

Azure Machine Learning SDK

By - Jayabharathi Hari
Deakin University
✉: s224643593@deakin.edu.au

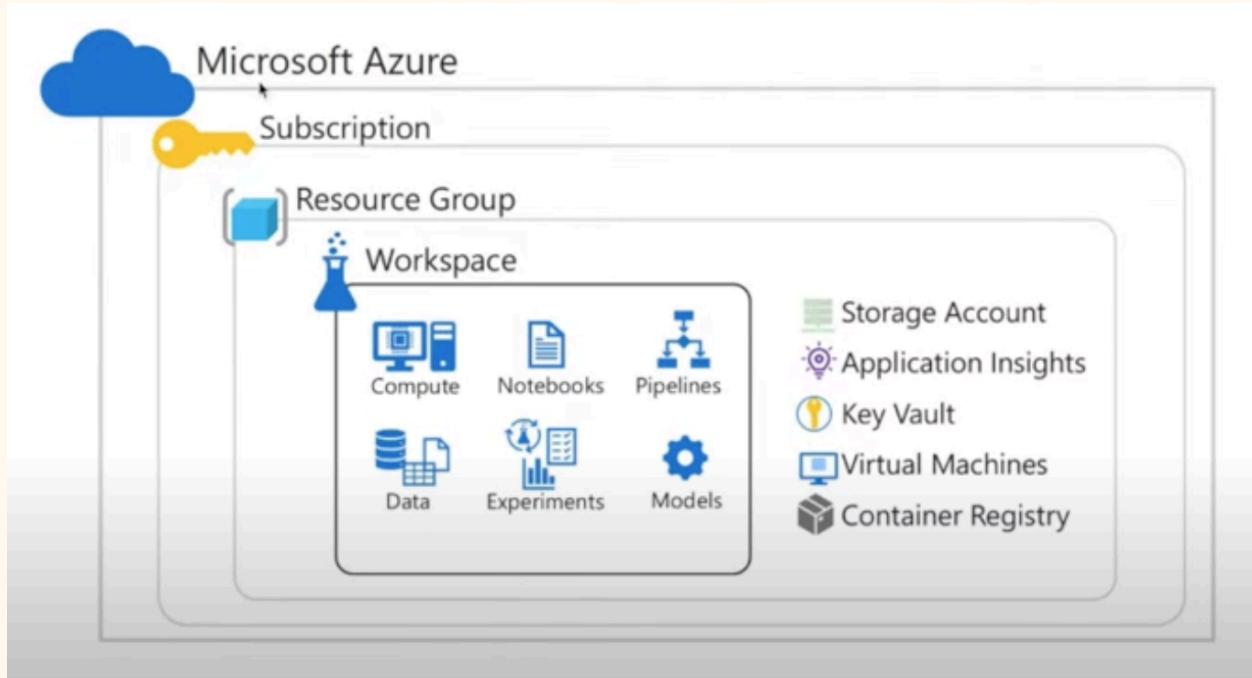
About Azure Machine Learning SDK for Python.....	1
1.Workspace:.....	2
2.Steps to create a workspace for machine learning using the Azure Portal.....	2
3.Lets discuss about Computing Instance & Compute Cluster:.....	4
Let's jump into Hands on - Azure Machine learning Python SDK.....	6
1.Set Up Azure Machine Learning.....	6
● Installing Azure Machine Learning SDK.....	6
● Mounting google to gdrive since I'm using google colab.....	6
● Importing the dependencies.....	6
● Loading the config file.....	7
● Creating the workspace.....	7
2.Train and Register the Model in Azure.....	8
● Training and Pickling the Model.....	8
● Registering the Pickled Model in Azure.....	12
3.Deploy the Registered Model.....	14
1.To deploy the Model , prerequisite files:.....	14
● Environment (.yaml) file.....	14
● Entry Script (score.py) file.....	14
2. Deploying Model in Azure.....	15
● Deployment using python code.....	15
● Deployment via Azure Workspace.....	15
3. Model - Testing/Prediction.....	19
● Testing with python code and scoring URI.....	19
● Testing in Workspace - Azure Portal.....	20

About Azure Machine Learning SDK for Python

Azure Machine Learning provides software development kits(SDK) for Python and which can be used to create, manage and utilize assets in an Azure Machine Learning Workspace by data scientists and AI developers to build and run machine learning projects/workflow((MLops)) on Microsoft's Azure cloud platform. This SDK works in environments like Jupyter Notebooks, Google Colab and Python IDEs.

It provides an intuitive interface for experimentation, model training, deployment, and monitoring within a Python environment. With features like Automated Machine Learning (AutoML) and seamless integration with Azure services, the SDK streamlines the end-to-end machine learning workflow, empowering users to create scalable and efficient machine learning solutions.

1. Workspace:



Picture Source: Microsoft Azure website.

Imagine a cloud storage unit for your machine learning projects. That's a Microsoft Machine Learning workspace. It lives within Azure and holds everything you need: code, data, models. Think of it as a central hub. Data scientists can use this workspace to collaborate, run experiments (train models), and even deploy finished models to use. It all happens securely in the Microsoft cloud.

