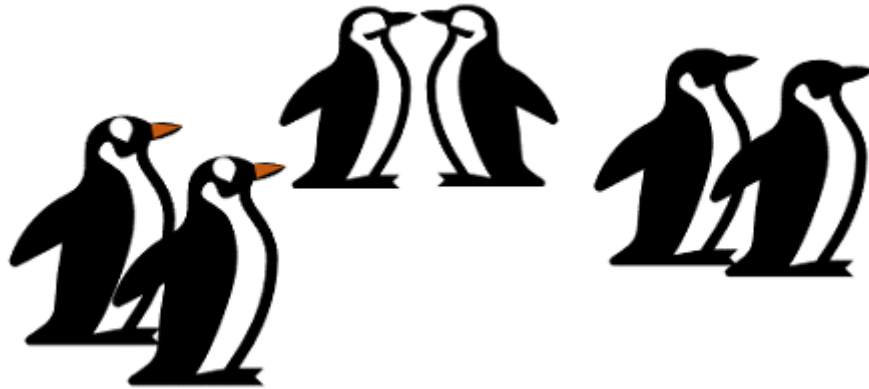


Introduction

Clustering is a form of machine learning that is used to group similar items into clusters based on their features. For example, a researcher might take measurements of penguins, and group them based on similarities in their proportions.



Clustering is an example of *unsupervised* machine learning, in which you train a model to separate items into clusters based purely on their characteristics, or *features*. There is no previously known cluster value (or *label*) from which to train the model.

You can use Microsoft Azure Machine Learning designer to create clustering models by using a drag and drop visual interface, without needing to write any code.

In this module, you'll learn how to:

- Use Azure Machine Learning designer to train a clustering model.
- Use a clustering model for inferencing.
- Deploy a clustering model as a service.

To complete this module, you'll need a Microsoft Azure subscription. If you don't already have one, you can sign up for a free trial at <https://azure.microsoft.com>.

Create an Azure Machine Learning workspace

Azure Machine Learning is a cloud-based platform for building and operating machine learning solutions in Azure. It includes a wide range of features and capabilities that help data scientists prepare data, train models, publish predictive services, and monitor their usage. One of these features is a visual interface called *designer*, that you can use to train, test, and deploy machine learning models without writing any code.

Create an Azure Machine Learning workspace

To use Azure Machine Learning, you create a *workspace* in your Azure subscription. You can then use this workspace to manage data, compute resources, code, models, and other artifacts related to your machine learning workloads.

Note

This module is one of many that make use of an Azure Machine Learning workspace. If you are completing this module in preparation for the [Azure AI Fundamentals](#) or [Azure Data Scientist](#) certification, consider creating the workspace once and reusing it in other modules. After completing each module, be sure to follow the **Clean Up** instructions at the end of the module to stop compute resources.

If you do not already have one, follow these steps to create a workspace:

1. Sign into the [Azure portal](#) using your Microsoft credentials.
2. Select **+Create a resource**, search for *Machine Learning*, and create a new **Machine Learning** resource the following settings:
 - **Subscription:** *Your Azure subscription*
 - **Resource group:** *Create or select a resource group*
 - **Workspace name:** *Enter a unique name for your workspace*
 - **Region:** *Select the geographical region closest to you*
 - **Storage account:** *Note the default new storage account that will be created for your workspace*
 - **Key vault:** *Note the default new key vault that will be created for your workspace*
 - **Application insights:** *Note the default new application insights resource that will be created for your workspace*

- **Container registry:** None (*one will be created automatically the first time you deploy a model to a container*)
- 3. Wait for your workspace to be created (it can take a few minutes). Then go to it in the portal.
- 4. On the **Overview** page for your workspace, launch Azure Machine Learning Studio (or open a new browser tab and navigate to <https://ml.azure.com>), and sign into Azure Machine Learning studio using your Microsoft account.
- 5. In Azure Machine Learning studio, toggle the ☰ icon at the top left to view the various pages in the interface. You can use these pages to manage the resources in your workspace.

You can manage your workspace using the Azure portal, but for data scientists and Machine Learning operations engineers, Azure Machine Learning studio provides a more focused user interface for managing workspace resources.

Create compute resources

To train and deploy models using Azure Machine Learning designer, you need compute on which to run the training process, and to test the trained model after deploying it.

Create compute targets

Compute targets are cloud-based resources on which you can run model training and data exploration processes.

