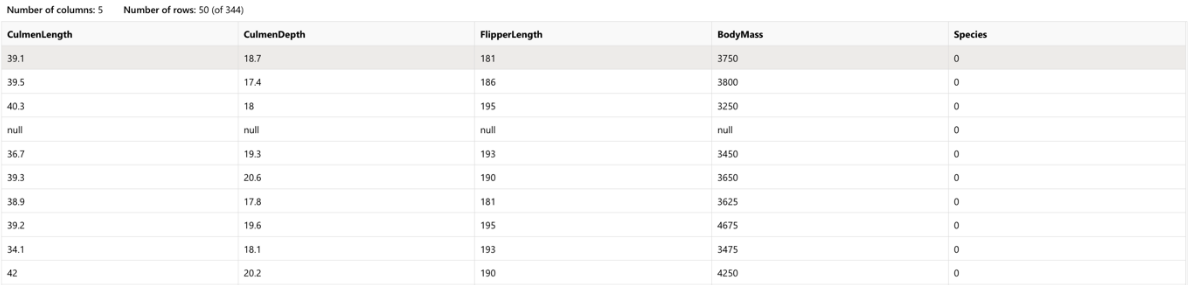
Penguin Clustering with Azure Machine Learning Studio

Overview

In this project, we utilized Microsoft Azure Machine Learning Designer to develop and deploy a clustering model for grouping penguins based on their characteristics. The workflow involved setting up the Azure workspace and compute resources, constructing and evaluating a clustering model pipeline, and creating an inference pipeline for real-time clustering. The clustering service was subsequently deployed to an Azure Container Instance and tested to ensure accurate clustering of penguins based on their attributes.

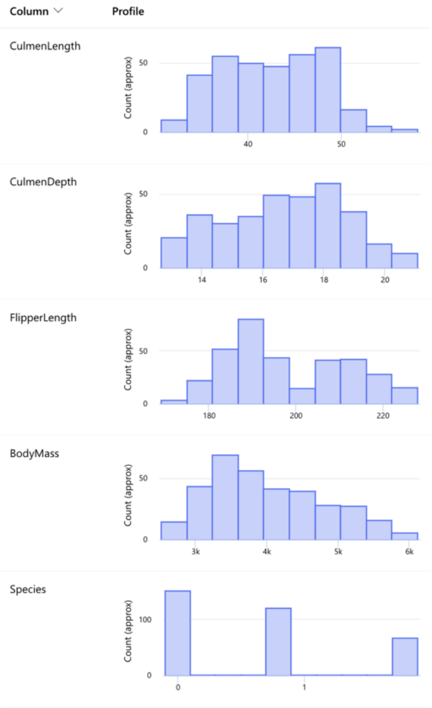
Dataset

The dataset used in this project is the "penguin-data" dataset downloaded from https://aka.ms/penguin-data. This dataset includes various features related to penguin characteristics, which are crucial for training the clustering model to group penguins based on these attributes.



Number of columns: 5 Number of rows: 50 (of 344)

CulmenLength	CulmenDepth	FlipperLength	BodyMass	Species
39.1	18.7	181	3750	0
39.5	17.4	186	3800	0
40.3	18	195	3250	0
null	null	null	null	0
36.7	19.3	193	3450	0
39.3	20.6	190	3650	0
38.9	17.8	181	3625	0
39.2	19.6	195	4675	0
34.1	18.1	193	3475	0
42	20.2	190	4250	0