2. Steps to create a workspace for machine learning using the Azure Portal.

1. Sign in to Azure portal and search for "Machine Learning".
2. Click "Create" and choose a unique name for your workspace.
3. Select the subscription and resource group (create a new one or choose existing).
4. Pick a region for your workspace and optionally configure application insights and container registry.
5. Hit "Create" to provision your workspace in Azure.

Azure Machine Learning Create a machine learning workspace

Validation passed

Basics

- Subscription: Azure for Students
- Resource group: s224643593-rg
- Region: East US
- Name: s224643593-ws
- Storage account: (new) s224643593ws2971611799
- Key vault: (new) s224643593ws2541248350
- Application insights: (new) s224643593ws0617692301
- Container registry: None

Networking

- Connectivity method: Enable public access from all networks
- Network isolation: Public

Encryption

- Encryption type: Microsoft-managed keys

Identity

- Identity type: System assigned
- Enable HBI Flag: Disabled

Create < Previous Next > Download a template for automation

We would be able to view the below kind of overview in the resource group once workspace got created, then click on “Go to resource” to launch Machine Learning Studio.

Microsoft.MachineLearningServices | Overview

Your deployment is complete

Deployment name : Microsoft.MachineLearningServices
Subscription : Azure for Students
Resource group : s224643593-rg
Start time : 02/04/2024, 21:40:24
Correlation ID : 367fcaca-2ab6-4a6f-9b59-95307cb2ba90

Deployment details

Resource	Type	Status	Operation details
s224643593-ws	Azure Machine Learning worksp	OK	Operation details
s224643593ws061795	Application Insights	OK	Operation details
s224643593ws297161	Storage account	OK	Operation details
s224643593ws254124	Key vault	OK	Operation details
s224643593ws98423	Log Analytics workspace	OK	Operation details

Next steps

Go to resource

s224643593-ws Azure Machine Learning workspace

Overview

Search Download config.json Delete

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Networking

Properties

Locks

Monitoring

Automation

Support + troubleshooting

Usage + quotas

Work with your models in Azure Machine Learning Studio

The Azure Machine Learning Studio is a web app where you can build, train, test, and deploy ML models. Launch it now to start exploring, or learn more about the Azure Machine Learning studio.

Launch studio

3.Lets discuss about Computing Instance & Compute Cluster:

<u>Computing Instance:</u>	<u>Compute Cluster:</u>
Imagine a powerful, pre-configured computer in the cloud. This single-user instance is ideal for development and testing tasks. It can run multiple jobs in parallel and even handle some training, but lacks the muscle for large-scale projects.	Think of a team of computers working together. A cluster offers multiple virtual machines (nodes) that can be scaled up or down based on your workload. This allows you to tackle demanding training jobs by distributing the work across the cluster for faster results.
<p>Use a Compute Instance:</p> <p>Development and Testing:</p> <ul style="list-style-type: none"> Perfect for experimenting with code, exploring data, and testing small-scale models. Its single-user environment keeps things organized. <p>Quick Iterations:</p> <ul style="list-style-type: none"> Rapidly prototype and iterate on your machine learning models without waiting for clusters to spin up. <p>Limited Training Needs:</p> <ul style="list-style-type: none"> If your training jobs are relatively small and don't require significant processing power, a compute instance can handle them efficiently. <p>Cost-Effectiveness:</p> <ul style="list-style-type: none"> For smaller projects or those on a tight budget, a compute instance offers a more cost-friendly option compared to a cluster. 	<p>Use a Compute Cluster:</p> <p>Large Datasets and Models:</p> <ul style="list-style-type: none"> When dealing with massive datasets or complex models that require significant processing power, a cluster distributes the workload for faster training. <p>Scalability:</p> <ul style="list-style-type: none"> Need to ramp up processing power for demanding jobs? A cluster allows you to easily scale the number of virtual machines to meet your needs. <p>Parallel Processing:</p> <ul style="list-style-type: none"> Tasks that benefit from parallelization, like hyperparameter tuning or running multiple training jobs simultaneously, run much faster on a cluster. <p>Production Deployments:</p> <ul style="list-style-type: none"> When deploying models for real-world use cases, a cluster provides the reliability and scalability required for handling production workload
<p>Creating a Compute Instance:</p> <p>Access Azure Machine Learning Studio:</p> <p>Sign in to Azure and navigate to your Machine</p>	<p>Creating a Compute Cluster:</p> <p>Navigate to Compute:</p> <p>Follow steps 1 and 2 from creating a compute</p>

Learning workspace.

Go to Compute:

Under the "Manage" section, select "Compute".

Create a New Instance:

Click the "+ New" button and choose "Compute instance".

Configure Settings:

Fill out the details like name, virtual machine size (consider resource requirements for your tasks), and optionally enable a managed identity for access control.

Review and Create:

Review your configuration and hit "Create" to provision the instance.

instance.

Choose Cluster:

Click the "+ New" button and select "Compute cluster".

