DEPLOYING MACHINE LEARNING MODELS IN AZURE ML WORKPACES

AUTHOR: RAMADIMETJE LETHABO LESHILO

About

In this report, we will deploy a machine learning model for the classification of the Iris dataset. This dataset consists of 150 samples of Iris flowers belonging to 3 different types: Setosa, Versicolor, and Virginica. Each sample has 4 properties which could potentially help in identifying whether the Iris flower belongs to a certain group. These features include:

- Sepal length
- Sepal width
- Petal length
- Petal width

The goal is to classify the flower into one of these species based on the provided features.

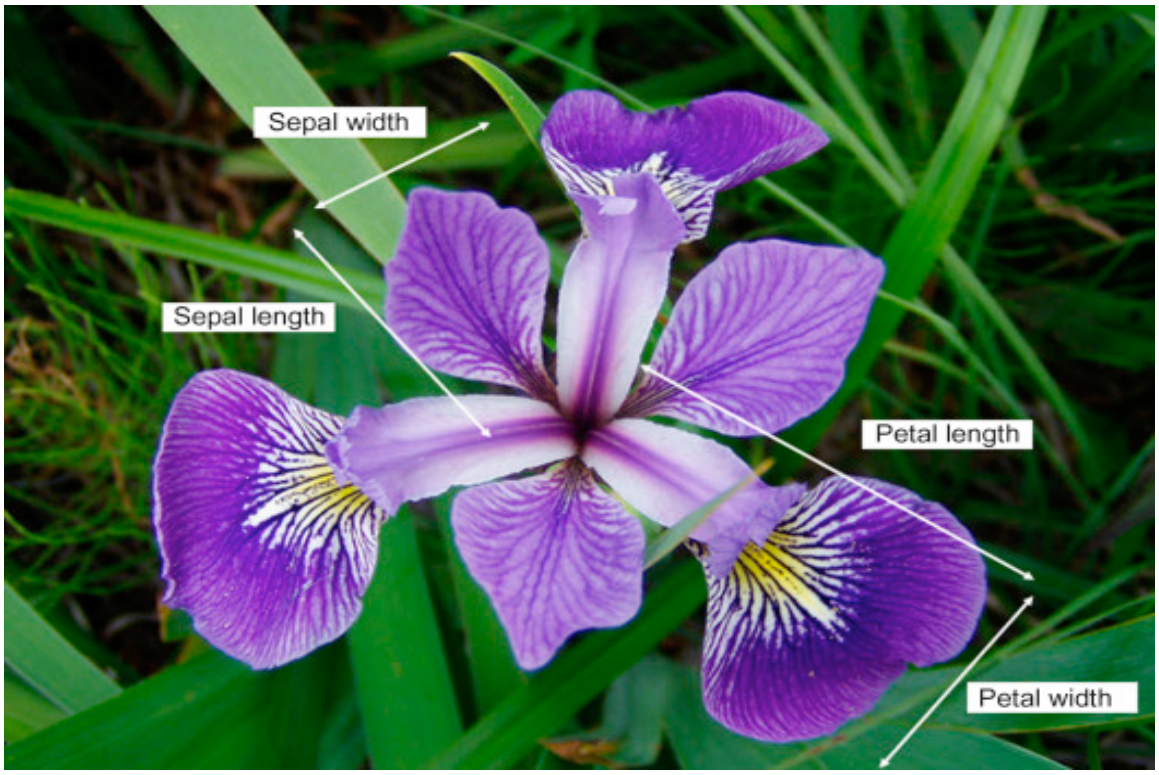


Figure 1: Visualization of the different properties of the Iris flower.

Data preparation

The dataset is loaded into a workspace, to retrieve it we have to load the workspace into our current python environment.

```
from azureml.core import Workspace, Dataset

# set the workspace to the current workspace
ws = Workspace.from_config()

print('We are currently working in:', ws)
```

```
We are currently working in: Workspace.create(name='lethaboworkspace', subscription_id='591fa0dd-8300-410d-9803-fd80e199ba02', resource_group='lethaboresource')
```

```
# Get a list of datasets in the workspace
ws.datasets
```

```
{'diabetes': DatasetRegistration(id='c04fd925-4ff5-43e1-b65f-378d9ab5d691', name='diabetes', version=1, description='', tags={}), 'iris': DatasetRegistration(id='fc04991e-b57c-4909-b950-08e3ef844ec0', name='iris', version=1, description='', tags={})}
```

