Lab: Azure ML & DevOps & Mlflow
Professor : Fahd KALLOUBI
Year : 2023/2024
Major : QFM

In this Lab, we will try to solve a real-world problem by leveraging azure DevOps, MLflow and Azure machine learning SDK.

Problem:
You work as a data scientist in a small team with three other data scientists for a shipping company based in the port of Turku in Finland. 90% of goods imported into Finland pass through cargo ships in the country's ports. For the transport of goods, weather conditions and logistics can sometimes be difficult at ports. Rainy conditions can distort operations and logistics at ports, which can affect supply chain operations. Predicting rainy conditions in advance helps optimize resources such as human, logistics and transportation for efficient supply chain operations at ports. From a business perspective, forecasting rainy conditions in advance allows ports to reduce operating costs by up to approximately 20% by enabling efficient planning and scheduling of human resources, logistics and shipping resources. transportation for supply chain operations.

Task:
As a data scientist, you are responsible for developing an ML-based solution to predict weather conditions 4 hours in advance at the port of Turku in Finland. This will allow the port to optimize its resources, resulting in savings of up to 20%. To start, you have a set of historical weather data covering a period of 10 years from the port of Turku. Your task is to create an ML solution focused on continuous learning to optimize operations at the Port of Turku.

To solve this problem, we will first use Microsoft Azure as one of the most used cloud services and MLFlow. This way we will learn how to couple an open source tool with a cloud service.

1. Create a free Azure account for students: https://azure.microsoft.com/en-us/free/students/

1. Setting up resources and tools

1.1 MLflow

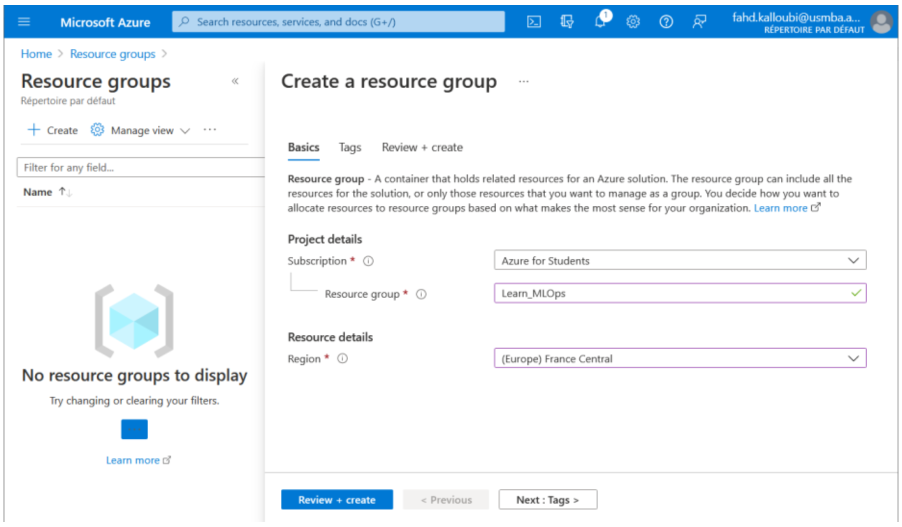
On your terminal, install MLflow: pip install mlflow Test your installation by running the command (mlflow ui), you can access MLflow UI on http://localhost:5000 .

1.2 Azure Machine learning

- **Create your resource group**

A resource group is a collection of resources for an Azure solution. Creating a resource group makes it easier to access and manage a solution. You will create your own resource group:

- Open the Azure portal
- While on the main page, click on “group resource” then on “create”


Microsoft Azure | Search resources, services, and docs (G+/) | fahd.kalloubi@usmba.a... RÉPERTOIRE PAR DÉFAUT

Home > Resource groups >

Resource groups
Répertoire par défaut

Create | Manage view

Filter for any field...

Name

No resource groups to display
Try changing or clearing your filters.

