

DEPLOYING AZURE ML MODEL USING IRIS DATASET OR DIABETES

Using Iris Dataset:

1. INTRODUCTION AND OVERVIEW

Purpose: *This document aims to guide stakeholders through the process of deploying a machine learning model trained on the Iris dataset using Azure Machine Learning.*

Audience: *Data scientists, developers, and operations teams involved in model deployment and maintenance.*

I used Iris Flower Dataset from Kuggle website, which has 150 samples of iris flower which has three species; Virginica, Versicolor, and Setosa. I have to train ML Model to be able to classify whether an Iris flower is Virginica, Versicolor, or Setosa by looking at the following features: Sepal Length , Sepal Width, Petal Length and Petal Width.

2. SYSTEM ARCHITECTURE

Diagrams: *Include a flowchart or architectural diagram showing how different components like Azure ML Workspace, Compute Instance and Model Deployment interact.*

3. DEPLOYMENT ENVIRONMENT

Hardware specifications:

System Manufacturer – HP

Processor – 12th Gen Intel(R) Core(TM) i7-1255U, 1700 Mhz, 10 Core(s), 12 Logical Processor(s)

Hardware Abstraction Layer – Version= “10.0.22621.2506”

BIOS Version/Date – AMI F.19, 2023/07/03

RAM – 16.0 GB

Total Physical Memory – 15.7 GB

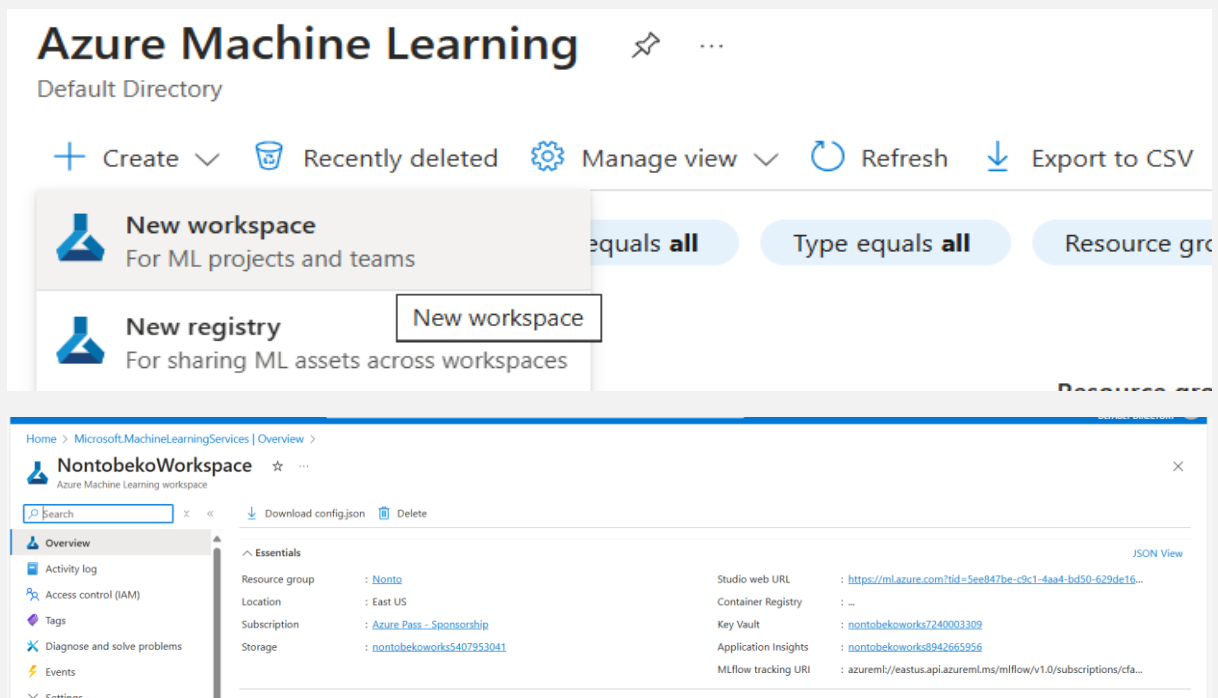
Total Virtual Memory – 32.6 GB

Software dependencies: *Azure ML SDK, Python 3.8*

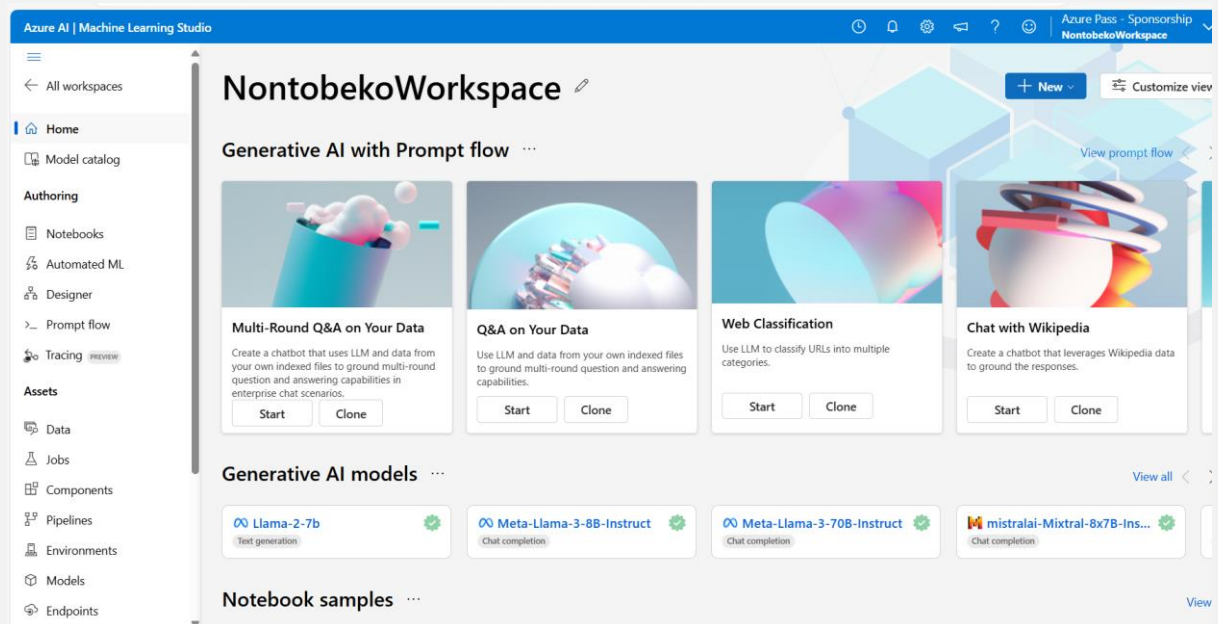
Operating System: *I am using Microsoft Windows 11 home Single Language*

4. DEPLOYMENT STEPS

Step 1 : Create AIML Workspace



Step 2 : Open the workspace studio



Step 3 : Upload the Iris dataset under Data component within the workspace.

Create data asset

Data type

Data source

Destination storage type

File selection

Review

Choose a file

Choose a file to upload from your local drive. With the uri_file type, you must upload a single file.

