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:

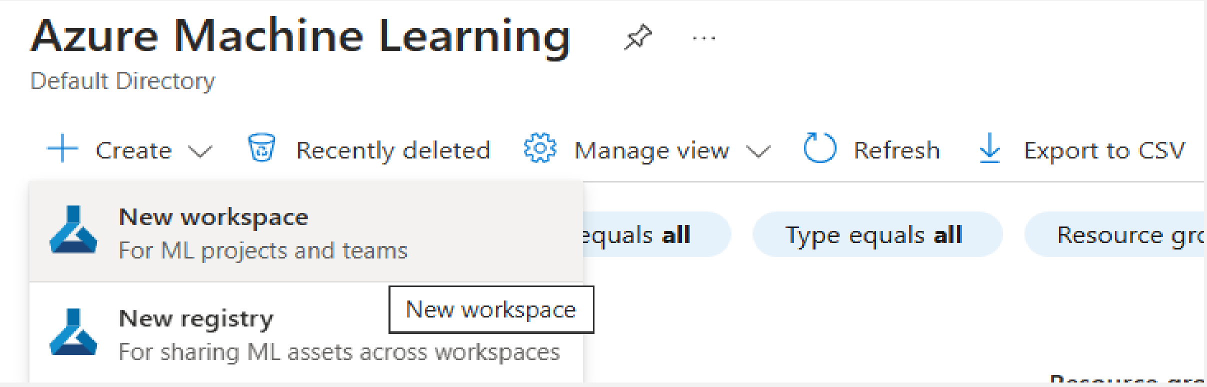
System Manufacturer – HP
Processor – 12th Gen Intel(R) Core(TM) i7-1255U, 1700 Mhz, 10 Core(s), 12 Logical Processor(s)
Hardware Abstraction Layer – Version= "10.0.22621.2506"
BIOS Version/Date – AMI F.19, 2023/07/03
RAM – 16.0 GB
Total Physical Memory – 15.7 GB
Total Virtual Memory – 32.6 GB

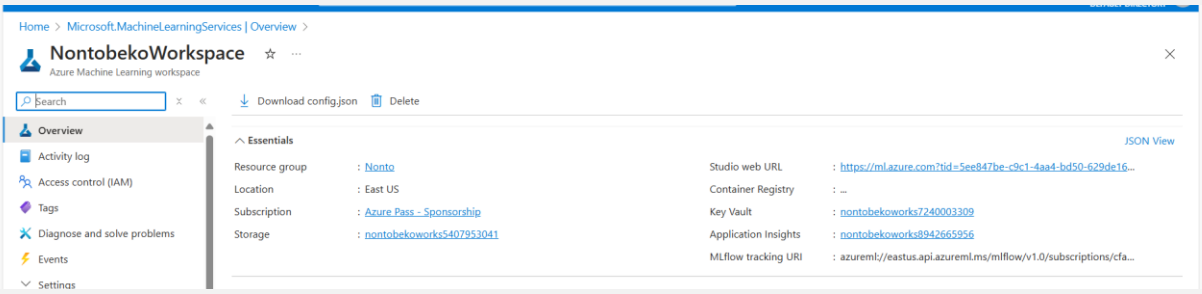
Software dependencies: *Azure ML SDK, Python 3.8*