1. In [Azure Machine Learning studio](#), view the **Compute** page (under **Manage**). This is where you manage the compute targets for your data science activities. There are four kinds of compute resource you can create:
 - **Compute Instances:** Development workstations that data scientists can use to work with data and models.
 - **Compute Clusters:** Scalable clusters of virtual machines for on-demand processing of experiment code.
 - **Inference Clusters:** Deployment targets for predictive services that use your trained models.
 - **Attached Compute:** Links to existing Azure compute resources, such as Virtual Machines or Azure Databricks clusters.
2. On the **Compute Instances** tab, add a new compute instance with the following settings. You'll use this as a workstation from which to test your model:

- **Virtual Machine type:** CPU
 - **Virtual Machine size:** Standard_DS11_v2 (Choose **Select from all options** to search for and select this machine size)
 - **Compute name:** *enter a unique name*
 - **Enable SSH access:** Unselected
3. While the compute instance is being created, switch to the **Compute Clusters** tab, and add a new compute cluster with the following settings. You'll use this to train a machine learning model:
- **Virtual Machine priority:** Dedicated
 - **Virtual Machine type:** CPU
 - **Virtual Machine size:** Standard_DS11_v2 (Choose **Select from all options** to search for and select this machine size)
 - **Compute name:** *enter a unique name*
 - **Minimum number of nodes:** 0
 - **Maximum number of nodes:** 2
 - **Idle seconds before scale down:** 120
 - **Enable SSH access:** Unselected

Note

If you decide not to complete this module, be sure to stop your compute instance to avoid incurring unnecessary charges to your Azure subscription.

The compute targets will take some time to be created. You can move onto the next unit while you wait.

Explore data

To train a clustering model, you need a dataset that includes multiple observations of the items you want to cluster, including numeric features that can be used to determine similarities between individual cases that will help separate them into clusters.

Create a dataset

In Azure Machine Learning, data for model training and other operations is usually encapsulated in an object called a *dataset*. In this module, you'll use a dataset that includes observations of three species of penguin.


1. In [Azure Machine Learning studio](#), view the **Datasets** page. Datasets represent specific data files or tables that you plan to work with in Azure ML.
2. Create a dataset from web files, using the following settings:
 - **Basic Info:**
 - **Web URL:** <https://aka.ms/penguin-data>
 - **Name:** penguin-data
 - **Dataset type:** Tabular
 - **Description:** Penguin data
 - **Settings and preview:**
 - **File format:** Delimited
 - **Delimiter:** Comma
 - **Encoding:** UTF-8
 - **Column headers:** Use headers from the first file
 - **Skip rows:** None
 - **Schema:**
 - Include all columns other than **Path**
 - Review the automatically detected types
 - **Confirm details:**
 - Do not profile the dataset after creation
3. After the dataset has been created, open it and view the **Explore** page to see a sample of the data. This data represents measurements of the culmen (bill) length and depth, flipper length, and body mass for multiple observations of penguins. There are three species of penguin represented in the dataset: *Adelie*, *Gentoo*, and *Chinstrap*.

Note

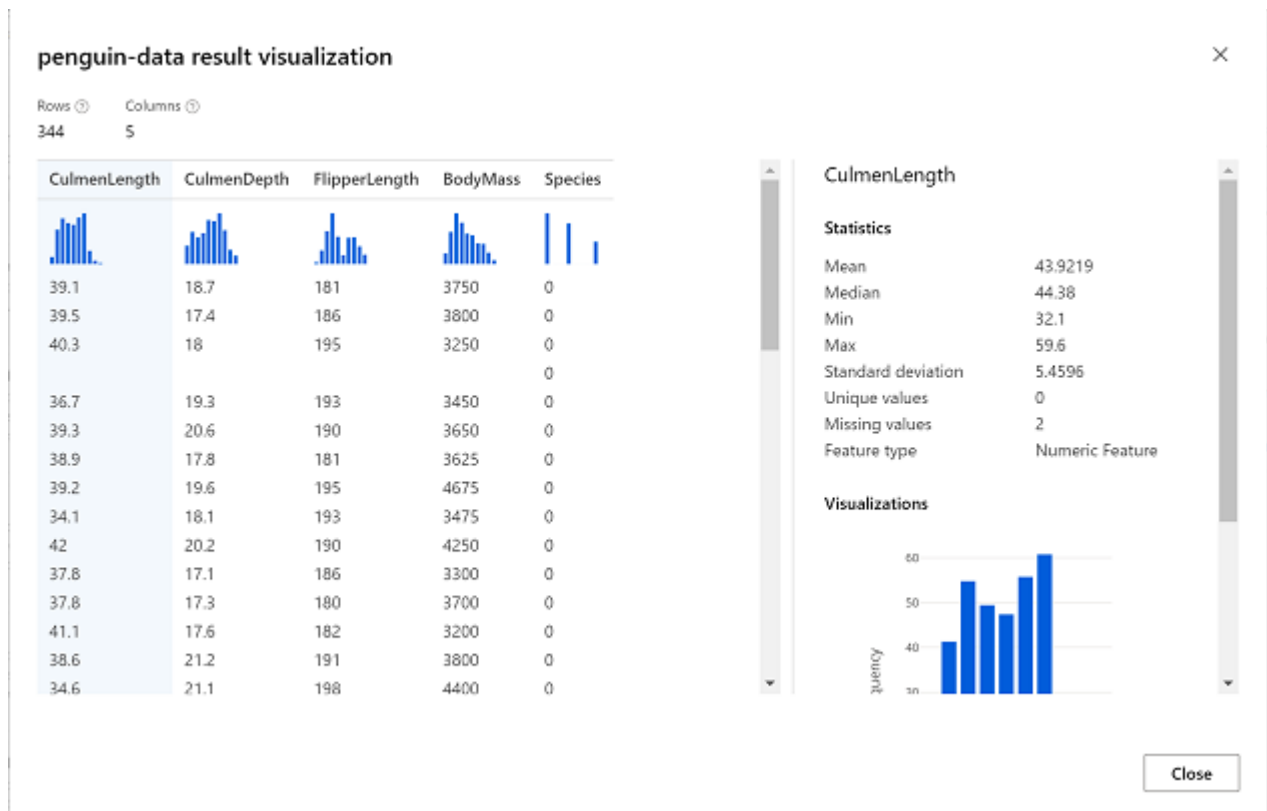
The penguins dataset used in the this exercise is a subset of data collected and made available by [Dr. Kristen Gorman](#) and the [Palmer Station, Antarctica LTER](#), a member of the [Long Term Ecological Research Network](#).