Learn more

Create a resource group

Basics | Tags | Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription *	Azure for Students
Resource group *	Learn_MLOps

Resource details

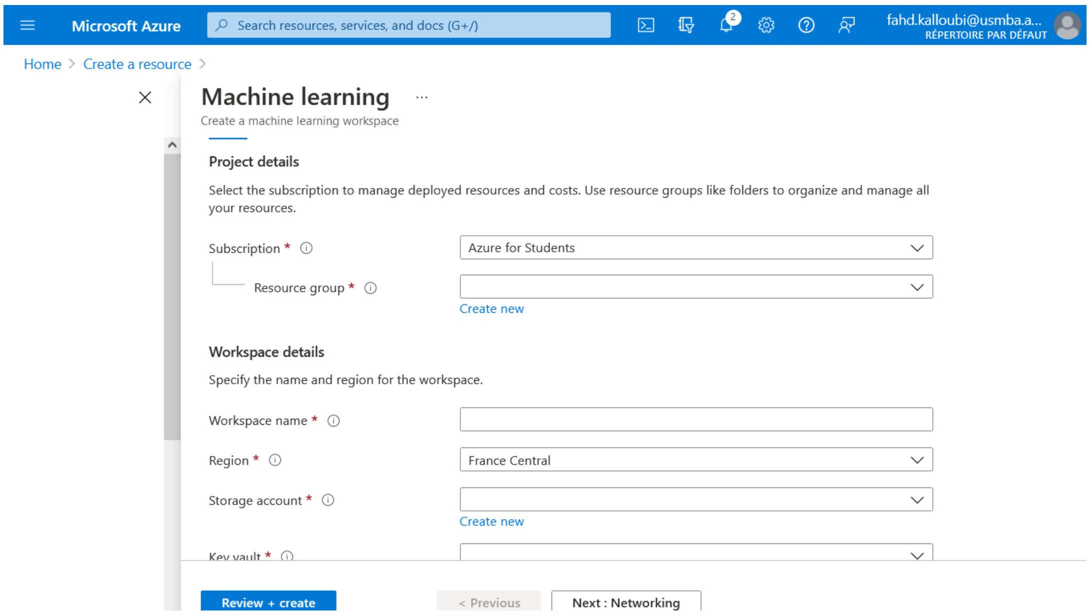
Region *	(Europe) France Central

Review + create | < Previous | Next : Tags >

- ➢ Choose a name for your resource group (Learn_MLOps is recommended)
- ➢ Choose a region close to you to have a good price and optimal performance (For us it will be France Central)
- ➢ Then click on “Review + Create”, Azure will validate the request
- ➢ Finally, you must click on “Create”.

The new “Resource group” will be listed in the list of resource groups. From now on, you can manage all services related to your ML solution in this group.

➢ Create your Azure Machine Learning workspace


Microsoft Azure | Search resources, services, and docs (G+/) | fahd.kalloubi@usmba.a... RÉPERTOIRE PAR DÉFAUT

Home > Create a resource >

Machine learning
Create a machine learning workspace

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure for Students
Resource group *	Create new

Workspace details

Specify the name and region for the workspace.

Workspace name *	
Region *	France Central
Storage account *	Create new
Key vault *	

Review + create | < Previous | Next : Networking

An ML workspace is a central hub for tracking and managing training, deployment and monitoring experiments. In order to create an Azure machine learning workspace, go to the Azure portal (i.e., Home) and click on “Create a resource” then on the categories menu choose “AI + machine learning” and click on “machine learning”.

Then choose your Resource group (i.e., Learn_MLOps) and name your workspace (for example “MLOps_WS”).

Finally, click on “review + create” and then “create”.

After creating this workspace, the platform will deploy all the resources that this service needs such as: Blob storage, key Vault, application insights. These resources will be consumed or used via the workspace and the SDK (more details: https://docs.microsoft.com/en-us/azure/machine-learning/how-to-manage-workspace)

➢ **Installing Azure machine learning SDK**

From your terminal (i.e., anaconda), install Azure Machine Learning SDK which will be used constantly in our code to orchestrate our experiments.

```
pip install --upgrade azureml-sdk
```

For more information visit: https://docs.microsoft.com/en-us/python/api/overview/azure/ml/install?view=azure-ml-py