Get the dataset and convert it into a pandas data frame and look at the format of the data. Drop any unnecessary features.

```
# Get the iris dataset from the workspace datasets
iris = Dataset.get_by_name(ws,name = 'iris')
```

```
df = iris.to_pandas_dataframe()
```

```
# Print the first 5 rows of the data
df.head(5)
```

	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

```
# Get the dimensions of the data
df.shape
```

```
(150, 5)
```

```
# Drop the id column since we do not need it for modelling
df = df.drop('Id', axis = 1)
```

```
df.head()
```

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

Summary statistics of the predictors.

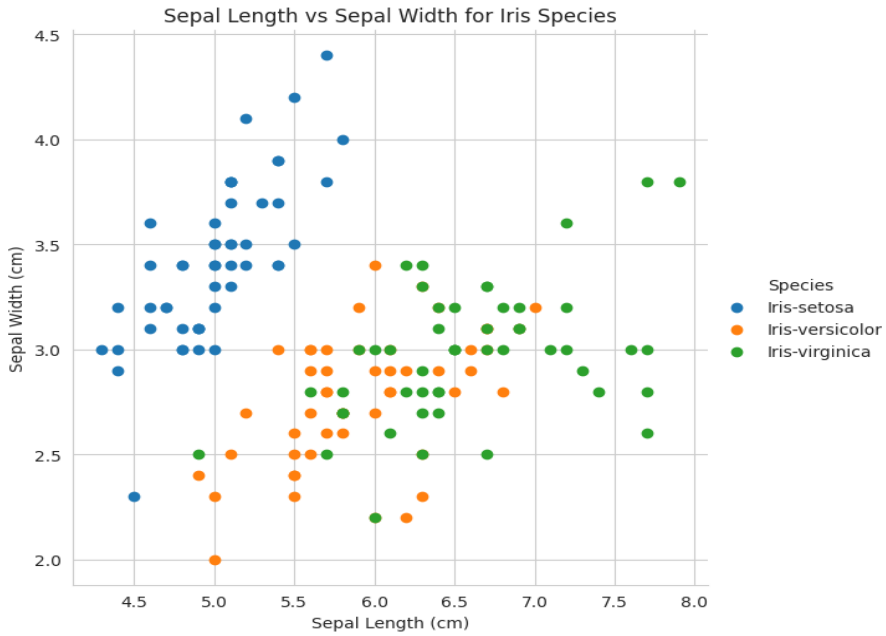
```
df.describe()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.00	150.00	150.00	150.00
mean	5.84	3.05	3.76	1.20
std	0.83	0.43	1.76	0.76
min	4.30	2.00	1.00	0.10
25%	5.10	2.80	1.60	0.30
50%	5.80	3.00	4.35	1.30
75%	6.40	3.30	5.10	1.80
max	7.90	4.40	6.90	2.50

Visualize the groups

This can be done by plotting a scatterplot of any of the two predictors that associate with one another e.g. properties related to the sepal part of the flower. From the plot we can see the different groups.

```
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(8, 10))
sns.set_style("whitegrid")

# Create the FacetGrid plot
g = sns.FacetGrid(df, hue="Species", height=6)
g.map(plt.scatter, 'SepalLengthCm', 'SepalWidthCm').add_legend()

# Add titles and axis labels
plt.title("Sepal Length vs Sepal Width for Iris Species")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Sepal Width (cm)")

# Show the plot
plt.show()
```



Model formulations and specifications

The model training will be conducted using the Azure AutoML procedure. This AutoML model will fit various classification models, and we will select the best model based on accuracy. The best model will be determined by its highest accuracy. A 5-fold cross validation resampling technique will be used to evaluate the performance the models being fitted.

```
# split the data into train and test set
from sklearn.model_selection import train_test_split
data_train, data_test = train_test_split(df, test_size = 0.3, random_state = 0)
```

```
from azureml.train.automl import AutoMLConfig

# Now let us configure our automl model
automl_config = AutoMLConfig(
    task = 'classification',
    debug_log = 'automl_errors.log',
    primary_metric = 'accuracy',
    label_column_name = 'Species',
    training_data = data_train,
    n_cross_validations = 5,
    featurization = 'auto',
    iteration_timeout_minutes = 2,
    experiment_timeout_minutes = 15,
    verbosity = logging.INFO
)
```

Create and submit a machine learning experiment. This will create a machine learning workspace job with the name “automl_experiment”.