Upload path

azureml://subscriptions/cfa09243-bd89-47df-a88e-eee19f42d17f/resourcegroups/Nonto/works...

Upload file

☐

Overwrite if already exists

Upload list

Iris.csv

4.99 KB/4.99 KB

...

Back

Next

Cancel

Information

What file types can I use?

Supported file types include: delimited (such as csv or tsv), Parquet, JSON Lines, and plain text.

Where are files uploaded?

Files will be uploaded to the selected datastore and made available in your workspace.

Automated ML

Designer

Prompt flow

Tracing

Assets

Data

Jobs

Components

Pipelines

Environments

Models

Default Directory > NontobekoWorkspace > Data

Data

Data assets

Datastores

Dataset monitors

Data import

Data connections

Data assets are immutable references to your data that can be created from datastores, local files, public URLs, or Open Datasets. Data assets created with AzureML v2 APIs cannot be deleted, but you can up-version or archive them for easy referencing and reuse in machine learning tasks. Deleting data assets created with v1 APIs will permanently delete the data asset and all metadata.

Create

Refresh

Archive

Reset view

Show latest version only

Include archived

View my data

Search

Filter

Columns

Name	Source	Version	Created on	Modified on	Type	Properties	Created by
Irisdataset	This workspace	1	Jun 10, 2024 9:44 AM	Jun 10, 2024 9:44 AM	File		Nompumele

Step 4: Create the compute instance and open the Jupyter Lab

Compute

The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created compute targets using those types.

Compute instances

Compute clusters

Kubernetes clusters

Attached computes

Serverless instances

Choose from a selection of CPU or GPU instances preconfigured with popular tools such as VS Code, JupyterLab, Jupyter, and RStudio, ML packages, deep learning frameworks, and GPU drivers.