Create a pipeline

To get started with Azure machine Learning designer, first you must create a pipeline and add the dataset you want to work with.

1. In [Azure Machine Learning studio](#) for your workspace, view the **Designer** page and create a new pipeline.
2. In the **Settings** pane, change the default pipeline name (**Pipeline-Created-on-date**) to **Train Penguin Clustering** (if the **Settings** pane is not visible, click the  icon next to the pipeline name at the top).
3. Note that you need to specify a compute target on which to run the pipeline. In the **Settings** pane, click **Select compute target** and select the compute cluster you created previously.

4. In the pane on the left side of the designer, expand the **Datasets** section, and drag the **penguin-data** dataset you created in the previous exercise onto the canvas.
5. Right-click (Ctrl+click on a Mac) the **penguin-data** dataset on the canvas, and on the **Visualize** menu, select **Dataset output**.
6. Review the schema of the data, noting that you can see the distributions of the various columns as histograms. Then select the **CulmenLength** column. The dataset should look similar to this:

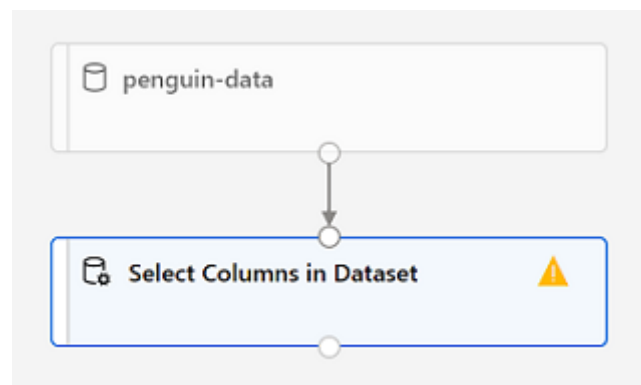


7. Note the following characteristics of the dataset:
 - The dataset includes the following columns:
 - **CulmenLength**: Length of the penguin's bill in millimeters.
 - **CulmenDepth**: Depth of the penguin's bill in millimeters.
 - **FlipperLength**: Length of the penguin's flipper in millimeters.
 - **BodyMass**: Weight of the penguin in grams.
 - **Species**: Species indicator (0:"Adelie", 1:"Gentoo", 2:"Chinstrap")
 - There are two missing values in the **CulmenLength** column (the **CulmenDepth**, **FlipperLength**, and **BodyMass** columns also have two missing values).
 - The measurement values are in different scales (from tens of millimeters to thousands of grams).
8. Close the dataset visualization so you can see the dataset on the pipeline canvas.

Apply transformations

To cluster the penguin observations, we're going to use only the measurements; so we'll discard the species column. We also need to remove rows where values are missing, and normalize the numeric measurement values so they're on a similar scale.