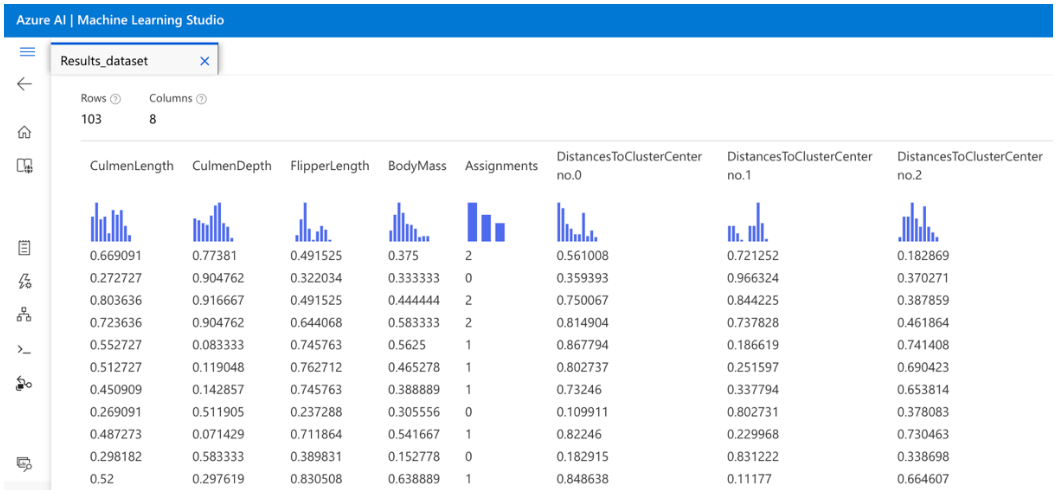


1- Setting Up Azure Machine Learning Workspace and Compute Resources

Established a Microsoft Azure Machine Learning workspace to manage data, compute resources, and models efficiently. Following this, compute targets were created to provide the necessary processing power for model training and deployment. A compute instance was set up as a development workstation, and a compute cluster was configured to handle the scalable processing of machine learning tasks, ensuring a robust environment for data science activities.

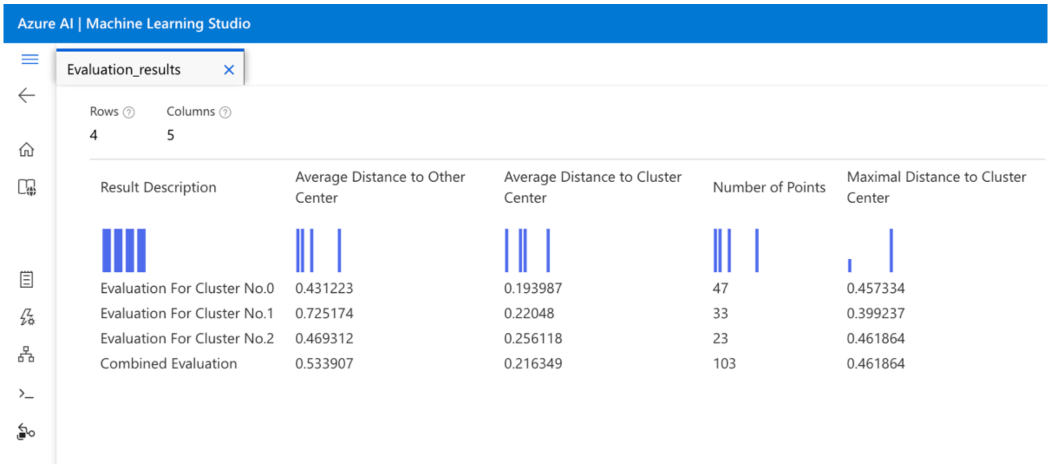
2- Clustering Model Pipeline Creation and Evaluation

- **Pipeline Setup:** Created a new pipeline in Azure Machine Learning named "Train Penguin Clustering" and selected the appropriate compute target.
- **Data Preparation:** Created, dragged and explored the "penguin-data" dataset, addressing missing values and normalizing numeric columns using data transformations.
- **Model Training:** Configured the pipeline to split the data into training and testing sets, applied the K-Means Clustering algorithm with three centroids to group the data into clusters, and trained the clustering model.
- **Model Evaluation:** Added an evaluation module to assess the model's performance using metrics such as average distance to cluster centers, average distance to other cluster centers, number of points per cluster, and maximal distance to cluster centers.