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

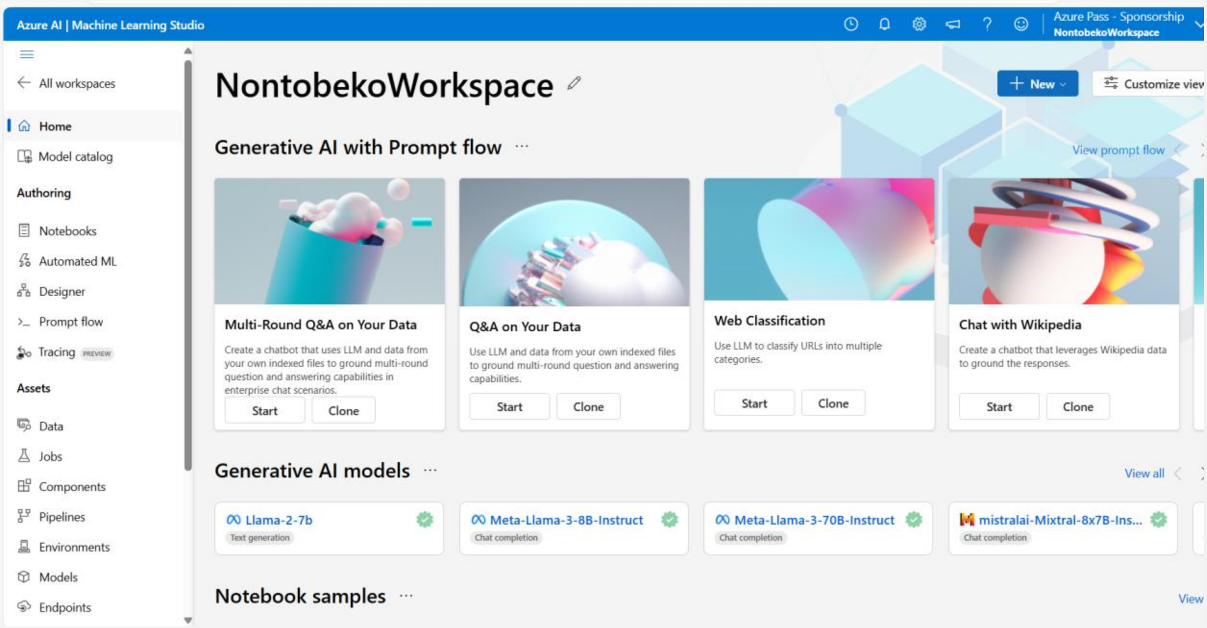
4. DEPLOYMENT STEPS

Step 1 : Create AIML Workspace

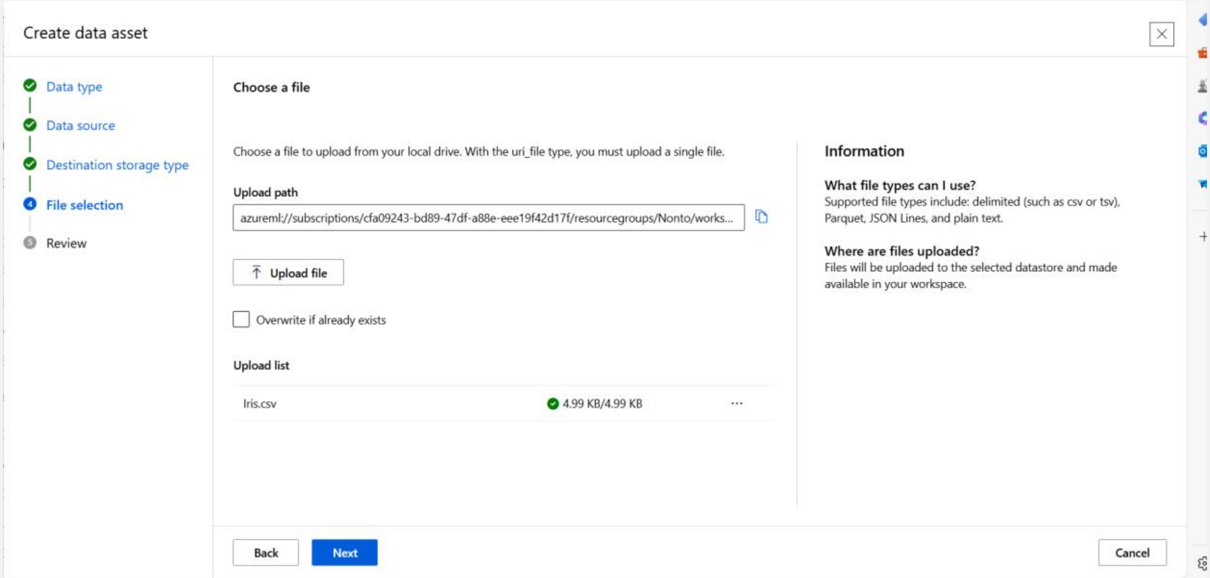