Configure Cluster Details:

Here you'll define cluster properties like:

- VM Size:** Choose the virtual machine size for each node in the cluster (based on processing needs).
- Number of Nodes:** Specify the desired number of virtual machines to work together in the cluster (scales workload).
- Other Settings:** Configure additional options like initial node count (minimum number of VMs running) and autoscale settings (automatic scaling based on workload).

Review and Create:

After customizing your cluster settings, review the configuration and click "Create" to provision the cluster.

The image shows two side-by-side screenshots of the Azure AI | Machine Learning Studio interface.

Left Screenshot: Create compute instance

This screen shows the configuration for a single compute instance. It includes sections for Required settings (Compute name: s2246435931, Virtual machine type: Standard_DS3_v4, 4 cores, 32GB RAM, 150GB storage), Scheduling (Auto shutdown: After 60 minutes of inactivity), Security (Enable SSH: no, Enable managed identity: no), Applications (Post (formerly RStudio) is no longer installed by default on compute instances. Instead, add it as a custom application), and Tags. Buttons for Create, Back, and Cancel are at the bottom.

Right Screenshot: Create compute cluster

This screen shows the configuration for a compute cluster. It includes sections for Configure Settings (Virtual Machine, Advanced Settings), Compute name (Standard_DS3_v2), Category (General purpose), Cores (4), Available quota (6 cores), RAM (14 GB), Storage (28 GB), Cost/Node (\$0.29/hr), Minimum number of nodes (0), Maximum number of nodes (1), Idle seconds before scale down (120), and Enable SSH access (unchecked). Buttons for Back, Create, and Cancel are at the bottom.

Let's jump into Hands on - Azure Machine learning Python SDK

1. [Set Up Azure Machine Learning](#)
2. [Train and Register the Model in Azure](#)
3. [Deploy the Registered Model](#)
4. [Model - Testing/Prediction](#)

1. Set Up Azure Machine Learning

- Installing Azure Machine Learning SDK.

```
!pip install azureml-sdk
```

- Mounting google to gdrive since I'm using google colab.

```
1 from google.colab import drive
2 drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

- Importing the dependencies.

Importing the dependencies

```
[ ] 1 import os
2 import json
3 import requests
4
5 from azureml.core import Workspace
6 from azureml.core.model import Model
7 from azureml.core.environment import Environment
8 from azureml.core.conda_dependencies import CondaDependencies
9 from azureml.core.model import InferenceConfig
10 from azureml.core.webservice import AciWebservice, Webservice
```

- Loading the config file

Config file has subscription and resource group details. Declared workspace and region info for workspace creation.

```

▶ 1 # laoding the configuration file
 2 config_file_path = "/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure/config.json"
 3
 4 # Read JSON data into a dictionary
 5 with open(config_file_path, 'r') as file:
 6     data = json.load(file)
 7
 8 subscription_id = data["subscription_id"]
 9 resource_group = data["resource_group"]
10 workspace_name = data["workspace_name"]
11 region = data["region"]

[ ] 1 print(resource_group)
 2 print(workspace_name)
 3 print(region)

ML-w-Azure-rg
ML-w-Azure-ws
East US

```

- Creating the workspace.

```

1 # Create a workspace
2
3 ws = Workspace.create(name=workspace_name,
4                         subscription_id=subscription_id,
5                         resource_group=resource_group,
6                         location=region)
7
8 print(f'Workspace {workspace_name} created')