New

Refresh

Start

Stop

Restart

Schedule and idle shutdown

Delete

Reset view

View quota

Search

Filter

Columns

Name	State	Idle shutdown	Applications	Size
NontobekoCompute	Running	1 hour	JupyterLab Jupyter VS Code (Web)	Standard_DS11_v2

Step 5: Use the python 3.8 Azure ML

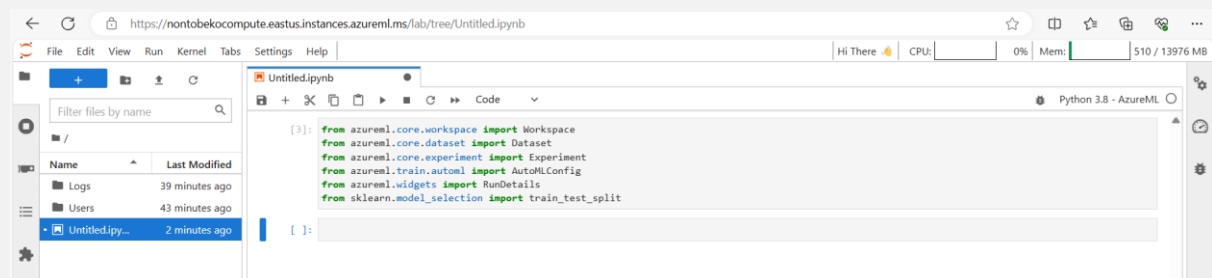
Import the necessary packages and libraries

```
[10]: pip install workspace

Requirement already satisfied: workspace in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (0.3.1)
Requirement already satisfied: sprinkles>=0.4.4 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from workspace) (0.4.6)
Note: you may need to restart the kernel to use updated packages.

[11]: pip install dataset

Requirement already satisfied: dataset in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (1.6.2)
Requirement already satisfied: banal>=1.0.1 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from dataset) (1.0.6)
Requirement already satisfied: alembic>=0.6.2 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from dataset) (1.13.1)
Requirement already satisfied: sqlalchemy<2.0.0,>=1.3.2 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from dataset) (1.4.52)
Requirement already satisfied: importlib-resources; python_version < "3.9" in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from alembic>=0.6.2->dataset) (5.12.0)
Requirement already satisfied: importlib-metadata; python_version < "3.9" in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from alembic>=0.6.2->dataset) (6.6.0)
Requirement already satisfied: typing-extensions>=4 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from alembic>=0.6.2->dataset) (4.12.2)
Requirement already satisfied: Mako in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from alembic>=0.6.2->dataset) (1.3.5)
Requirement already satisfied: greenlet<=0.4.17; python_version >= "3" and (platform_machine == "aarch64" or (platform_machine == "ppc64le" or (platform_machine == "x86_64" or (platform_machine == "amd64" or (platform_machine == "AMD64" or (platform_machine == "win32" or platform_machine == "WIN32"))))) in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from sqlalchemy<2.0.0,>=1.3.2->dataset) (2.0.2)
Requirement already satisfied: zipp>=3.1.0; python_version < "3.10" in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from importlib-resources; python_version < "3.9"->alembic>=0.6.2->dataset) (3.12.0)
Requirement already satisfied: MarkupSafe>=0.9.2 in /anaconda/envs/azureml_py38/lib/python3.8/site-packages (from Mako->alembic>=0.6.2->dataset) (2.1.5)
Note: you may need to restart the kernel to use updated packages.
```



Step 6: Connect to your workspaces

```
[7]: ws = Workspace.from_config()
```

Step 7: Work with the datasets and read your data. (I used Pandas)

```
[3]: ws.datasets

[3]: {'training_data': DatasetRegistration(id='afd30256-2f14-4615-923a-5a8d8e3b7506', name='training_data', version=14, description='', tags={}, 'IsDataset': DatasetRegistration(id='f86fc89f-117f-40cf-bded-55175b4e2f3a', name='IrisDataset', version=4, description='Iris flower dataset', tags={}, 'Iris': DatasetRegistration(id='d869bd0a-ec9a-477b-a568-32db3d5a051e', name='Iris', version=1, description='', tags={})}

[4]: iris_ds = Dataset.get_by_name(workspace=ws, name="IrisDataset")
iris_df = iris_ds.to_pandas_dataframe()
iris_df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Step 8 : Train and split your data

```
dtype='object')

[14]: x_train , x_test = train_test_split(data , test_size = 0.2)
```

Step 9: Set up your automl and your experiments settings

The screenshot shows a Jupyter Notebook interface. On the left, the file explorer displays a directory structure with files: Logs, Users, Iris.csv, and Untitled.ipynb. The main area shows the code for Step 9:

```
[14]: x_train, x_test = train_test_split(data, test_size = 0.2)