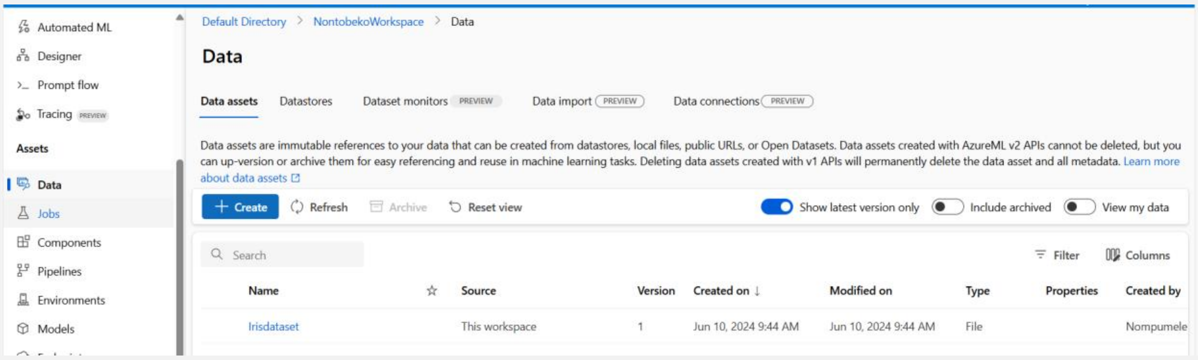


Step 2 : Open the workspace studio

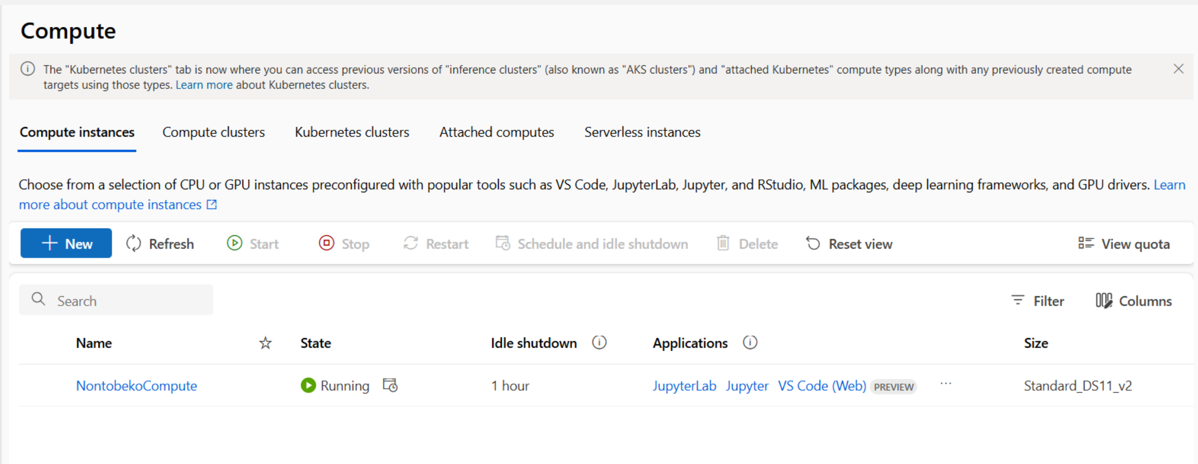


Step 3 : Upload the Iris dataset under Data component within the workspace.