Performing interactive authentication. Please follow the instructions on the terminal.
WARNING:azureml._vendor.azure_cli_core.auth.identity:To sign in, use a web browser to open the page
Interactive authentication successfully completed.
Cleaning up past default Resource Group Deployments on the subscription to avoid limit of 10
Deleting past Resource Group Deployment with name: DeployResourceGroup-47aed8d534
Deploying StorageAccount with name mlwazurorestoragefaaa0ad5e.
Deploying KeyVault with name mlwazurekeyvault03a24f1b.
Deploying Workspace with name ML-w-Azure-ws.
Deploying AppInsights with name mlwazureinsightsb63af21.
Deployed AppInsights with name mlwazureinsightsb63af21. Took 36.13 seconds.
Deployed Workspace with name ML-w-Azure-ws. Took 35.13 seconds.
Workspace ML-w-Azure-ws created

```

- We can verify in Azure portal that all resources were created.

The screenshot shows the Azure portal interface for the 'ML-w-Azure-rg' resource group. The left sidebar lists navigation options like Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings (Deployments, Security, Deployment stacks, Policies, Properties, Locks), and Cost Management (Cost analysis, Cost alerts (preview)). The main content area is titled 'Essentials' under 'Resources'. It shows a table of 8 records with columns for Name, Type, and Location. The records include:

Name	Type	Location
4a7b431ccd334ba7b1c0691963b63757	Container registry	East US
Application Insights Smart Detection	Action group	Global
deploymodel-HEN7jPNp0UxwGkZY7Y3Vw	Container instances	East US
Failure Anomalies - mlazureinsightsb63af21	Smart detector alert rule	Global
ML-w-Azure-ws	Azure Machine Learning workspace	East US
mlazureinsightsb63af21	Application Insights	East US
mlazurekeyvault03a24fb	Key vault	East US

2. Train and Register the Model in Azure

- Training and Pickling the Model

We need the best model which was trained and the same has to be saved in the pickled format for registering the model in azure. For that I'm using a diabetes dataset.

About the Dataset:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Column Level Details:

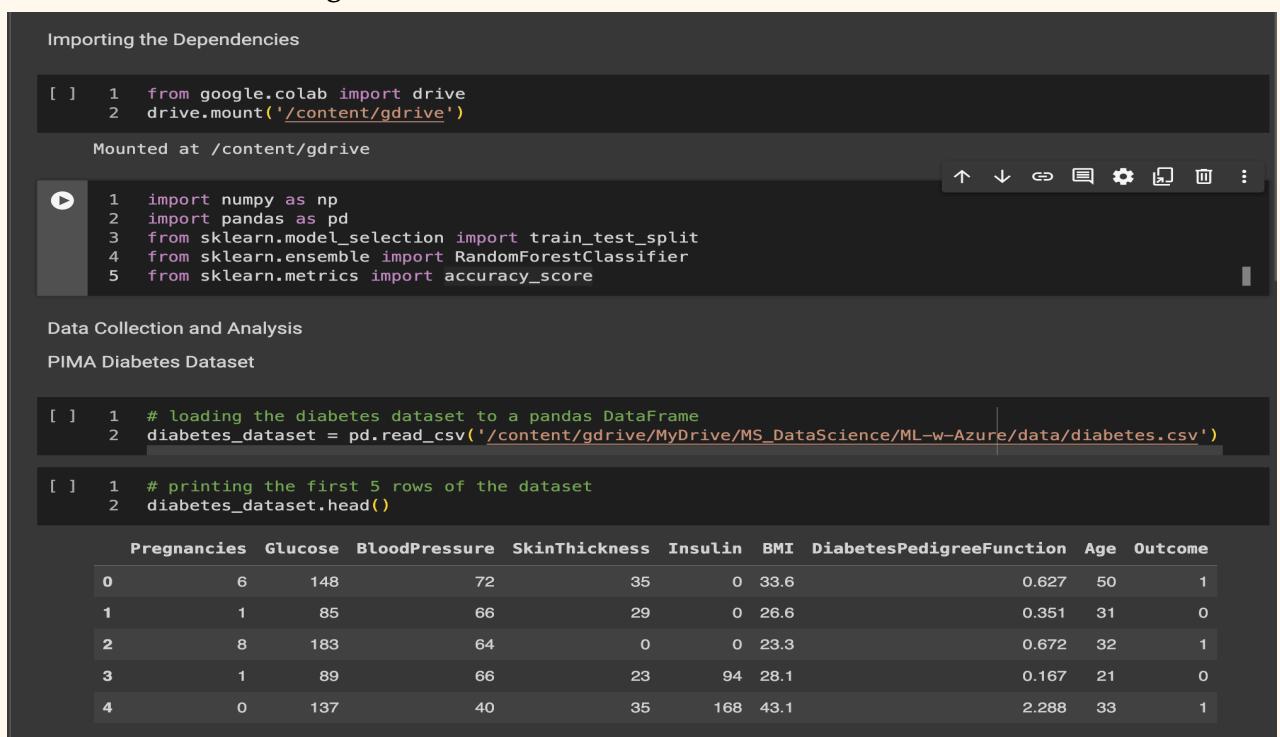
1. Pregnancies: Number of times pregnant
2. Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. BloodPressure: Diastolic blood pressure (mm Hg)
4. SkinThickness: Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Age (years)

9. Outcome: Class variable (0 or 1)

Training the Model:

1. Loaded the dataset from gdrive
2. Did EDA and found its an imbalance dataset because class "0"(Non-Diabetic) is 500 in count and class "1"(Diabetic) is 268 in count. There are no duplicate rows.
3. Did Data Pre-Processing - Undersampling to get evenly distributed class to 268 in nos.
4. Did Train Test Split
5. Trained the model with RandomForestClassifier
6. Accuracy Score
7. Saved the Model in Pickle format.

Loaded the dataset from gdrive



```

Importing the Dependencies

[ ] 1 from google.colab import drive
2 drive.mount('/content/gdrive')

Mounted at /content/gdrive

[ ] 1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.metrics import accuracy_score

Data Collection and Analysis

PIMA Diabetes Dataset

[ ] 1 # loading the diabetes dataset to a pandas DataFrame
2 diabetes_dataset = pd.read_csv('/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure/data/diabetes.csv')

[ ] 1 # printing the first 5 rows of the dataset
2 diabetes_dataset.head()

   Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
0          6      148           72            35     0  33.6        0.627    50       1
1          1       85            66            29     0  26.6        0.351    31       0
2          8      183           64            0     0  23.3        0.672    32       1
3          1       89            66            23     94  28.1        0.167    21       0
4          0      137           40            35    168  43.1        2.288    33       1

```

EDA and Data Pre-Processing:

```
[ ] 1 diabetes_dataset['Outcome'].value_counts()