[26]: automl_settings = {
    "iteration_timeout_minutes":2,
    "experiment_timeout_minutes":15,
    "enable_early_stopping":True,
    "primary_metric": 'AUC_weighted',
    "featurization": 'auto',
    "n_cross_validations":5,
}
```

Step 10: Specify the task and algorithm to use and the specie column as your label (dependent variable)

The screenshot shows the code for Step 10 in the Jupyter Notebook:

```
[28]: automl_config = AutoMLConfig(task = 'classification', debug_log = 'automl_errors.log',
    training_data = x_train,
    label_column_name = "species",
    **automl_settings)
```

Step 11: Create your experiment to use for deployment

The screenshot shows the code and output for Step 11 in the Jupyter Notebook. The code is:

```
[*]: run = experiment.submit(automl_config, show_output=True)
```

The output shows the experiment details and status:

Note: you may need to restart the kernel to use updated packages.

No run configuration provided, running on local with default configuration.
Running in the active local environment.

Experiment	Id	Type	Status	Details Page	Docs Page
iris_experiment	AutoML_b34c0fa6-ec37-4678-81f3-db6dde8c1947	automl	Preparing	Link to Azure Machine Learning studio	Link to Documentation

Current status: DatasetEvaluation. Gathering dataset statistics.
Current status: FeaturesGeneration. Generating features for the dataset.
Current status: DatasetFeaturization. Beginning to fit featurizers and featurize the dataset.
Current status: DatasetFeaturizationCompleted. Completed fit featurizers and featurizing the dataset.
Current status: DatasetCrossValidationSplit. Generating individually featurized CV splits.

2024/06/12 08:09:33 WARNING mlflow.sklearn: Model was missing function: predict. Not logging python_function flavor!

DATA GUARDRAILS:

TYPE: Class balancing detection
STATUS: PASSED
DESCRIPTION: Your inputs were analyzed, and all classes are balanced in your training data.
Learn more about imbalanced data: <https://aka.ms/AutomatedMLImbalancedData>

TYPE: Missing feature values imputation
STATUS: PASSED
DESCRIPTION: No feature missing values were detected in the training data.
Learn more about missing value imputation: <https://aka.ms/AutomatedMLFeaturization>

TYPE: High cardinality feature detection
STATUS: PASSED

```
*****

TYPE:      Missing feature values imputation
STATUS:    PASSED
DESCRIPTION: No feature missing values were detected in the training data.
            Learn more about missing value imputation: https://aka.ms/AutomatedMLFeaturization

*****

TYPE:      High cardinality feature detection
STATUS:    PASSED
DESCRIPTION: Your inputs were analyzed, and no high cardinality features were detected.
            Learn more about high cardinality feature handling: https://aka.ms/AutomatedMLFeaturization

*****

Current status: ModelSelection. Beginning model selection.