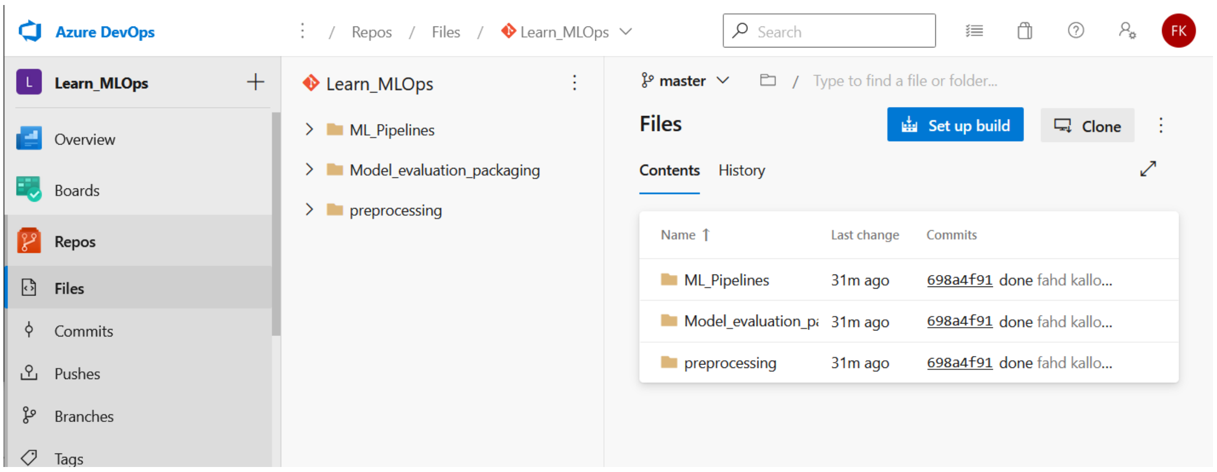
1.3 Azure DevOps

All source code and CI/CD operations will be managed and orchestrated using Azure DevOps. The code we manage in the repository in Azure DevOps will be used to train, deploy, and monitor ML models enabled by CI/CD pipelines.

- ➢ In the search bar type “Azure DevOps” and create a free account, your account will be automatically created following your registration and you will be directed to the project creation page.
- ➢ Name your project “Learn_MLOps” (put it private or public)
- ➢ Then go to the “Repos” section then to the “import a repository” section and click on the “Import” button.
- ➢ In the “clone URL” area paste this link https://github.com/FahdKalloubi1/MLOps_SDAD

After the import, the contents of the github repository will be displayed.

2. Data preprocessing



Name ↑	Last change	Commits
ML_Pipelines	31m ago	698a4f91 done fahd kallo...
Model_evaluation_pa	31m ago	698a4f91 done fahd kallo...
preprocessing	31m ago	698a4f91 done fahd kallo...

We will start by accessing the data to preprocess it for training the model. To do this, clone the repository that you imported into Azure DevOps (the button at the top right allows you to copy the link), on the Git command prompt type the command below:

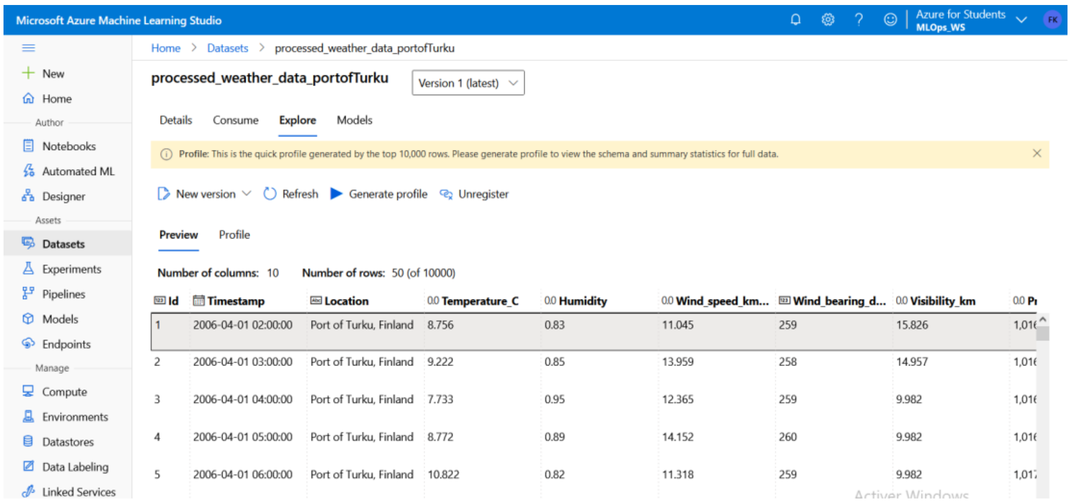
```
git clone
https://xxxxxx@dev.azure.com/xxxxxx/Learn_MLOps/_git/Learn_MLOps
```

Then navigate to the cloned folder using jupyter notebook.