```
from azureml.core.experiment import Experiment
# Create an experiment
automl_experiment = Experiment(ws,'automl_experiment')
# Submit the experiment
automl_run = automl_experiment.submit(automl_config)
automl_run.wait_for_completion(show_output=True)
```

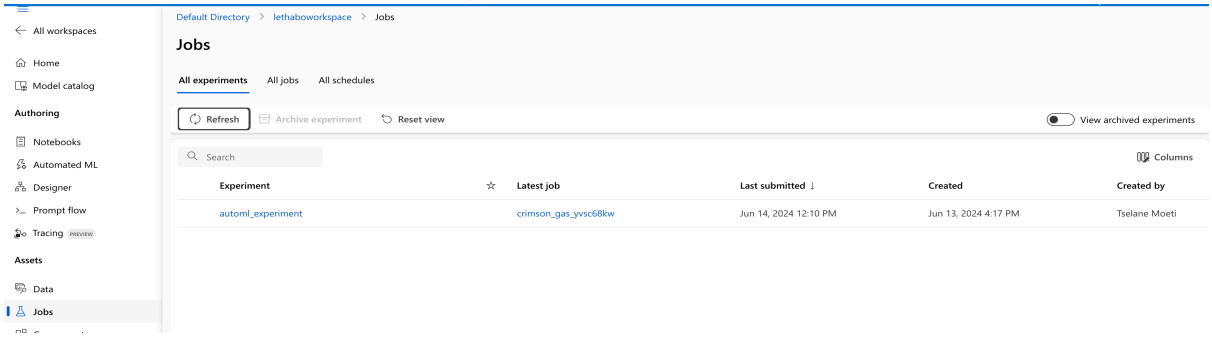
```
2024-06-14:10:10:16,168 WARNING  [local_experiment_driver.py:142] Running on local machine. Note that local runs al
ways run synchronously even if you use the parameter 'show_output=False'
```

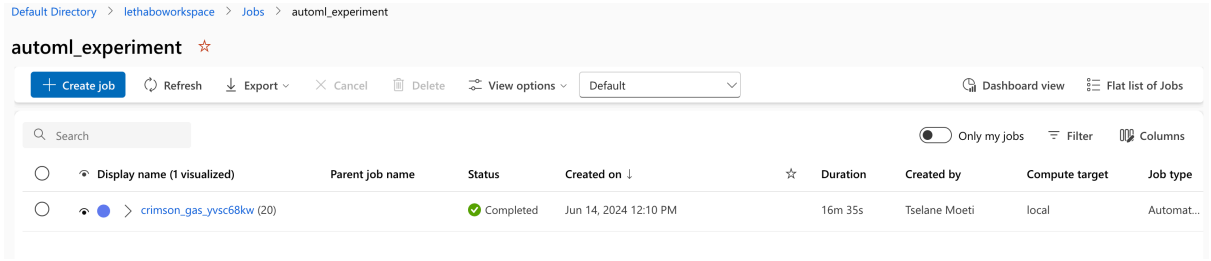
Experiment	Id	Type	Status	Details Page	Docs Page
automl_experiment	AutoML_e7f5166d-e40c-43ef-8458-cc5bab5eb72d	automl	Preparing	Link to Azure Machine Learning studio	Link to Documentation

```
2024/06/14 10:10:37 WARNING mlflow.sklearn: Model was missing function: predict. Not logging python_function flavo
r!
2024-06-14:10:27:53,527 INFO     [explanation_client.py:334] Using default datastore for uploads
```

Experiment	Id	Type	Status	Details Page	Docs Page
automl_experiment	AutoML_e7f5166d-e40c-43ef-8458-cc5bab5eb72d	automl	Completed	Link to Azure Machine Learning studio	Link to Documentation

We can check that the machine learning job has been created successfully.





Retrieve the information of the best model based on the accuracy metric.