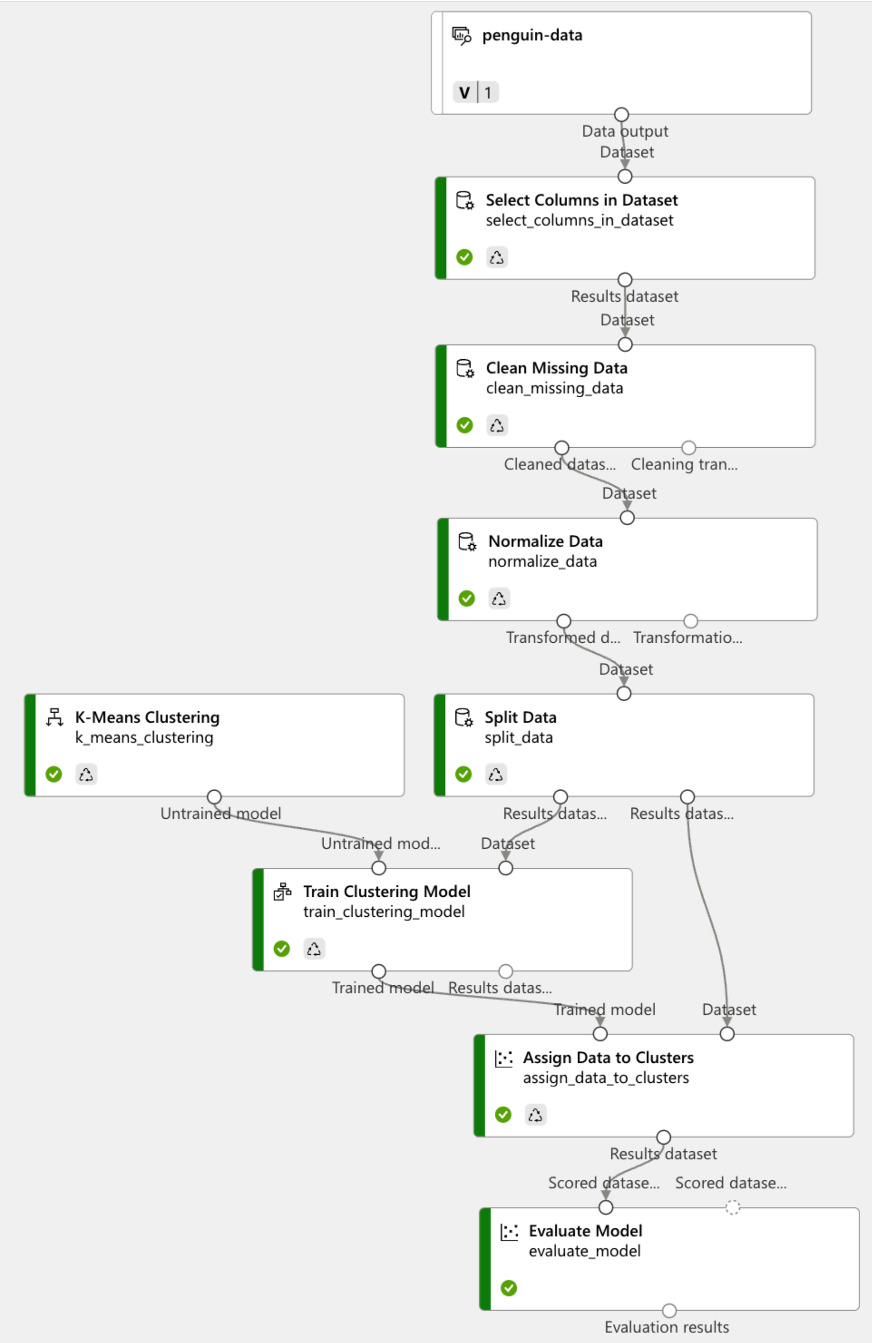
Rows 103 Columns 8

CulmenLength	CulmenDepth	FlipperLength	BodyMass	Assignments	DistancesToClusterCenter no.0	DistancesToClusterCenter no.1	DistancesToClusterCenter no.2
0.669091	0.77381	0.491525	0.375	2	0.561008	0.721252	0.182869
0.272727	0.904762	0.322034	0.333333	0	0.359393	0.966324	0.370271
0.803636	0.916667	0.491525	0.444444	2	0.750067	0.844225	0.387859
0.723636	0.904762	0.644068	0.583333	2	0.814904	0.737828	0.461864
0.552727	0.083333	0.745763	0.5625	1	0.867794	0.186619	0.741408
0.512727	0.119048	0.762712	0.465278	1	0.802737	0.251597	0.690423
0.450909	0.142857	0.745763	0.388889	1	0.73246	0.337794	0.653814
0.269091	0.511905	0.237288	0.305556	0	0.109911	0.802731	0.378083
0.487273	0.071429	0.711864	0.541667	1	0.82246	0.229968	0.730463
0.298182	0.583333	0.389831	0.152778	0	0.182915	0.831222	0.338698
0.52	0.297619	0.830508	0.638889	1	0.848638	0.11177	0.664607