  Outcome
  0      500
  1      268
  Name: count, dtype: int64

0 --> Non-Diabetic
1 --> Diabetic

Undersample to get evenly distributed class

▶ 1 df_class_0 = diabetes_dataset[diabetes_dataset["Outcome"]==0]
  2 df_class_1 = diabetes_dataset[diabetes_dataset["Outcome"]==1]
  3
  4 print("After undersampling", len(df_class_0), len(df_class_1))
  5
  6 df_class_0 = df_class_0.sample(len(df_class_1))
  7
  8 print("After undersampling" , len(df_class_0), len(df_class_1))
  9
 10 diabetes_dataset = pd.concat([df_class_0, df_class_1])
➡ After undersampling 500 268
  After undersampling 268 268

[ ] 1 diabetes_dataset['Outcome'].value_counts()

  Outcome
  0      268
  1      268
  Name: count, dtype: int64
```

Train Test Split

Train Test Split

```
[ ] 1 X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)

[ ] 1 print(X.shape, X_train.shape, X_test.shape)
  (536, 8) (428, 8) (108, 8)
```

Model Training

Training the Model

```
[ ] 1 classifier = RandomForestClassifier(n_estimators=100, random_state=0)
[ ] 2 # classifier = svm.SVC(kernel='linear')
[ ] 3
[ ] 4 # training the support vector Machine Classifier
[ ] 5 classifier.fit(X_train, Y_train)
```

```
▼      RandomForestClassifier
RandomForestClassifier(random_state=0)
```

Accuracy Score

Accuracy Score

```
[ ] 1 # accuracy score on the training data
[ ] 2 X_train_prediction = classifier.predict(X_train)
[ ] 3 training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

[ ] 1 print('Accuracy score of the training data : ', training_data_accuracy)
Accuracy score of the training data :  1.0

[ ] 1 # accuracy score on the test data
[ ] 2 X_test_prediction = classifier.predict(X_test)
[ ] 3 test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

[ ] 1 print('Accuracy score of the test data : ', test_data_accuracy)
Accuracy score of the test data :  0.7592592592592593
```

Picking the Model.

Saving the trained model

```

1 import pickle
2
3 # Specify the path where you want to save the pickle file
4 SAVE_PATH = '/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure' # Adjust the path as needed
5
6 # Save the object as a pickle file
7 with open(SAVE_PATH + '/diabetes_model.pkl', 'wb') as outfile:
8     pickle.dump(classifier, outfile)

[ ] 1 # filename = 'diabetes_model.pkl'
2 # pickle.dump(classifier, open(filename, 'wb'))
3 # loading the saved model
4 filename = '/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure/diabetes_model.pkl'
5 loaded_model = pickle.load(open(filename, 'rb'))

[ ] 1 input_data = (5,166,72,19,175,25.8,0.587,51)
2
3 # changing the input_data to numpy array
4 input_data_as_numpy_array = np.asarray(input_data)
5
6 # reshape the array as we are predicting for one instance
7 input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
8
9 prediction = loaded_model.predict(input_data_reshaped)
10 print(prediction)
11
12 if (prediction[0] == 0):
13     print('The person is not diabetic')
14 else:
15     print('The person is diabetic')

[1]
The person is diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifi
warnings.warn(

```

- Registering the Pickled Model in Azure.

```

[ ] 1 # Specify the path to your model file
2 import pickle
3 model_path = '/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure/diabetes_model.pkl'
4 # Load the pickled model
5 loaded_model = pickle.load(open(model_path, 'rb'))

[ ] 1 model_name='diabetes_prediction_model'

[ ] 1 # Register the model in Azure Machine Learning
2 registered_model = Model.register(model_path=model_path, model_name=model_name, workspace=ws)

Registering model diabetes_prediction_model

```

We can see registered model details in the workspace in the name of “diabetes_prediction_model”, also we can see the pickled file

The screenshot shows the Azure AI | Machine Learning Studio interface. The left sidebar navigation includes 'All workspaces', 'Home', 'Model catalog', 'Authoring' (with 'Notebooks', 'Automated ML', 'Designer', 'Prompt flow'), 'Assets' (with 'Data', 'Jobs', 'Components', 'Pipelines', 'Environments'), 'Models' (selected), and 'Endpoints'. The main content area displays the details for the 'diabetes_prediction_model:1' model. The 'Details' tab is selected, showing the following attributes:

- Name: diabetes_prediction_model
- Version: 1
- Created on: Apr 12, 2024 1:10 PM
- Created by: JAYABHARATHI HARI
- Type: CUSTOM
- Created by job: --
- Asset ID: azureml://locations/eastus/workspaces/4a7b431c-cd33-4ba7-b1c0-691963b63757/models/diabetes_prediction_model/versions/1

On the right side, there are sections for 'Tags' (No tags), 'Properties' (No properties), and 'Description' (Click edit icon to add a description). The top right corner shows the user profile 'JAYABHARATHI H...' and the workspace 'ML-w-Azure-ws'.

The screenshot shows the 'Artifacts' tab selected in the 'diabetes_prediction_model:1' details page. The main content area lists the artifacts:

- diabetes_model.pickle

Below the artifact list are 'Refresh' and 'Download all' buttons. The top right corner shows the user profile 'JAYABHARATHI H...' and the workspace 'ML-w-Azure-ws'.

3. Deploy the Registered Model.

1. To deploy the Model , prerequisite files:

- Environment (.yaml) file
- Entry Script (score.py) file.

Environment (.yaml) file:

```

name: model-env
channels:
  - conda-forge
dependencies:
  - python=3.9
  - numpy=1.23.5
  - pip=23.0.1
  - scikit-learn=1.2.2
  - scipy=1.10.1
  - pip:
      -azureml-defaults==1.53.0
      -inference-schema[numpy-support]==1.5.1
      -joblib==1.2.0

```

Entry Script (score.py) file.