*****

ITER: The iteration being evaluated.
PIPELINE: A summary description of the pipeline being evaluated.
DURATION: Time taken for the current iteration.
METRIC: The result of computing score on the fitted pipeline.
BEST: The best observed score thus far.
*****
```

ITER	PIPELINE	DURATION	METRIC	BEST
0	MaxAbsScaler LightGBM	0:00:47	1.0000	1.0000
1	MaxAbsScaler XGBoostClassifier	0:01:18	1.0000	1.0000
2	MaxAbsScaler ExtremeRandomTrees	0:00:48	1.0000	1.0000
3	MaxAbsScaler RandomForest	0:00:48	1.0000	1.0000
4	StandardScalerWrapper LightGBM	0:00:48	1.0000	1.0000
5	StandardScalerWrapper KNN	0:00:47	1.0000	1.0000
6	SparseNormalizer XGBoostClassifier	0:01:04	0.9941	1.0000
7	SparseNormalizer RandomForest	0:00:48	0.9860	1.0000
8	RobustScaler KNN	0:00:47	1.0000	1.0000
9	MinMaxScaler RandomForest	0:00:47	1.0000	1.0000
10	StandardScalerWrapper LogisticRegression	0:00:47	1.0000	1.0000
11	StandardScalerWrapper SVM	0:00:47	1.0000	1.0000
12	StandardScalerWrapper XGBoostClassifier	0:01:04	1.0000	1.0000
13	SparseNormalizer KNN	0:00:47	0.9937	1.0000
14	RobustScaler ExtremeRandomTrees	0:00:48	1.0000	1.0000
15	SparseNormalizer XGBoostClassifier	0:01:03	0.9863	1.0000
16	MinMaxScaler ExtremeRandomTrees	0:00:48	1.0000	1.0000
17	VotingEnsemble	0:00:49	1.0000	1.0000
18	StackEnsemble	0:00:50	1.0000	1.0000

Stopping criteria reached at iteration 19. Ending experiment.

Current status: BestRunExplainModel. Best run model explanations started

2024-06-12:08:27:37,23 INFO [explanation_client.py:334] Using default datastore for uploads

Current status: ModelExplanationDataSetSetup. Model explanations data setup completed

Current status: PickSurrogateModel. Choosing LightGBM as the surrogate model for explanations

Current status: EngineeredFeatureExplanations. Computation of engineered features started

Current status: EngineeredFeatureExplanations. Computation of engineered features completed

Current status: RawFeaturesExplanations. Computation of raw features started

Current status: RawFeaturesExplanations. Computation of raw features completed

Current status: BestRunExplainModel. Best run model explanations completed

Step 12: Get the run output

[12]: best_run , model = run.get_output()
RunDetails(run).show()

AutoML_b34c0fa6-ec37-4678-81f3-db6dde8c1947:
Status: Completed

Status-

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

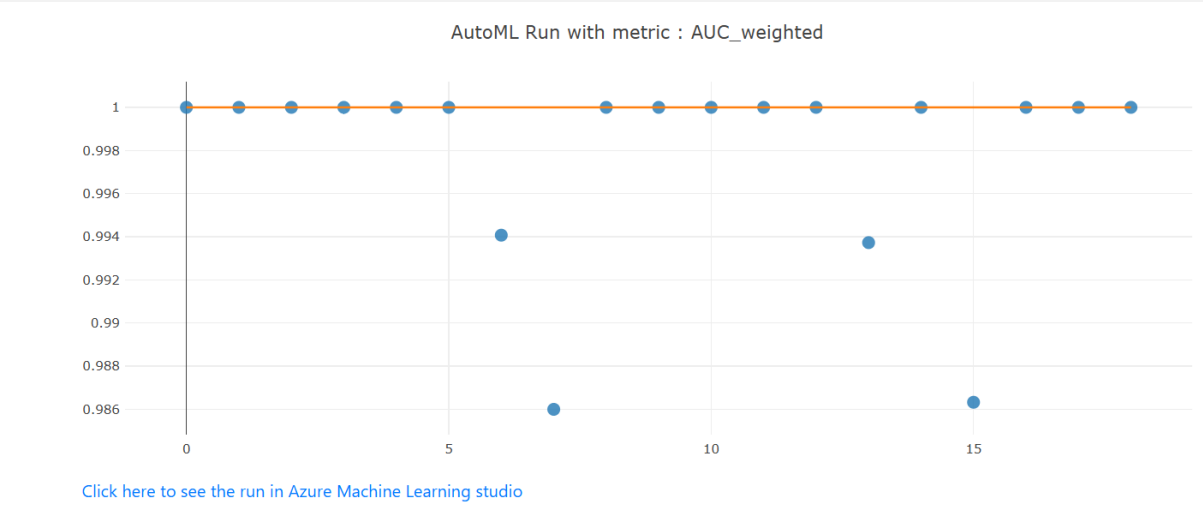
15

16

17

18

Iteration	Pipeline	Iteration metric	Best metric	Status	Duration	Started	Run Id
0	MaxAbsScaler, LightGBM	1	1	Completed	0:00:47	Jun 12, 2024 10:09 AM	
1	MaxAbsScaler, XGBoostClassifier	1	1	Completed	0:01:18	Jun 12, 2024 10:10 AM	
2	MaxAbsScaler, ExtremeRandomTrees	1	1	Completed	0:00:47	Jun 12, 2024 10:11 AM	
3	MaxAbsScaler, RandomForest	1	1	Completed	0:00:47	Jun 12, 2024 10:12 AM	
4	StandardScalerWrapper, LightGBM	1	1	Completed	0:00:48	Jun 12, 2024 10:13 AM	



Step 13: See the experiment and the scoring file created

Default Directory > NontobekoWorkspace4 > Jobs > iris_experiment > bright_wolf_8xswjfz9

bright_wolf_8xswjfz9

OverviewData guardrailsModels + child jobsOutputs + logsChild jobs

RefreshEdit and submit (preview)Register modelCancelDelete

Display name	Parent job name	Status	Created on	Durati...	Created by	Compute target
hungry_helmet_ptn9zq22	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:09 A...	47s	Nompumetelo Ng...	local
neat_bag_tz37zx9f	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:10 A...	1m 18s	Nompumetelo Ng...	local
plum_hand_tychl430	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:11 A...	47s	Nompumetelo Ng...	local
khaki_neck_8b74kyg6	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:12 A...	47s	Nompumetelo Ng...	local
icy_feast_wzcs3vsf	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:13 A...	48s	Nompumetelo Ng...	local
red_fly_94dlkw3c	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:14 A...	47s	Nompumetelo Ng...	local
heroic_pumpkin_kcdn1757	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:15 A...	1m 4s	Nompumetelo Ng...	local
sleepy_salt_vj6sbnf1	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:16 A...	48s	Nompumetelo Ng...	local
green_lobster_xgtwxgjd	AutoML_b34c0fa6-ec3...	Completed	Jun 12, 2024 10:16 A...	47s	Nompumetelo Ng...	local

hungry_helmet_ptn9zq22

OverviewModelExplanations (preview)Responsible AI (preview)MetricsData transformation (preview)Test results (preview)Outputs + logsImage

RefreshDeployDownloadExplain modelView generated codeTest model (preview)Register modelCancel

explanation

outputs

conda_env_v_1_0_0.yml

engineered_feature_names.json

env_dependencies.json

featurization_summary.json

internal_cross_validated_models.pkl

model.pkl

pipeline_graph.json

run_id.txt

scoring_file_pbl_v_1_0_0.py

scoring_file_v_1_0_0.py

scoring_file_v_2_0_0.py

File Explorer Pane

Step 14: Create and register your model

[Click here to see the run in Azure Machine Learning studio](#)