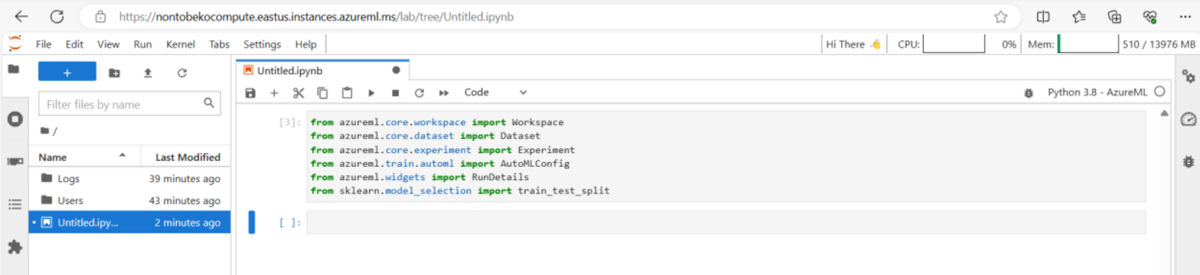
Step 4: Create the compute instance and open the Jupyter Lab



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 alemb
ic>=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 alembi
c>=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 (p
latform_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-re
sources; 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={}), 'Ir
isDataset': DatasetRegistration(id='f86fc89f-117f-40cf-bded-55175b4e2f3a', name='IrisDataset', version=4, description='Iris flower dataset', tag
s={}), '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()
```

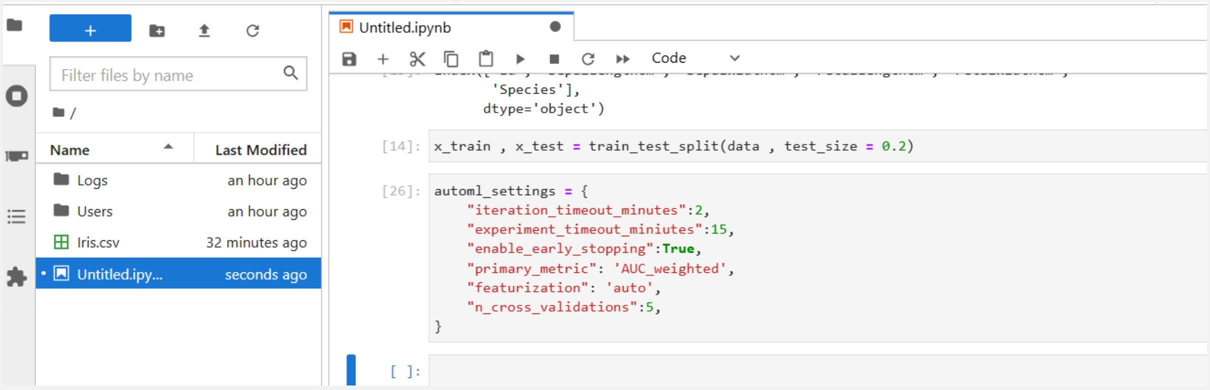
[4]:

	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

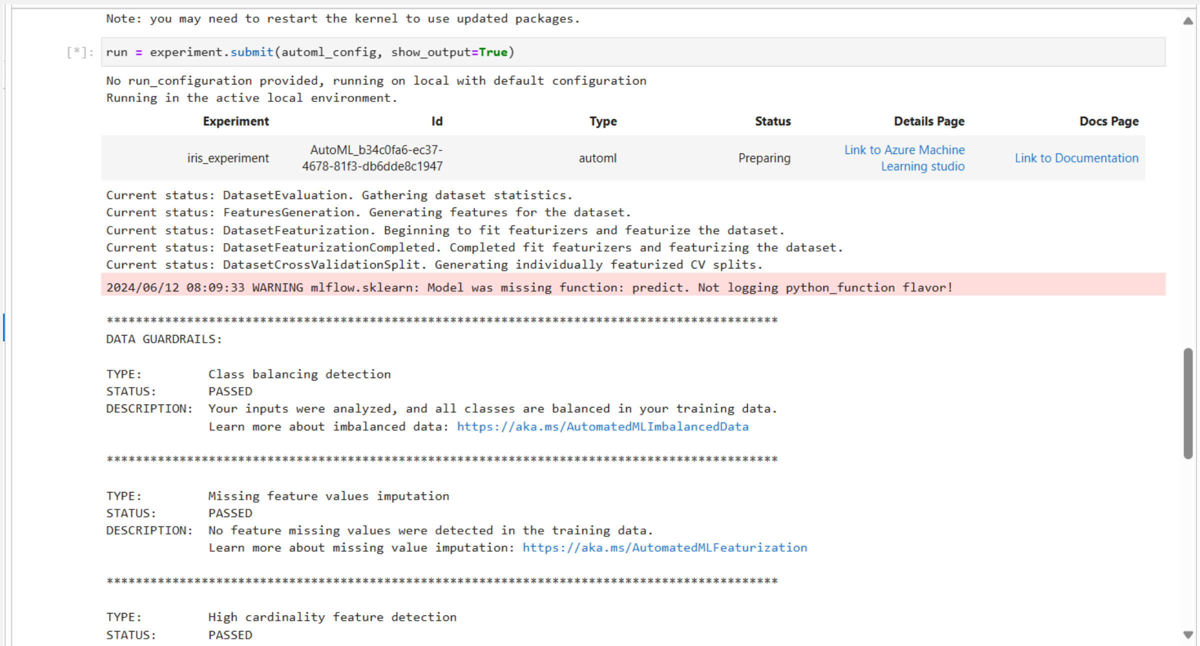
```
      dtype='object')