Open the "Dataprocessing_register.ipynb" notebook locally and run it.

In the registration section in the preprocessed dataset workspace, replace the registration ID with yours.

Finally, to view the dataset saved on the workspace, while on the "MLOps_WS" workspace page, click on the "Launch studio" button and then on "Data" in the menu on the left and then on "explore".



processed_weather_data_portofTurku

Profile: This is the quick profile generated by the top 10,000 rows. Please generate profile to view the schema and summary statistics for full data.

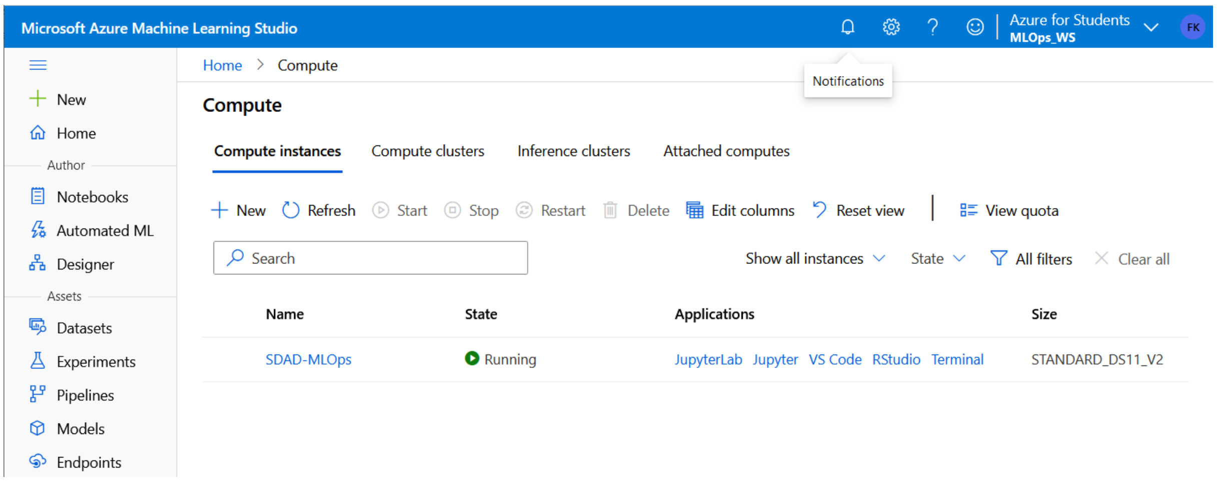
Number of columns: 10 Number of rows: 50 (of 10000)

Id	Timestamp	Location	Temperature_C	Humidity	Wind_speed_km...	Wind_bearing_d...	Visibility_km	Pr
1	2006-04-01 02:00:00	Port of Turku, Finland	8.756	0.83	11.045	259	15.826	1,01
2	2006-04-01 03:00:00	Port of Turku, Finland	9.222	0.85	13.959	258	14.957	1,01
3	2006-04-01 04:00:00	Port of Turku, Finland	7.733	0.95	12.365	259	9.982	1,01
4	2006-04-01 05:00:00	Port of Turku, Finland	8.772	0.89	14.152	260	9.982	1,01
5	2006-04-01 06:00:00	Port of Turku, Finland	10.822	0.82	11.318	259	9.982	1,01

3. Pipeline construction

We processed the data on our local computer. For training ML models and implementing the pipeline, we use computing resources provisioned in the cloud (Microsoft Azure). In the following, we will create an Azure compute instance.