```

import json
import joblib
import numpy as np
from azureml.core.model import Model

def init():
    global model
    model_path = Model.get_model_path('diabetes_prediction_model')
    model = joblib.load(model_path)

def run(raw_data):
    try:
        data = json.loads(raw_data)['data']
        data = np.array(data).reshape(1, -1)

        # Perform prediction using the loaded scikit-learn model
        result = model.predict(data)

        # You can return the result as a dictionary or in any desired format
        return json.dumps({"result": result.tolist()})
    except Exception as e:
        return json.dumps({"error": str(e)})

```

2. Deploying Model in Azure

We can deploy using python and via Azure Portal - Workspace. I have given below two ways.

- Deployment using python code

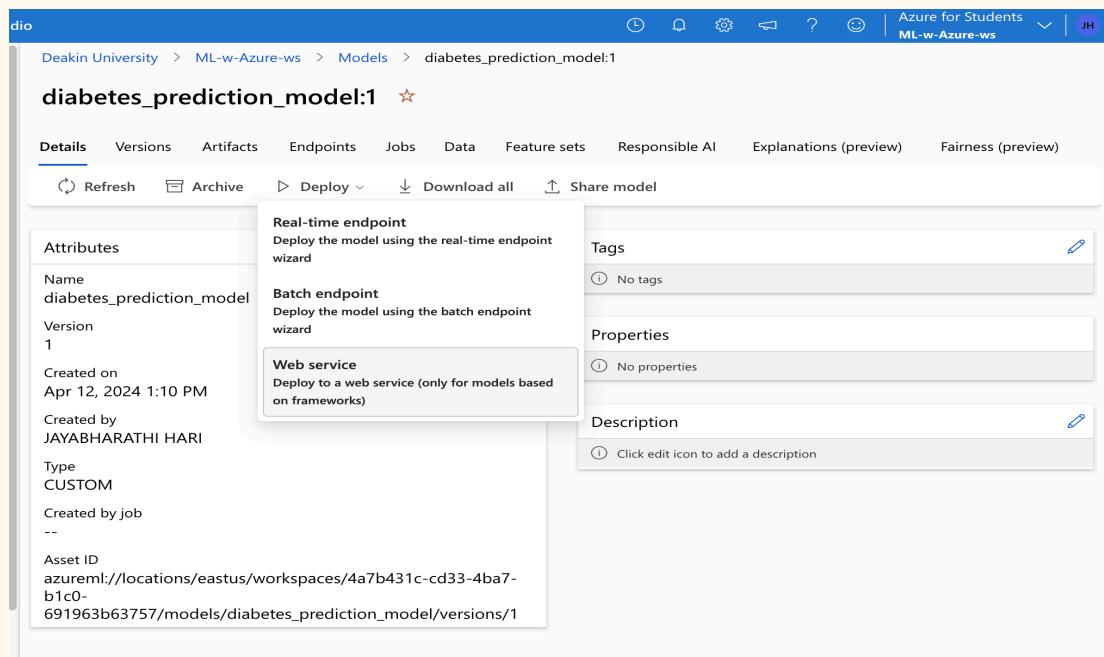
```
[ ] 1 cd /content/gdrive/MyDrive/Education/Masters_Data_Science/Azure_ML_SDK/
[ ] 1 import os
2
3 entry_script_path = os.path.join(os.getcwd(), 'score.py')
4
5 if not os.path.exists(entry_script_path):
6     print('Entry script not found at:', entry_script_path)