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

Step 9: Set up your automl and your experiments settings



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

Step 11: Create your experiment to use for deployment



```
********************************************************************************************

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
********************************************************************************************
```

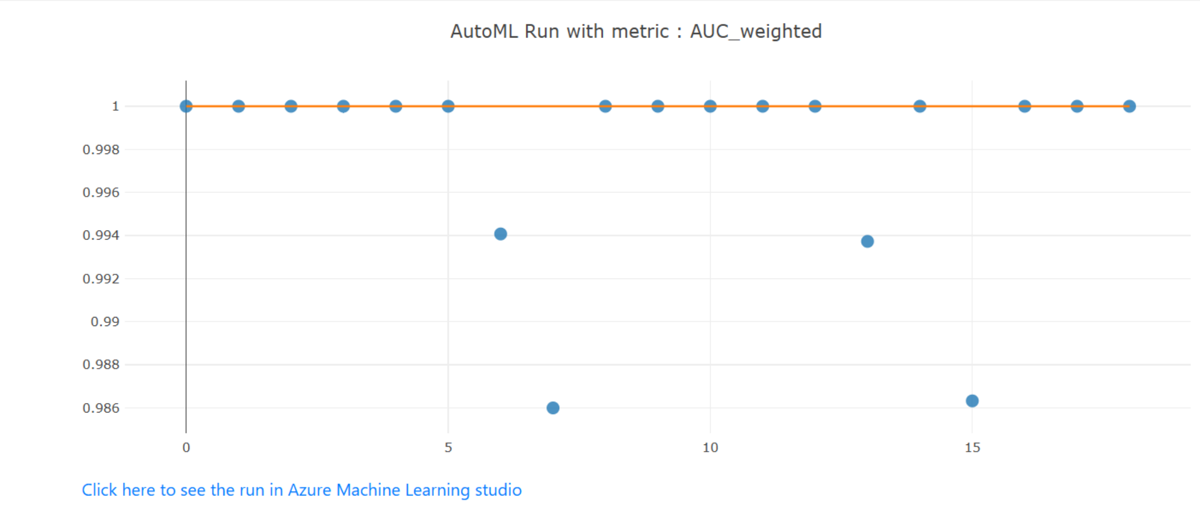
Step 12: Get the run output

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

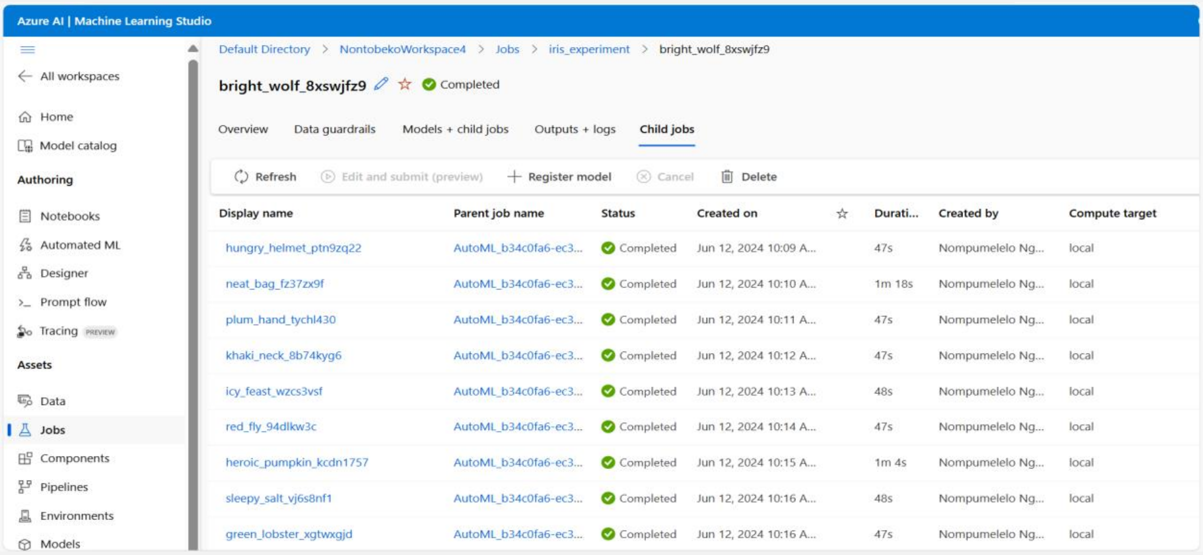
AutoML_b34c0fa6-ec37-4678-81f3-db6dde8c1947:

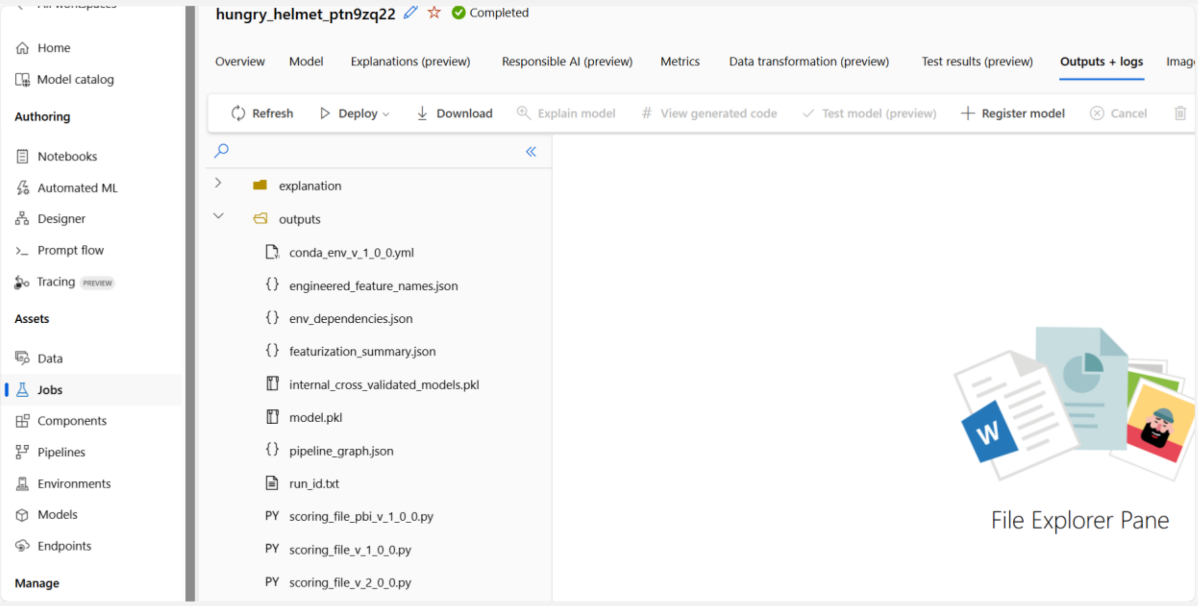
Status: Completed

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





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)
```

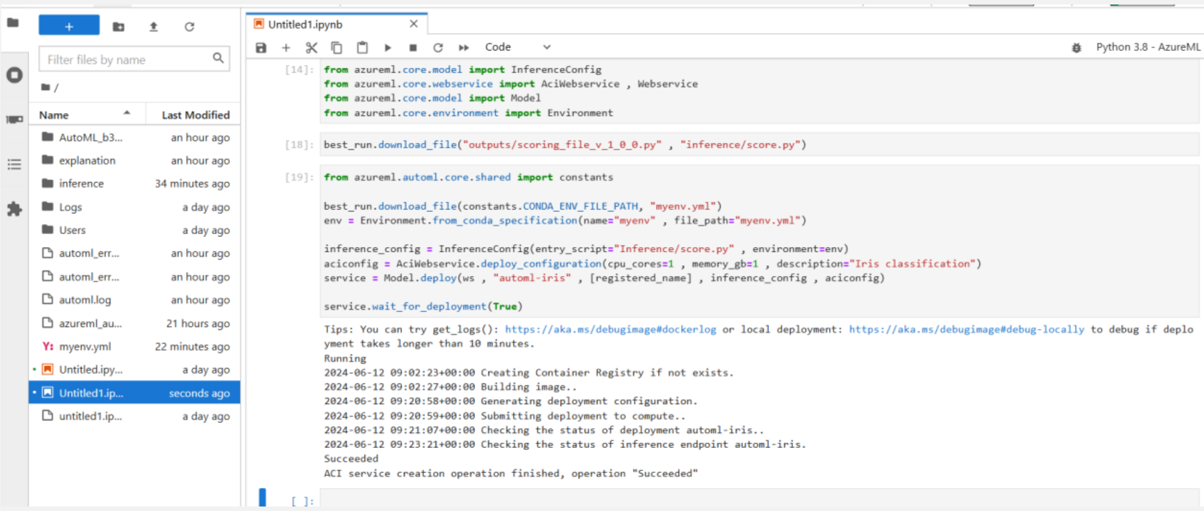