1. In the pane on the left, expand the **Data Transformation** section, which contains a wide range of modules you can use to transform data before model training.
2. To cluster the penguin observations, we're going to use only the measurements - we'll ignore the species column. So, drag a **Select Columns in Dataset** module to the canvas, below the **penguin-data** module and connect the output at the bottom of the **penguin-data** module to the input at the top of the **Select Columns in Dataset** module, like this:



3. Select the **Select Columns in Dataset** module, and in its **Settings** pane on the right, select **Edit column**. Then in the **Select columns** window, select **By name** and use the + links to select the column names **CulmenLength**, **CulmenDepth**, **FlipperLength**, and **BodyMass**; like this:

Select columns

×

Select columns

☐ With rules
 ☒ By name

Available columns

All types ▾ 🔍 Search

1 Columns

Species

Add all +

Selected columns

All types ▾ 🔍 Search

4 Columns

CulmenLength

CulmenDepth

FlipperLength

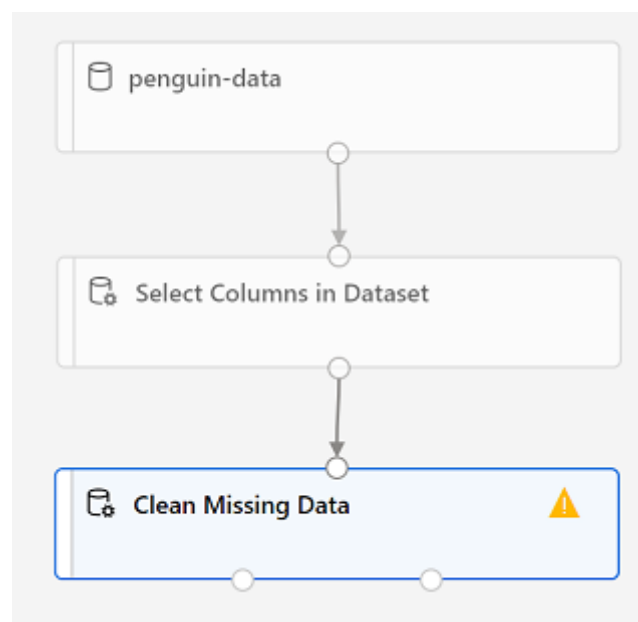
BodyMass

Remove all —

Save

Cancel

- Save the **Select Columns in a Dataset** module settings to return to the designer canvas.
- Drag a **Clean Missing Data** module to the canvas, below the **Select columns in a dataset** module and connect them like this:



- Select the **Clean Missing Data** module, and in the settings pane on the right, click **Edit column**. Then in the **Select columns** window, select **With rules** and include **All columns**; like this:

Columns to transform

Select columns

☒ With rules

☐ By name

Allow duplicates and preserve column order in selection

☐

Include

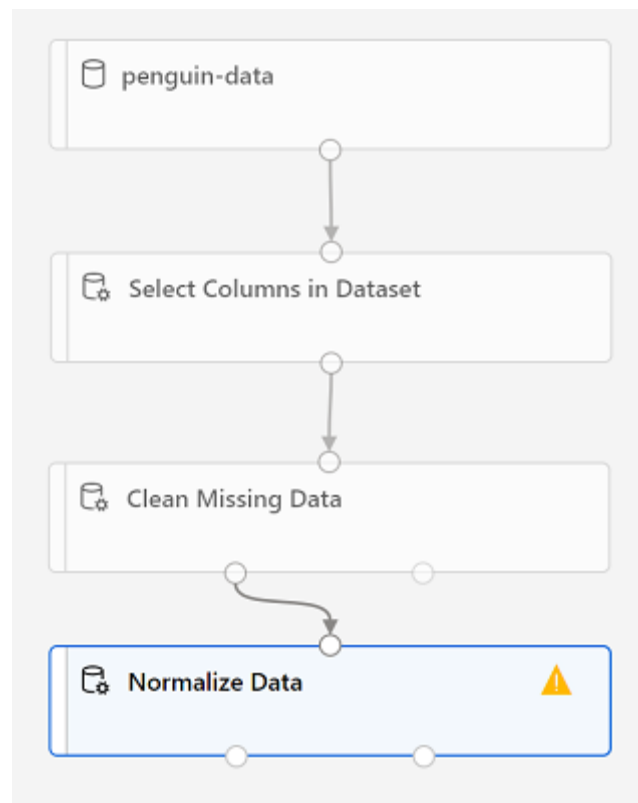
All columns

+

Save

Cancel

7. With the **Clean Missing Data** module still selected, in the settings pane, set the following configuration settings:
 - **Minimum missing value ratio:** 0.0
 - **Maximum missing value ratio:** 1.0
 - **Cleaning mode:** Remove entire row
8. Drag a **Normalize Data** module to the canvas, below the **Clean Missing Data** module. Then connect the left-most output from the **Clean Missing Data** module to the input of the **Normalize Data** module.