Rows 4 Columns 5

Result Description	Average Distance to Other Center	Average Distance to Cluster Center	Number of Points	Maximal Distance to Cluster Center
Evaluation For Cluster No.0	0.431223	0.193987	47	0.457334
Evaluation For Cluster No.1	0.725174	0.22048	33	0.399237
Evaluation For Cluster No.2	0.469312	0.256118	23	0.461864
Combined Evaluation	0.533907	0.216349	103	0.461864

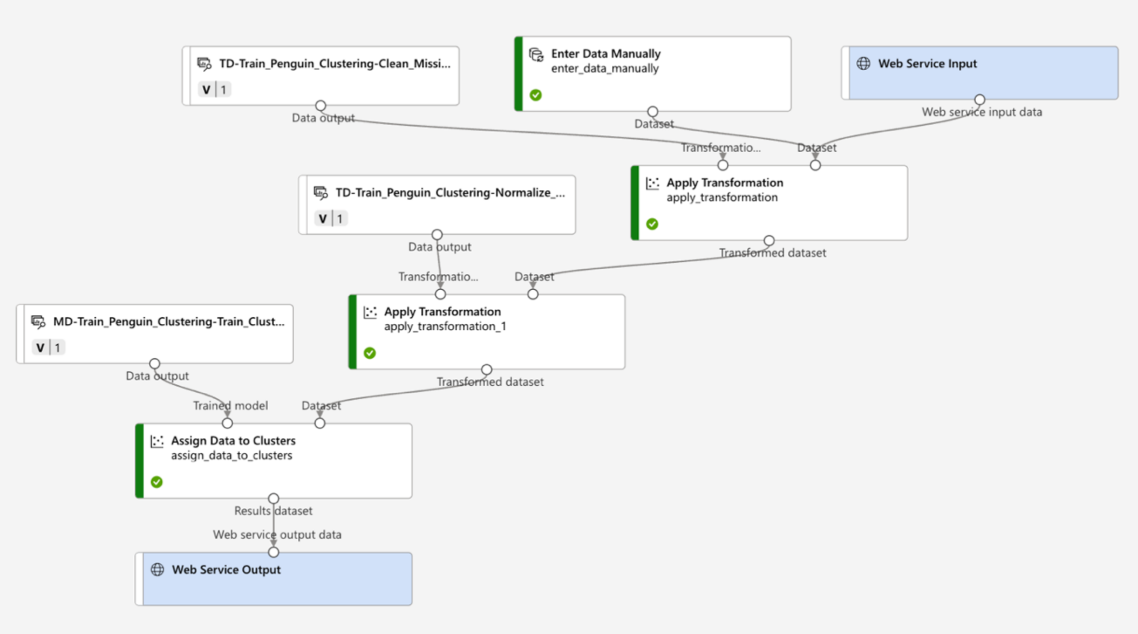


3- Inference Pipeline Creation

- **Pipeline Setup:** Opened the previously created " Train Penguin Clustering" pipeline and selected the "Real-time inference pipeline" option, creating a new pipeline named "Predict Penguin Cluster".
- **Pipeline Modifications:** Replaced the "penguin-data" dataset with an "Enter Data Manually" module that excludes the Species column and removed unnecessary modules. Connected the Web Service Input and "Enter Data Manually" modules to the input of the first Apply Transformation module, and connected the outputs from both the Web Service Input and "Enter Data Manually" modules to the Dataset input of the first Apply Transformation module.