Step 15: Import the packages for deployments

```
[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
```

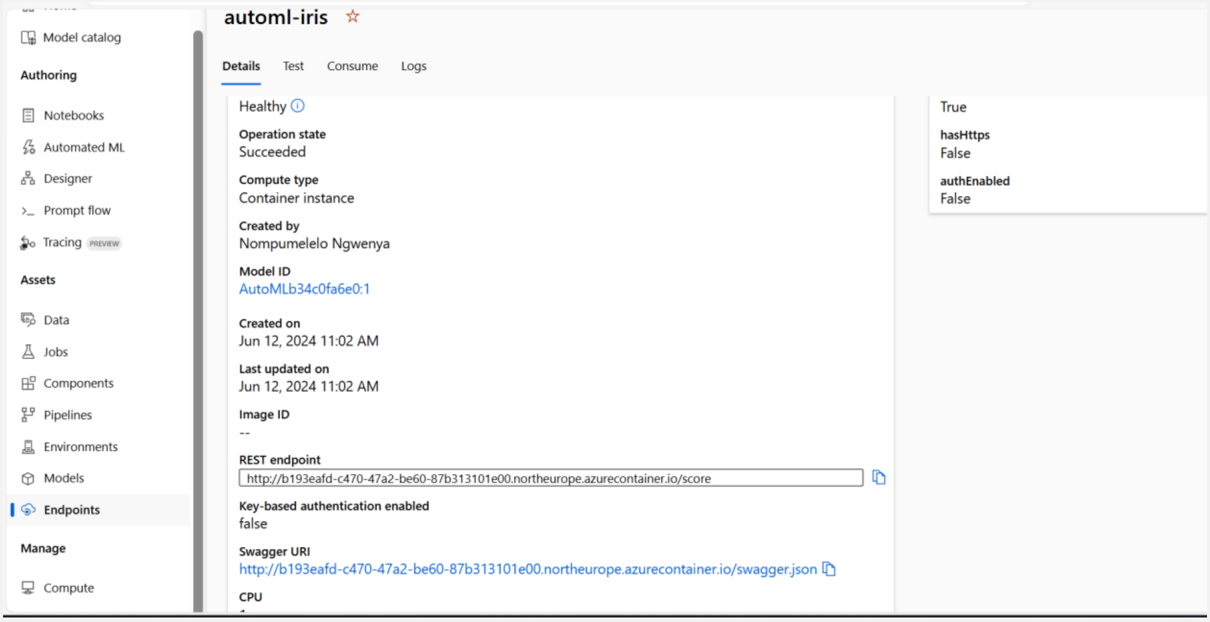
Step 16: Download and bring in the scored .py file