```
[13]: model_name = best_run.properties["model_name"]
      registered_name = run.register_model(model_name = model_name , description = "AutoML Iris" , tags = None)
```

Step 15: Import the packages for deployments

```
[14]: from azureml.core.model import InferenceConfig
      from azureml.core.webservice import AciWebservice , Webservice
      from azureml.core.model import Model
      from azureml.core.environment import Environment
```

Step 16: Download and bring in the scored .py file

```
from azureml.core.environment import Environment

[18]: best_run.download_file("outputs/scoring_file_v_1_0_0.py" , "inference/score.py")
```

Step 17: Wait for the deployment to complete (10 – 20mins)

The screenshot displays the Azure Machine Learning Studio interface. On the left, a file explorer shows a directory structure with files like 'AutoML_b3...', 'explanation', 'inference', 'Logs', 'Users', 'automl_err...', 'automl_err...', 'automl.log', 'azureml_au...', 'myenv.yml', and 'Untitled1.py...'. The 'Untitled1.py...' file is selected. The main area shows a Jupyter notebook with the following code:

```
[14]: from azureml.core.model import InferenceConfig
      from azureml.core.webservice import AciWebservice , Webservice
      from azureml.core.model import Model
      from azureml.core.environment import Environment

[18]: best_run.download_file("outputs/scoring_file_v_1_0_0.py" , "inference/score.py")

[19]: from azureml.automl.core.shared import constants

      best_run.download_file(constants.CONDA_ENV_FILE_PATH, "myenv.yml")
      env = Environment.from_conda_specification(name="myenv" , file_path="myenv.yml")

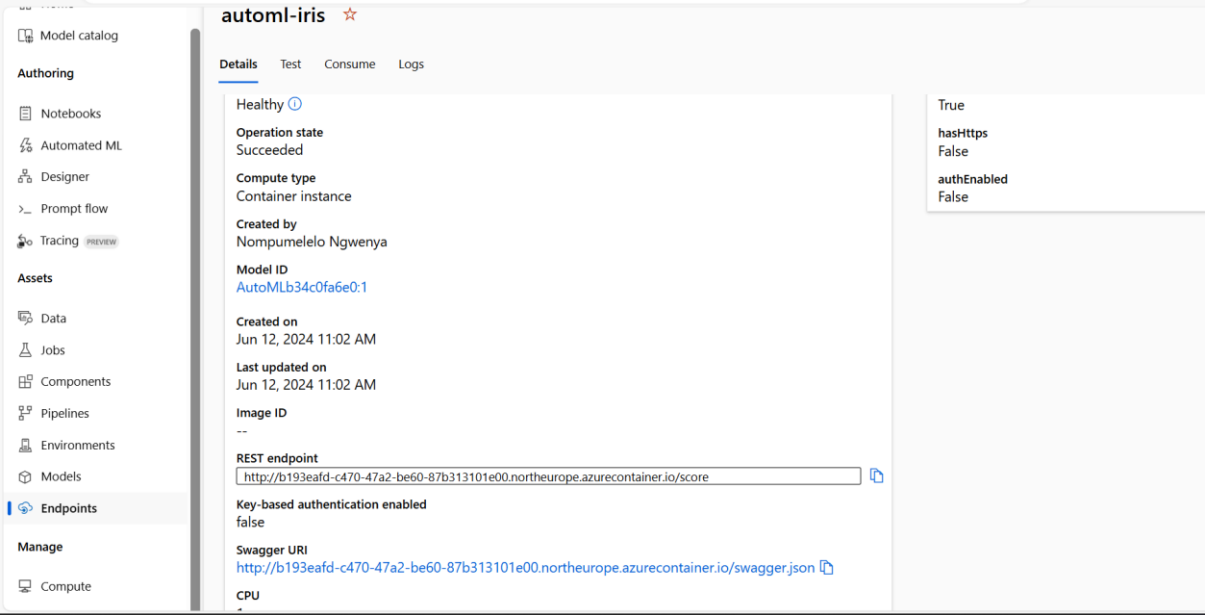
      inference_config = InferenceConfig(entry_script="inference/score.py" , environment=env)
      aciconfig = AciWebservice.deploy_configuration(cpu_cores=1 , memory_gb=1 , description="Iris classification")
      service = Model.deploy(ws , "automl-iris" , [registered_name] , inference_config , aciconfig)