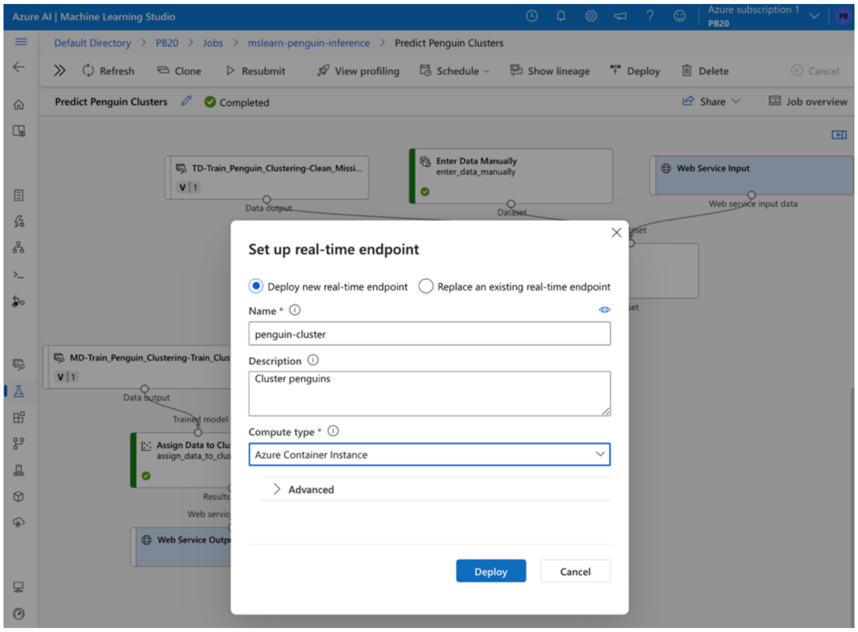
- **Execution and Validation:** Submitted the pipeline as a new experiment on the compute cluster. Visualized the "Results dataset" output from the "Assign Data to Clusters" module to review the predicted cluster assignments for the new penguin observations entered in "Enter Data Manually" module.

The inference pipeline prepares new data and applies the trained clustering model to assign data to clusters.



4- Deploying a Clustering Service

Deployed the "Predict Penguin Cluster" inference pipeline by selecting "Deploy" and creating a new real-time endpoint named "penguin-cluster" on Azure Container Instance (ACI). This deployment allows for development and testing purposes.



Set up real-time endpoint

● Deploy new real-time endpoint ○ Replace an existing real-time endpoint

Name * : penguin-cluster

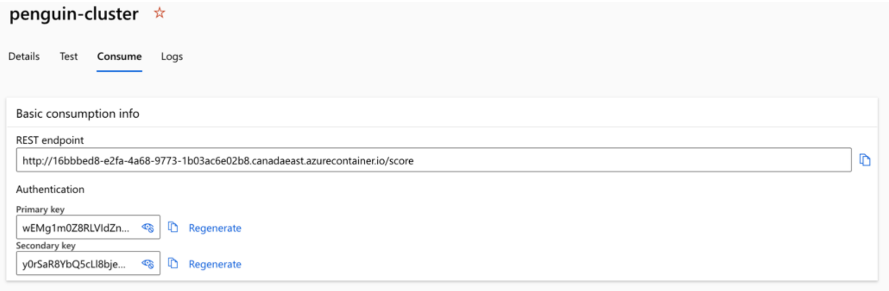
Description: Cluster penguins

Compute type * : Azure Container Instance

> Advanced

5- Real-Time Endpoint Testing

Accessed the "penguin-cluster" endpoint on the Endpoints page to retrieve the REST endpoint and Primary Key.