```
# Get the information of the best run and the model
best_run, fitted_model = automl_run.get_output()
print(best_run)
```

```
Run(Experiment: automl_experiment,
Id: AutoML_e7f5166d-e40c-43ef-8458-cc5bab5eb72d_16,
Type: None,
Status: Completed)
```

```
print(fitted_model)
```

```
Pipeline(memory=None,
         steps=[('datatransformer',
                 DataTransformer(enable_dnn=False, enable_feature_sweeping=True, feature_sweeping_config={}, featur
e_sweeping_timeout=86400, featurization_config=None, force_text_dnn=False, is_cross_validation=True, is_onnx_compat
ible=False, observer=None, task='classification', working_dir='/mnt/batch/tasks/shared/LS_root/mounts/clusters/leth
abocomp/code/Users/tsemoeti24')),
                ('SparseNormalizer', Normalizer(copy=True, norm='max')),
                ('KNeighborsClassifier',
                 KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                      metric='manhattan', metric_params=None,
                                      n_jobs=1, n_neighbors=10, p=2,
                                      weights='distance'))],
         verbose=False)
Y_transformer(['LabelEncoder', LabelEncoder()])
```

One can also check the detailed run details throughout the AutoML experiment. This will display all the iterations and their status as well as a graphical representation of the metric of choice from the list of all metrics.

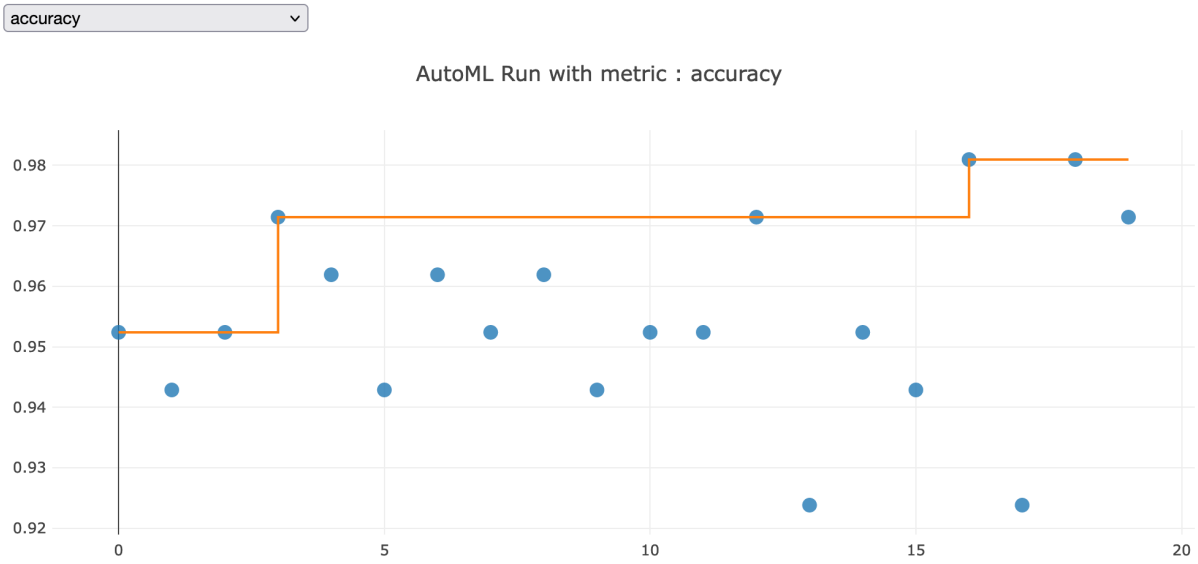
```
from azureml.widgets import RunDetails
RunDetails(automl_run).show()
```

AutoML_e7f5166d-e40c-43ef-8458-cc5bab5eb72d:
Status: Completed

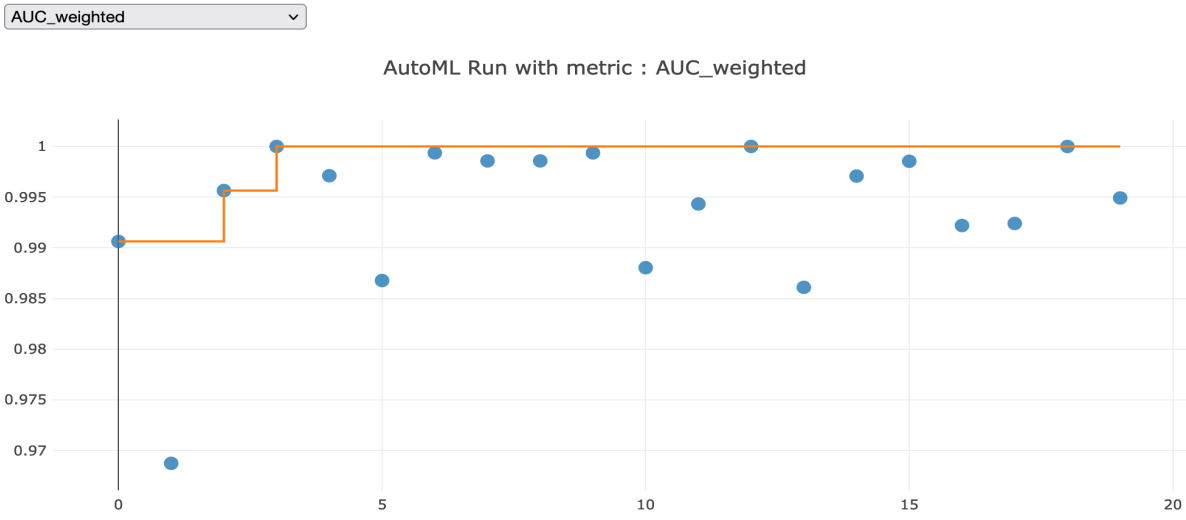
Iteration	Pipeline	Iteration metric	Best metric	Status	Duration	Started	R
16	SparseNormalizer, KNN	0.98095238	0.98095238	Completed	0:00:40	Jun 14, 2024 12:23 PM	
18	VotingEnsemble	0.98095238	0.98095238	Completed	0:00:47	Jun 14, 2024 12:25 PM	
3	SparseNormalizer, XGBoostClassifier	0.97142857	0.97142857	Completed	0:00:58	Jun 14, 2024 12:13 PM	
12	StandardScalerWrapper, LogisticRegression	0.97142857	0.97142857	Completed	0:00:43	Jun 14, 2024 12:20 PM	
19	StackEnsemble	0.97142857	0.98095238	Completed	0:00:42	Jun 14, 2024 12:26 PM	

Pages: 1 2 3 4 Next Last 5 per page

accuracy



Click here to see the run in Azure Machine Learning studio



Click here to see the run in Azure Machine Learning studio

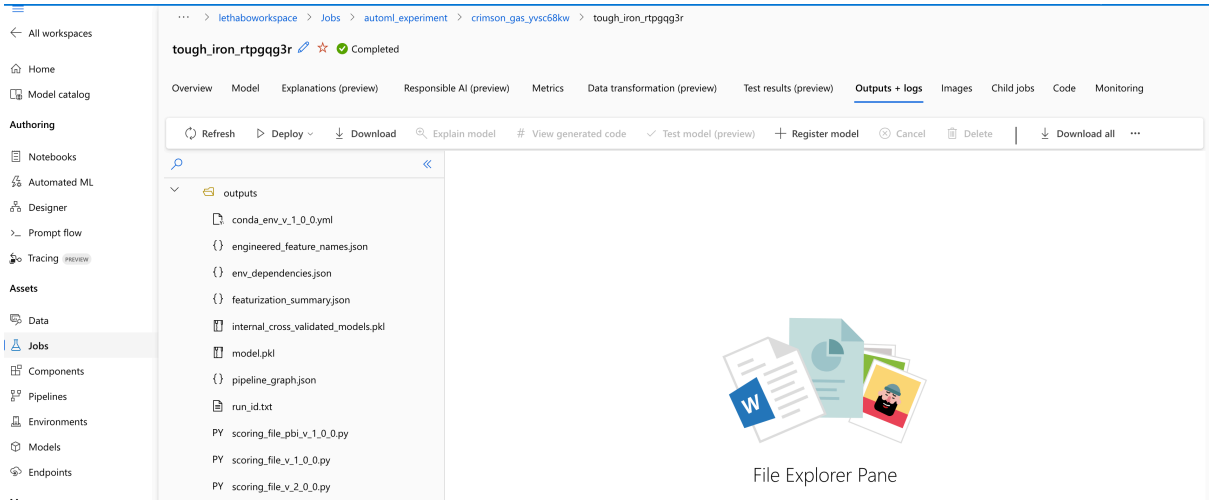
Model registration

Register the best model from the AutoML run experiment.