      service.wait_for_deployment(True)
```

Below the code, the execution output is shown, indicating the deployment process:

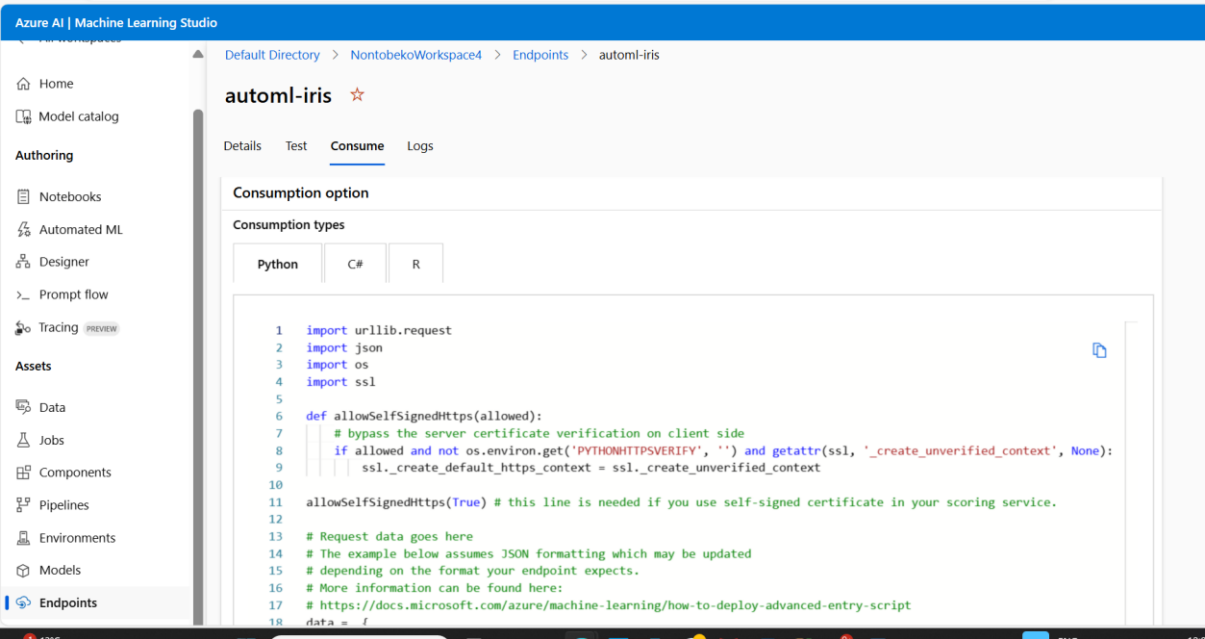
```
Tips: You can try get_logs(): https://aka.ms/debugimage#dockerlog or local deployment: https://aka.ms/debugimage#debug-locally to debug if deployment takes longer than 10 minutes.
Running
2024-06-12 09:02:23+00:00 Creating Container Registry if not exists.
2024-06-12 09:02:27+00:00 Building image..
2024-06-12 09:20:58+00:00 Generating deployment configuration.
2024-06-12 09:20:59+00:00 Submitting deployment to compute..
2024-06-12 09:21:07+00:00 Checking the status of deployment automl-iris..
2024-06-12 09:23:21+00:00 Checking the status of inference endpoint automl-iris.
Succeeded
ACI service creation operation finished, operation "Succeeded"
```

Step 18: Look at the completed deployment and copy the url link and test it



Test the predicted result: REST Endpoint link

<http://b193eafd-c470-47a2-be60-87b313101e00.northeurope.azurecontainer.io/score>



5. CONFIGURATION SETTINGS

automl_settings = {