Basic consumption info

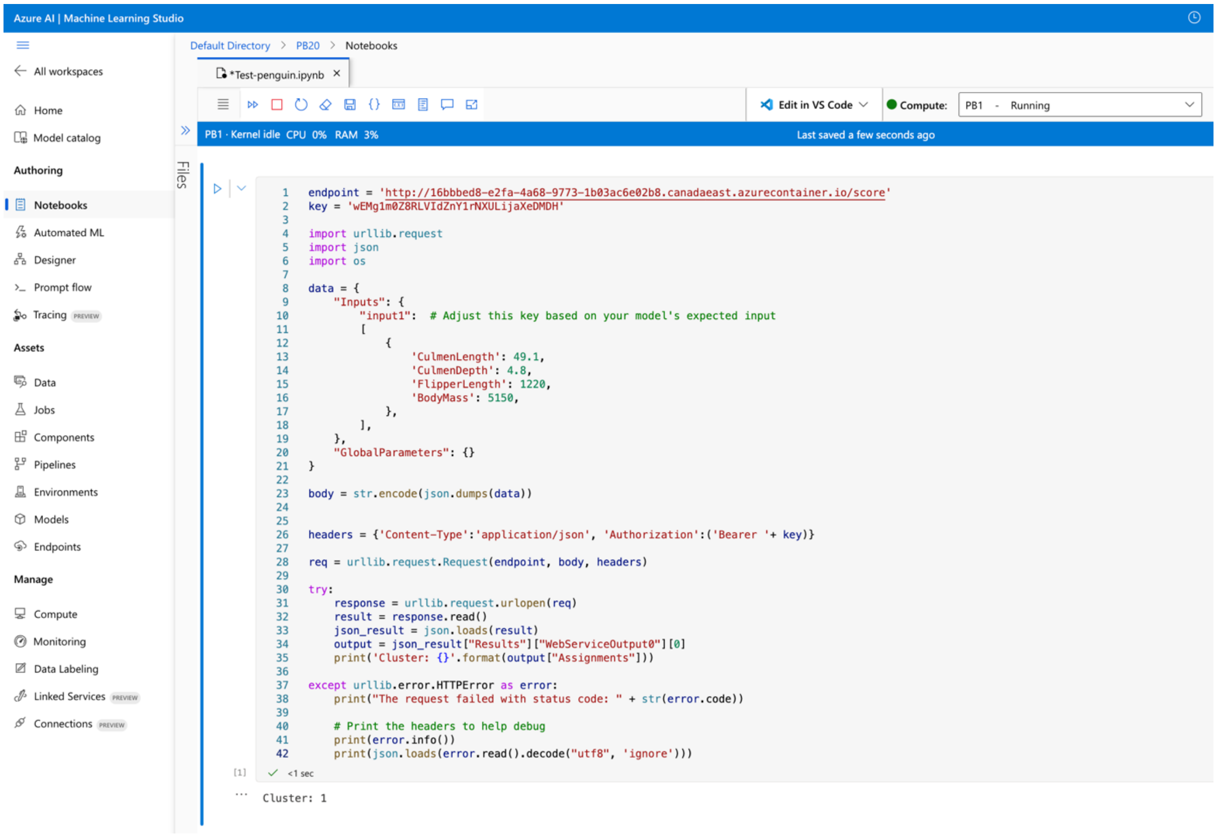
REST endpoint: http://16bbbed8-e2fa-4a68-9773-1b03ac6e02b8.canadaeast.azurecontainer.io/score

Authentication

Primary key: wEMg1m0Z8RLVIdZn...

Secondary key: y0rSaR8YbQ5cLI8bje...

Tested the deployed service by retrieving the REST endpoint and Primary Key, then using these details in a new notebook within Azure Machine Learning Studio to run a test and confirm that the service accurately assigns data to clusters.



```python
endpoint = 'http://16bbbed8-e2fa-4a68-9773-1b03ac6e02b8.canadaeast.azurecontainer.io/score'
key = 'wEMg1m0Z8RLVIdZnY1rNXULijaXeDMDH'

import urllib.request
import json
import os

data = {
    "Inputs": {
        "input1":  # Adjust this key based on your model's expected input
        [
            {
                'CulmenLength': 49.1,
                'CulmenDepth': 4.8,
                'FlipperLength': 1220,
                'BodyMass': 5150,
            },
        ],
    },
    "GlobalParameters": {}
}

body = str.encode(json.dumps(data))


headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ key)}

req = urllib.request.Request(endpoint, body, headers)

try:
    response = urllib.request.urlopen(req)
    result = response.read()
    json_result = json.loads(result)
    output = json_result["Results"]["WebServiceOutput0"][0]
    print('Cluster: {}'.format(output["Assignments"]))

except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))

    # Print the headers to help debug
    print(error.info())
    print(json.loads(error.read().decode("utf8", 'ignore')))
```

```
Cluster: 1
```

The deployment and testing process ensures that the clustering service is operational and accessible for client applications, delivering real-time cluster assignments based on the trained model.