a. While still on Microsoft Azure machine learning Studio, click on "compute" and then "+ New"
b. Give a name to your instance and choose an instance type appropriate to our problem (I chose "Standard_DS11_v2" a 2 core with 14 GB of Ram)
c. Click "create", now your compute instance is ready for cloud computing



Name	State	Applications	Size
SDAD-MLOps	Running	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS11_V2

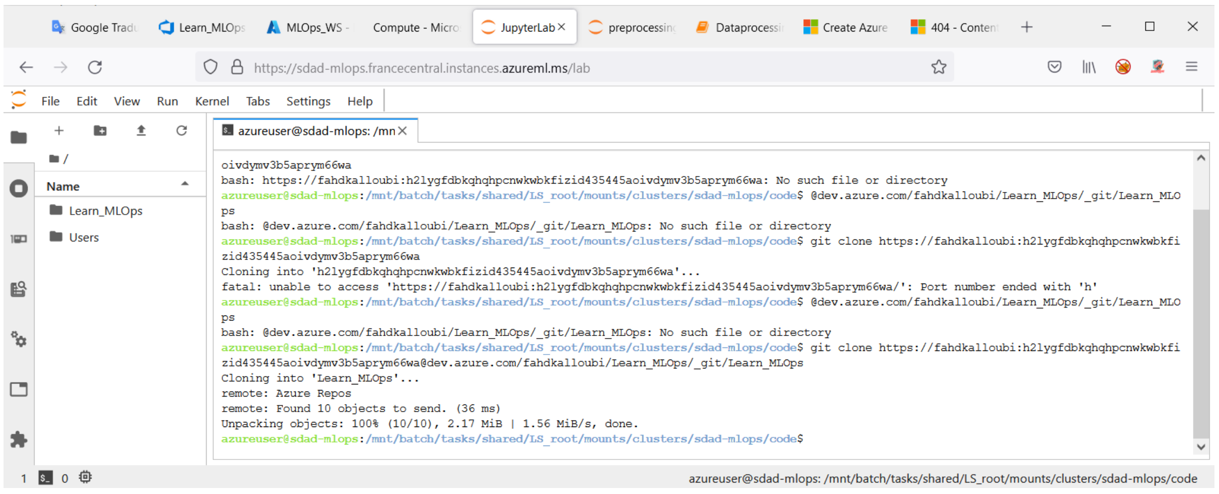
Once it is provisioned, select the JupyterLab option. JupyterLab is an open source web user interface. It comes with features like a text editor, code editor, terminal, and custom components built in an extensible manner. We will use it as a programming interface connected to the provisioned compute to train the ML models.

Now we will start with the practical implementation of the ML pipeline. Follow these steps to implement the ML pipeline:

1. To begin our implementation, clone the repository you imported into Azure DevOps. To do this, go to Azure DevOps then click on “Repos” then “files” and finally on “clone”
2. Then click on “Generate Git Credentials”, a password will be created.
3. Copy the generated password and add it to the repository link by adding the password just after the first username separated by: before the @ character. Then it is possible to use the following git clone command without getting permission errors:

```
git clone
https://xxxxxx:password@dev.azure.com/xxxxxx/Learn_MLOps/_git/Learn_MLOps
```