[ ] 1
2 # Load environment from Conda specification file
3 env = Environment.from_conda_specification(name="azure_ml", file_path="/content/gdrive/MyDrive/MS_DataScience/ML-w-Azure/conda.yaml")
4
5 # Define inference configuration
6 inference_config = InferenceConfig(entry_script="score.py", environment=env)
7
8 # Deploy the model as a web service
9 deployment_config = AciWebservice.deploy_configuration(cpu_cores=1, memory_gb=1)
10 service = Model.deploy(workspace=ws,
11                         name="diabetes-prediction-service", ##Endpoint Name
12                         models=[model], ##model is variable of registered model
13                         inference_config=inference_config,
14                         deployment_config=deployment_config)
15 service.wait_for_deployment(show_output=True)
```

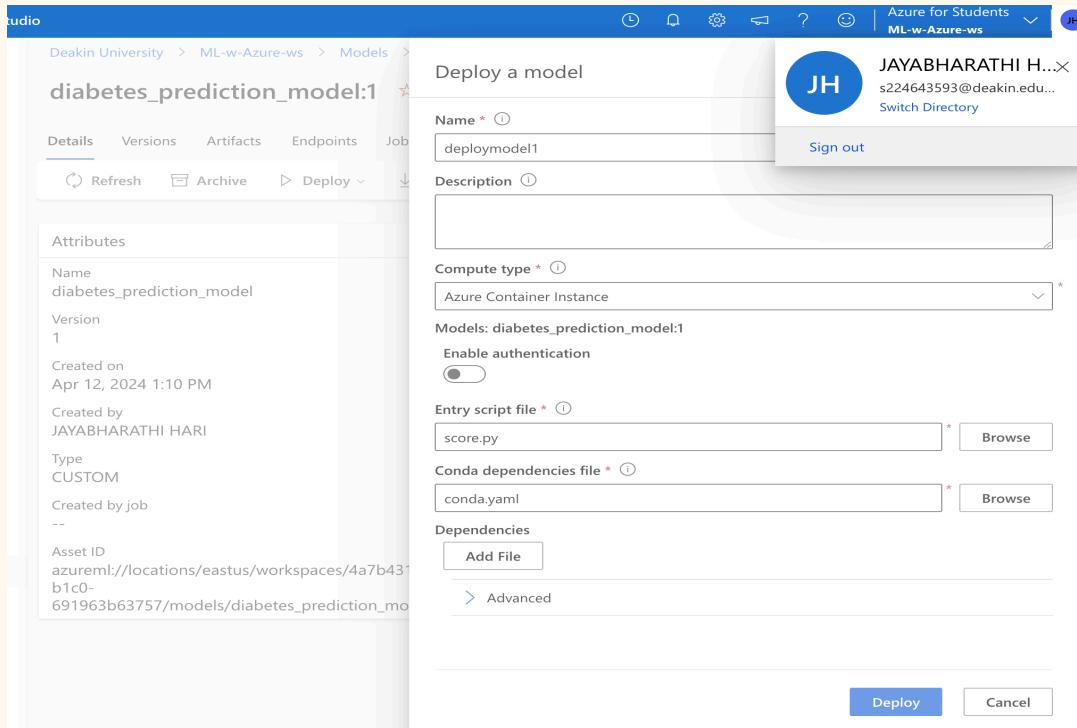
- Deployment via Azure Workspace

We are Deploying model as “Web Service” –

Step1: Model → Select Registered Model which wants to Deploy → Then Select “Deploy”
 → Select “Web Service” from the dropdown as given below.



Step2: We need to provide “endpoint name”, select compute instance, then upload an entry script and Environment file then deploy the model.



Step3: We can check Jobs as shown below.

The screenshot shows the Azure AI | Machine Learning Studio interface. The left sidebar includes sections for All workspaces, Home, Model catalog, Authoring (Notebooks, Automated ML, Designer, Prompt flow), Assets (Data, Jobs, Components, Pipelines, Environments, Models, Endpoints), Manage (Compute, Monitoring, Data Labeling, Linked Services), and a preview for Data Labeling. The 'Jobs' section is currently selected. In the center, a specific job named 'eager_kumquat_45bklmb5' is displayed. The job status is 'Completed'. The 'Properties' section shows details such as Status (Completed), Job type (Command), Experiment (prepare_image), Environment (AzureML-Image-Build:32), Duration (4m 34.68s), Compute duration (4m 34.68s), Name (imgbldrurn_f0cea02), Script name (script.py), and Created by (JAYABHARATHI HARI). The 'Compute' section shows Target (Serverless), Priority (Dedicated (Standard)), Virtual Machine Size (STANDARD_E4DS_V4), and Instance count (1). The 'Metrics' and 'Description' sections are currently empty.

Step4: Once job(Preparing Image) is done, then we can see the Endpoint

The screenshot shows the 'Endpoints' section of the Azure AI | Machine Learning Studio. At the top, there are tabs for 'Real-time endpoints', 'Batch endpoints', 'Azure OpenAI', and 'Serverless endpoints' (with a 'PREVIEW' link). Below the tabs is a toolbar with buttons for '+ Create', 'Refresh', 'Delete', and 'Reset view'. A 'Filter' and 'Columns' button are also present. The main table lists one endpoint:

Name	Description	Quota type	Created on	Created by	Updated on	Compute type	Compute target
deploymodel			Apr 12, 2024 2:53 PM	JAYABHARATHI HARI	Apr 12, 2024 2:53 PM	Container instance	

Step5: we can use scoring uri for predicting.

The image contains two side-by-side screenshots of the Azure AI | Machine Learning Studio interface, both showing the details for the 'deploymodel' endpoint.

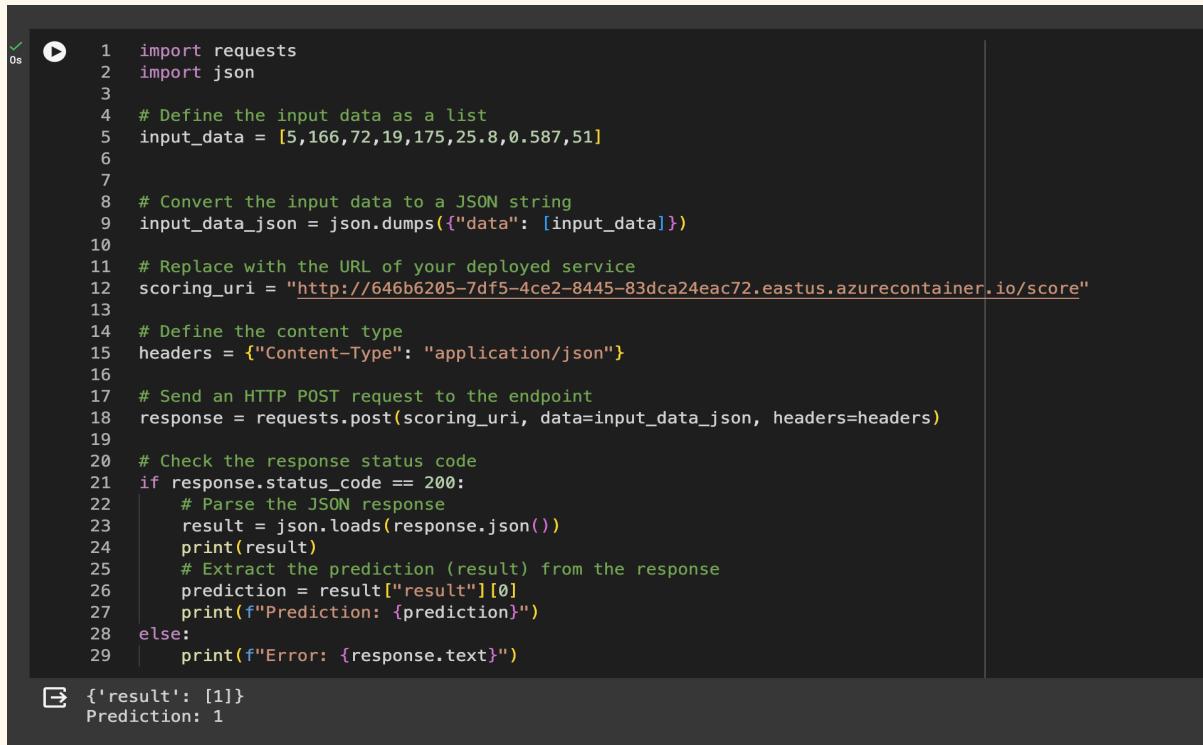
Screenshot 1 (Left): This view shows the 'Details' tab selected. It displays the REST endpoint URL: <http://646b6205-7df5-4ce2-8445-83dca24eac72.eastus.azurecontainer.io/score>. Other visible details include the Model ID 'diabetes_prediction_model:1', Created on 'Apr 12, 2024 2:53 PM', and Last updated on 'Apr 12, 2024 2:53 PM'. The Swagger URI is listed as <http://646b6205-7df5-4ce2-8445-83dca24eac72.eastus.azurecontainer.io/swagger.json>.

Screenshot 2 (Right): This view shows the 'Test' tab selected. It displays the same REST endpoint URL and other endpoint attributes. The 'Properties' section includes the following settings:

- hasInferenceSchema: True
- hasHttps: False
- authEnabled: False

3. Model - Testing/Prediction

- Testing with python code and scoring URI
 - This code sends an HTTP POST request to a deployed machine learning model hosted as a web service, passing input data in JSON format. The model processes the data and returns a prediction. If the response status is 200 (OK), it extracts and prints the prediction; otherwise, it prints the error message.



The screenshot shows a Jupyter Notebook cell with the following Python code:

```
1 import requests
2 import json
3
4 # Define the input data as a list
5 input_data = [5,166,72,19,175,25.8,0.587,51]
6
7
8 # Convert the input data to a JSON string
9 input_data_json = json.dumps({"data": [input_data]})
```

Line 10 is partially visible. Line 11 contains a URL starting with "http://". Lines 12 through 29 are part of a conditional block. Line 29 ends with a closing brace. The output pane at the bottom shows the result of the execution:

```
{'result': [1]}
Prediction: 1
```

- Testing in Workspace - Azure Portal

- We can predict/test in workspace-Azure Portal “Endpoint→Test→Past the Data”

Deakin University > ML-w-Azure-ws > Endpoints > deploymodel

deploymodel

Details Test Consume Logs

Input data to test endpoint

Test

Test result

```
{"data": [5, 166, 72, 19, 175, 25.8, 0.587, 51]}
```

```
{
  "result": [
    {
      "id": 1
    }
  ]
}
```

Azure AI | Machine Learning Studio

Deakin University > ML-w-Azure-ws > Endpoints > deploymodel

deploymodel

Details Test Consume Logs

Input data to test endpoint

Test

Test result

```
{"data": [5, 166, 72, 19, 175, 25.8, 0.587, 51]}
```

```
{
  "result": [
    {
      "id": 1
    }
  ]
}
```

Sign out