```
    "iteration_timeout_minutes":2,  
    "experiment_timeout_minutes":15,  
    "enable_early_stopping":True,  
    "primary_metric" : 'AUC_weighted',  
    "featurization": 'auto',  
    "n_cross_validation":5,  
}  
  
automl_config = AutoMLConfig(task = 'classification', debug_log = 'automl_errors.log',  
    training_data = x_train,  
    label_column_name = 'Species',  
    **automl_settings
```

6. TESTING AND VALIDATION

Dataset : Iris Dataset (<https://www.kaggle.com/datasets/arshid/iris-flower-dataset>)

Test the predicted result: REST EndPoint

<http://b193eafd-c470-47a2-be6087b313101e00.northeurope.azurecontainer.io/score>

Procedure for testing the deployed model to ensure it performs as expected:

- (a) Environment Configuration
- (b) Data Preparation
- (c) Input Data Validation
- (d) Testing (Use typical examples of data that the model is expected to encounter in production)
- (e) Prediction Output & Accuracy Assessment
- (f) Performance Testing (Latency & Throughput
- (g) Integration Testing
- (h) Validation Against Baselines
- (i) Bias and Fairness Testing (if applicable)
- (j) Documentation of Testing Results
- (k) Based on testing results, iteratively refine the model if necessary, addressing any identified issues or performance gaps.

7. MONITORING AND LOGGING

Monitoring the performance and health of a deployed model is crucial for ensuring it continues to operate effectively and meets service level expectations. Azure provides several tools and services that can be leveraged for performance monitoring.

Azure Monitor

Metrics - Collects performance metrics such as CPU usage, memory usage, and response times of the deployed model endpoint.

Alerts - Set up alerts based on predefined thresholds for metrics, for example if response time exceeds a certain limit.

Logs: Azure Monitor can also collect logs from various Azure services, including Application Insights and Azure Machine Learning, to provide deeper insights into model performance.

Logging mechanisms are essential for troubleshooting, debugging, and auditing purposes.

Azure Monitor Logs

Querying Logs - Use Azure Monitor Logs to query and analyse logs collected from various Azure services, including Application Insights and Azure Machine Learning.

Log Analytics - Leverage Log Analytics to perform advanced queries, create dashboards, and gain insights into the operational health of the deployed model.

8. SCALABILITY AND PERFORMANCE

Scalability Considerations

When scaling a machine learning model to handle increased traffic or larger datasets on Azure, several considerations come into play to ensure optimal performance and efficiency:

- *Compute Resources, Auto-scaling, Data Storage, Load Balancing, and Caching.*

Performance Optimization

Optimizing the performance of a machine learning model on Azure involves enhancing its speed, efficiency, and resource utilization. Here are techniques and benchmarks for achieving optimal performance:

- *Model Optimization, Hardware Acceleration, Batch Processing, Model Compression, Pipeline Optimization, Benchmarking and Monitoring.*

9. SECURITY CONSIDERATIONS

Security Measures

Ensuring robust security measures are implemented during the deployment of machine learning models on Azure is crucial to protect data, maintain privacy, and comply with regulatory requirements. Security practices and compliance considerations:

- *Authentication and Authorization, Network Security, and Data Encryption*

Compliance

- *Regulatory Compliance, Data Privacy, Audit and Compliance Reporting, Legal and Ethical Considerations.*

10. MAINTENANCE AND SUPPORT

Maintenance Guidelines

Maintaining a deployed machine learning model on Azure involves regular updates, monitoring, and ensuring the model continues to perform effectively. Here are guidelines for maintaining and updating the deployed model over time:

- *Version Control, Monitoring and Performance Evaluation, Regular Updates and Retraining, Security Updates*

Troubleshooting

Documenting common issues and troubleshooting steps is essential for efficiently resolving issues that may arise during the deployment and maintenance of machine learning models on Azure. Here are common issues and corresponding troubleshooting steps:

- *Problem: Model predictions are inaccurate due to changes in input data quality or distribution.*
Troubleshooting: Implement data drift monitoring and retrain the model periodically using updated datasets.
- *Problem: Unauthorized access or data breach related to Azure resources hosting the model.*
Troubleshooting: Review Azure Security Center alerts and audit logs for suspicious activities. Implement Azure AD authentication and RBAC to restrict access.