```
: # Get the properties of the best model such as as model name
  model_name = best_run.properties['model_name']
  registered_name = automl_run.register_model(model_name = model_name, description = "Automl Iris dataset", tags = Non
```

Deployment

Get all the information needed to deploy the model, specifically the scoring file which contains the model pickle file and inferences for prediction and the conda environment file generated by AutoML. Here is where you can get the specific location of the files found in the most recent child completed



File Explorer Pane

Store the files into the current user folder.

```
# Import the packages required to deploy the model
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
```

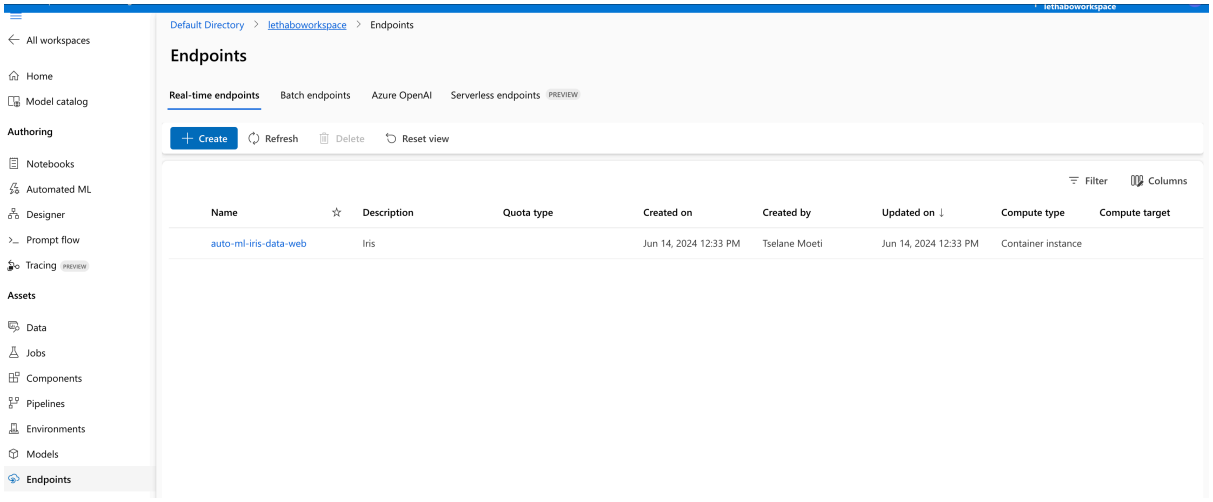
```
# Download the scoring file for the best model from the Azure ML run's output and save it locally in the 'inference'
best_run.download_file('outputs/scoring_file_v_1_0_0.py', 'inference/score.py')


from azureml.automl.core.shared import constants

# Download the Conda environment file used by the best model and save it locally
best_run.download_file(constants.CONDA_ENV_FILE_PATH, 'myenv.yml')

# Create an Azure ML environment using the downloaded Conda specification
env = Environment.from_conda_specification(name='myenv', file_path='myenv.yml')

# Define the inference configuration using the scoring script and the environment
inference_config = InferenceConfig(entry_script='inference/score.py', environment=env)