```
[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)

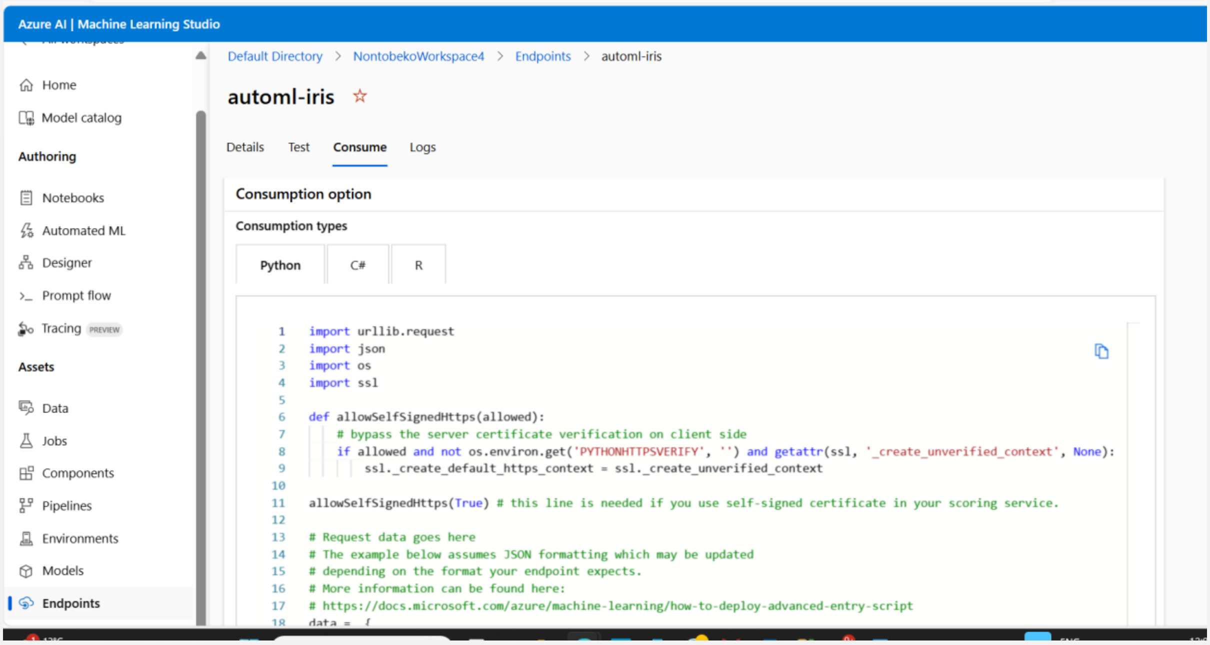


Step 18: Look at the completed deployment and copy the url link and test it



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.