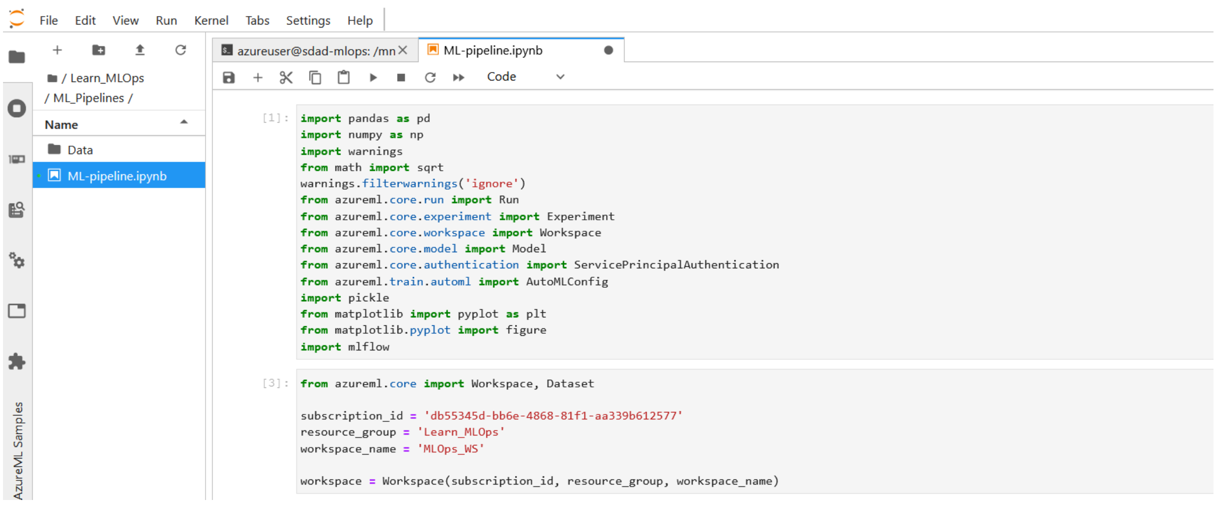
4. Once you are on JupyterLab, go to the terminal to clone the respository to the Azure instance. To access the terminal, you need to select the Terminal option from the Launcher tab. Run the command above to clone the repository.



```
oivdymv3b5aprym66wa
bash: https://fahdkalloubi:h2lygfdbkqhqhpcnwkwbkfizid435445aoivdymv3b5aprym66wa: No such file or directory
azureuser@sdad-mlops:/mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code$ @dev.azure.com/fahdkalloubi/Learn_MLOps/_git/Learn_MLO
ps
bash: @dev.azure.com/fahdkalloubi/Learn_MLOps/_git/Learn_MLOps: No such file or directory
azureuser@sdad-mlops:/mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code$ git clone https://fahdkalloubi:h2lygfdbkqhqhpcnwkwbkfi
zid435445aoivdymv3b5aprym66wa
Cloning into 'h2lygfdbkqhqhpcnwkwbkfizid435445aoivdymv3b5aprym66wa'...
fatal: unable to access 'https://fahdkalloubi:h2lygfdbkqhqhpcnwkwbkfizid435445aoivdymv3b5aprym66wa/': Port number ended with 'h'
azureuser@sdad-mlops:/mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code$ @dev.azure.com/fahdkalloubi/Learn_MLOps/_git/Learn_MLO
ps
bash: @dev.azure.com/fahdkalloubi/Learn_MLOps/_git/Learn_MLOps: No such file or directory
azureuser@sdad-mlops:/mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code$ git clone https://fahdkalloubi:h2lygfdbkqhqhpcnwkwbkfi
zid435445aoivdymv3b5aprym66wa@dev.azure.com/fahdkalloubi/Learn_MLOps/_git/Learn_MLOps
Cloning into 'Learn_MLOps'...
remote: Azure Repos
remote: Found 10 objects to send. (36 ms)
Unpacking objects: 100% (10/10), 2.17 MiB | 1.56 MiB/s, done.
azureuser@sdad-mlops:/mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code$
```

https://sdad-mlops.francecentral.instances.azureml.ms/lab
File Edit View Run Kernel Tabs Settings Help
Name
Learn_MLOps
Users
azureuser@sdad-mlops: /mnt/batch/tasks/shared/LS_root/mounts/clusters/sdad-mlops/code