# Set up the deployment configuration for an ACI web service with 1.5 CPU cores, 1 GB of memory, and a description
aciconfig = AciWebservice.deploy_configuration(cpu_cores=1.5, memory_gb=1, description='Iris')
```

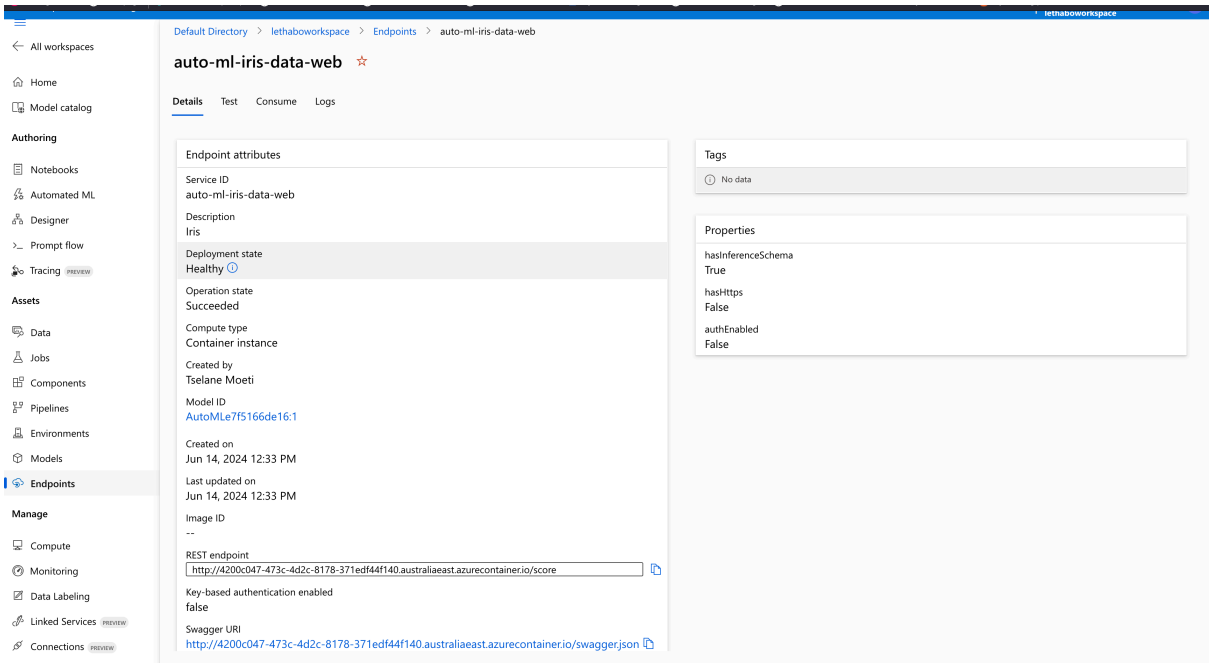
Deploy the model.

```
# Deploy the model
# Note that the name of the service cannot have spaces
service = Model.deploy(ws, "auto-ml-iris-data-web", [registered_name], inference_config, aciconfig)
service.wait_for_deployment(True)
```

