68 }
69
70 > new_application_details = cml.create_application(create_application_params)
71 > application_url = new_application_details["url"]
72 > application_id = new_application_details["id"]
73
74 print("Application may need a few minutes to finish deploying. Open link below in about a minute..")
75
76 > print("Application created, deploying at ", application_url)
77
78 Application created, deploying at https://02022021210720.ml-86b5138c-077.cdptrial.hs71-0qe
79 7.cloudersa.site
80
81 Wait for the application to deploy.
82
83 > is_deployed = False
84 > while is_deployed == False:
85     # Wait for the application to deploy.
86     app = cml.get_application(str(application_id), {})
87     if app["status"] == "running":
88         print("Application is deployed")
89         break
90     else:
91         print("Deploying Application.....")
92         time.sleep(10)
93
94 Deploying Application.....
95 Deploying Application.....
96 Application is deployed
97
98 > HTML("<a href='{0}'>Open Application UI</a>".format(application_url))
99
100 Open Application UI

```

Within the Flask Application UI you can experiment with some of the parameters that will run against the model to predict how likely a customer is to churn:

Single Prediction View

Churn Probability **0.737**

Contract	Month-to-month	0.12	Month-to-month	One year	Two year	
Dependents	No	0	No	Yes		
DeviceProtection	Yes	0	No	No internet service	Yes	
InternetService	Fiber optic	0.19	DSL	Fiber optic	No	
MonthlyCharges	97.85	-0.24	mean 64.80	min 18.25	max 118.75	<input type="text"/> <input type="button" value="Submit"/>
MultipleLines	Yes	0.06	No	No phone service	Yes	
OnlineBackup	No	0	No	No internet service	Yes	
OnlineSecurity	No	0.05	No	No internet service	Yes	
PaperlessBilling	Yes	0	No	Yes		
Partner	No	0	No	Yes		
PaymentMethod	Bank transfer (automatic)	0	Bank transfer (automatic)	Credit card (automatic)	Electronic check	Mailed check
PhoneService	Yes	0.04	No	Yes		
SeniorCitizen	No	0	No	Yes		
StreamingMovies	Yes	0.09	No	No internet service	Yes	
StreamingTV	Yes	0.07	No	No internet service	Yes	
TechSupport	No	0	No	No internet service	Yes	
TotalCharges	1105.4	-0.08	mean 2283.30	min 18.80	max 8684.80	<input type="text"/> <input type="button" value="Submit"/>
gender	Female	0	Female	Male		
tenure	11	0.11	mean 32.42	min 1.00	max 72.00	<input type="text"/> <input type="button" value="Submit"/>

Part 6: Run the last code snippet in the workbench to complete the project build script (8_build_project.py). This goes through a process of simulating a model that drifts over 1000 calls to the model. The file contains comments with details of how this is done.

Select and highlight the contents of part 6, then Right click -> Run Line(s)

```
284
285 #####
286 #####
287 #####
288 #####
289 #####
290
291 # This will run the model operations section that makes calls to t
292 # mertics and track metric aggregations
293
294 exec(open("7a_ml_ops_simulation.py").read())
295
```

Line 291, Column 1 ★ 295 Lines Python Spaces 2

Update
Update
Adding
Update
Update
Adding
Update
Update

You can also share the workbench session with other users if they would like to view results of the code. A URL can be shared out if users outside of CML would like to view code/results of work.

Click on share at the top of the Session pane

← Project >_ Terminal Access ✂ Clear ⚡ Interrupt ■ Stop Sessions ▾

telco_churn_session_1 Running

By [trial31_admin](#) — Python 3 Session — 1 vCPU / 2 GiB Memory — 33 minutes ago

Session Logs Spark UI Collapse Share Export PDF

Added 0 records
Added 50 records
Added 100 records
Added 150 records
Added 200 records
Added 250 records
Added 300 records
Added 350 records
Added 400 records
Added 450 records
Added 500 records
Added 550 records

🔒 These results are being shared.

Stop Sharing

☐ Hide code and text

Who can view:

☐ Any logged in user with the link

☒ Specific users/teams with the link ([Change...](#))

Currently shared with no one.

A full CML project should now be running with a Job, Model, and Application deployed! Go back to the project and take a look at the Model and Applications page.

Going to the **Model** page will show you the deployed model. Clicking on the Model will give various tabs of monitoring and statistics in addition to previous deployments of the same model.

On the **Applications** page we can see our running flask app we looked at previously.

Models and Applications will continue to run even if the workbench session is stopped or timeout occurs. These will run on engines as part of the CML workspace.

CLOUDERA

Machine Learning

All Projects

Overview

Sessions

Experiments

Models

Jobs

Applications

Files

Collaborators

Project Settings

trial31_admin / Telco_churn

Q Project quick find

+ trial31_admin

Telco_churn

0 Fork

New Session

2 running

Models

Model	Status	Replicas	CPU	Memory	Last Deployed	Actions
Model Explainer 02022021210720	Deployed	1 / 1	1	2.00 GIB	Feb 2, 2021, 03:08 PM	Stop

Jobs

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Model 02022021210720	1 / 0	00:12	Success	27 minutes ago	Run

Files

Download + New Upload

Name	Size	Last Modified
__pycache__	-	27 minutes ago
flask	-	26 minutes ago
images	-	38 minutes ago
models	-	38 minutes ago
raw	-	27 minutes ago
0_bootstrap.py	2.22 kiB	38 minutes ago
1_data_ingest.py	7.41 kiB	38 minutes ago
2_data_exploration.ipynb	602.63 kiB	38 minutes ago
3_model_building.ipynb	71.53 kiB	38 minutes ago
4_train_models.py	9.56 kiB	38 minutes ago
5_model_serve_explainer.py	8.10 kiB	38 minutes ago
6_application.py	7.74 kiB	38 minutes ago
7a_ml_ops_simulation.py	7.96 kiB	26 minutes ago