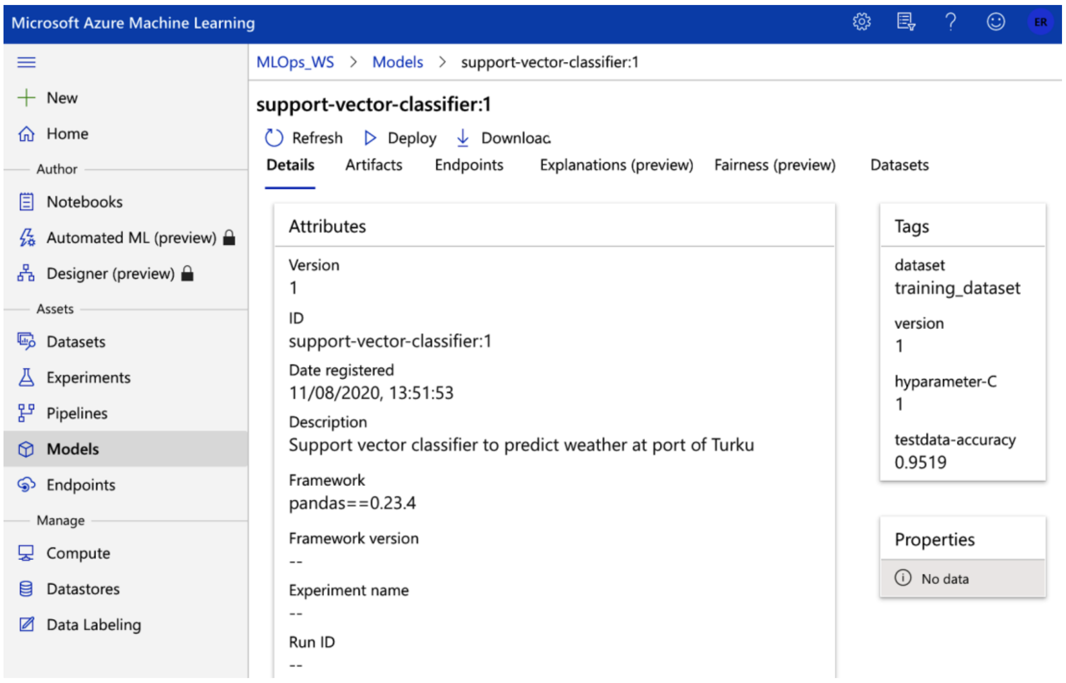
5. Now that it is cloned, click on "Learn_MLOps" and go to "ML_Pipelines" and open the notebook "ML_pipeline.ipynb" and then run it. It is recommended to run it locally with your own installation and then run it in the cloud.



```
[1]: import pandas as pd
     import numpy as np
     import warnings
     from math import sqrt
     warnings.filterwarnings('ignore')
     from azureml.core.run import Run
     from azureml.core.experiment import Experiment
     from azureml.core.workspace import Workspace
     from azureml.core.model import Model
     from azureml.core.authentication import ServicePrincipalAuthentication
     from azureml.train.automl import AutoMLConfig
     import pickle
     from matplotlib import pyplot as plt
     from matplotlib.pyplot import figure
     import mlflow

[3]: from azureml.core import Workspace, Dataset

     subscription_id = 'db55345d-bb6e-4868-81f1-aa339b612577'
     resource_group = 'Learn_MLOps'
     workspace_name = 'MLOps_WS'

     workspace = Workspace(subscription_id, resource_group, workspace_name)
```

File Edit View Run Kernel Tabs Settings Help
/ Learn_MLOps / ML_Pipelines /
Name
Data
ML-pipeline.ipynb
Code
AzureML Samples

6. After running the notebook, you can see the saved model and the artifacts (parameters, version, etc.) on Azure machine learning studio by clicking on "Models".



support-vector-classifier:1

Attributes

Version
1

ID
support-vector-classifier:1

Date registered
11/08/2020, 13:51:53

Description
Support vector classifier to predict weather at port of Turku

Framework
pandas==0.23.4

Framework version
--

Experiment name
--

Run ID
--

Tags

dataset
training_dataset

version
1

hyperparameter-C
1

testdata-accuracy
0.9519

Properties

No data

4. Evaluating and packaging our models

While on JupyterLab, go to the "Model_evaluation_packaging" folder and open the "model_evaluation_packaging.ipynb" notebook and then run it.

The steps can be summarized as follows:

a. We must connect to the workspace and import the artifacts

We import the required packages, connect to the ML workspace using the Workspace() function, and then download the serialized scaler and model to make predictions. Scaler will be used to scale the input data into the same data scale used for training the model. The model file is serialized in ONNX format. Scaler and Model files are imported using the Model() function.

b. We load the artifacts for inference

We open and load the Scaler and Model files into variables that can be used for model inference. The scaler is read and loaded into a variable using pickle, and the ONNX runtime is used to load the ONNX file using InferenceSession() to make predictions.