```
Tips: You can try get_logs(): https://aka.ms/debugimage#dockerlog or local deployment: https://aka.ms/debugimage#de
bug-locally to debug if deployment takes longer than 10 minutes.
Running
2024-06-14 10:33:28+00:00 Creating Container Registry if not exists.
2024-06-14 10:33:28+00:00 Registering the environment.
2024-06-14 10:33:34+00:00 Use the existing image.
2024-06-14 10:33:35+00:00 Generating deployment configuration.
2024-06-14 10:33:36+00:00 Submitting deployment to compute..
2024-06-14 10:33:46+00:00 Checking the status of deployment auto-ml-iris-data-web..
2024-06-14 10:35:44+00:00 Checking the status of inference endpoint auto-ml-iris-data-web.
Succeeded
ACI service creation operation finished, operation "Succeeded"
```

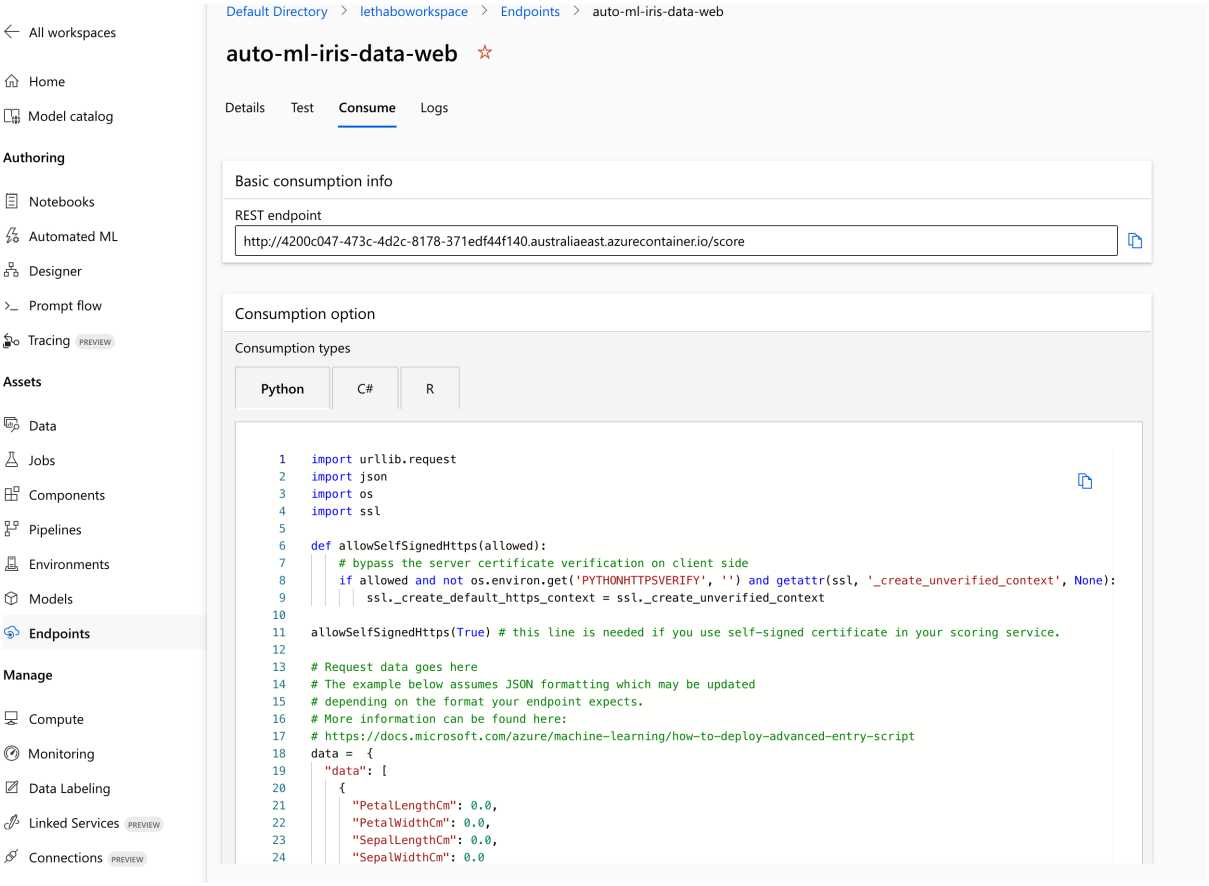
Since the model has been successfully deployed we can check our endpoint for the details, the link and information on consumption of the model.





Test the model/ Consumption

The model is now ready for consumption. We will test the model using R code which can be found in the end point “Consume” tab.



Testing the model using R code. To do so suppose that we have the following information about a particular Iris flower:

Sepal length: 3.7

Sepal width: 1.2

Petal length: 6.2

Petal width: 4.2

Here is the R code for the test prediction.

```r
library("RCurl")
library("rjson")

# Accept SSL certificates issued by public Certificate Authorities
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"), ssl.verifypeer = FALSE))

h = basicTextGatherer()
hdr = basicHeaderGatherer()

# Request data goes here
# The example below assumes JSON formatting which may be updated
# depending on the format your endpoint expects.
# More information can be found here:
# https://docs.microsoft.com/azure/machine-learning/how-to-deploy-advanced-entry-script
req = fromJSON('{
  "data": [
    {
      "PetalLengthCm": 3.7,
      "PetalWidthCm": 1.2,
      "SepalLengthCm": 6.2,
      "SepalWidthCm": 4.2
    }
  ],
  "method": "predict"
}')

requestBody = enc2utf8(toJSON(req))

h$reset()

# The azureml-model-deployment header will force the request to go to a specific deployment.
# Remove this header to have the request observe the endpoint traffic rules
curlPerform(
  url = "http://4200c047-473c-4d2c-8178-371edf44f140.australiaeast.azurecontainer.io/score",
  httpheader=c('Content-Type' = "application/json"),
  postfields=requestBody,
  writefunction = h$update,
  headerfunction = hdr$update,
  verbose = TRUE
)

headers = hdr$value()
httpStatus = headers["status"]
if (httpStatus >= 400)
{
  print(paste("The request failed with status code:", httpStatus, sep=" "))

  # Print the headers - they include the request ID and the timestamp, which are useful for debugging the failure
  print(headers)
}

print("Result:")
result = h$value()
print(result)

```

The predicted result for the given input is "Iris-versicolor".

```
> print(result)
[1] "\"{\\\"result\\\": [\\\"Iris-versicolor\\\"]}\""
```

This concludes the report on deploying a machine learning model for Iris classification using Azure ML workspaces. The process covered data preparation, model training using AutoML, model registration, deployment, and testing the deployed model.