9. Select the **Normalize Data** module, and in its **Settings** pane on the right, set the **Transformation method** to **MinMax** and select **Edit column**. Then in the **Select columns** window, select **With rules** and include **All columns**; like this:

Columns to transform ×

Select columns ☒ With rules ☐ By name

Allow duplicates and preserve column order in selection ☒

Include All columns +

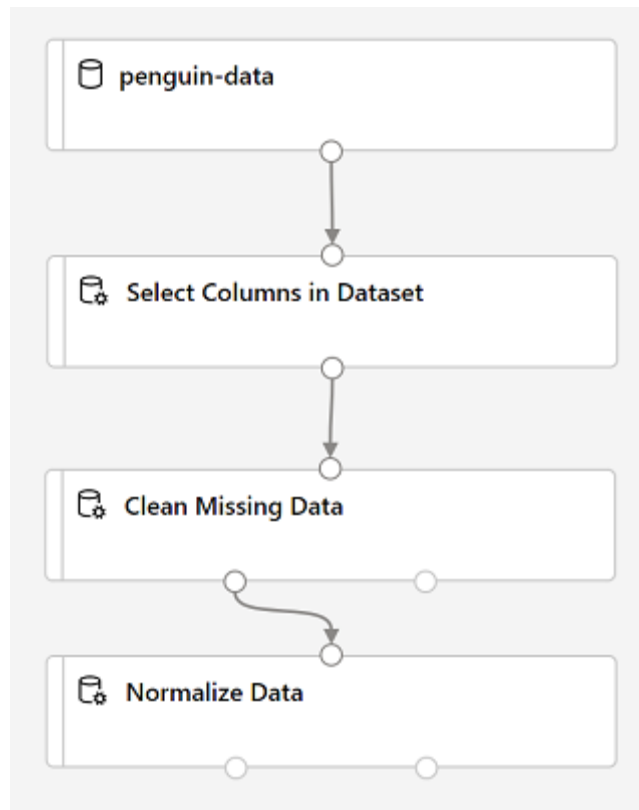
Save Cancel

10. Save the **Normalize Data** module settings to return to the designer canvas.

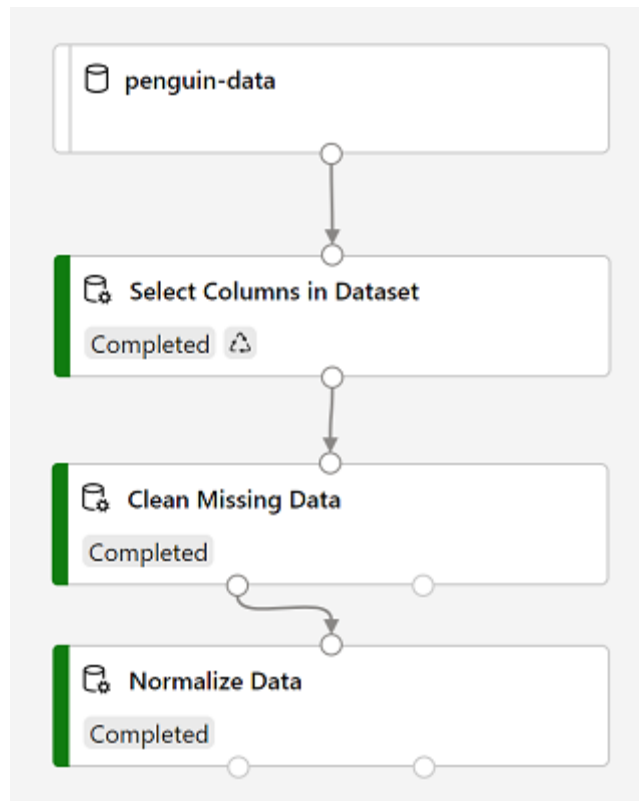
Run the pipeline

To apply your data transformations, you need to run the pipeline as an experiment.

1. Ensure your pipeline looks similar to this:



2. Select **Submit**, and run the pipeline as a new experiment named **mslearn-penguin-training** on your compute cluster.
3. Wait for the run to finish. This may take 5 minutes or more. When the run has completed, the modules should look like this:



View the transformed data

The dataset is now prepared for model training.

1. Select the completed **Normalize Data** module, and in its **Settings** pane on the right, on the **Outputs + logs** tab, select the **Visualize** icon for the **Transformed dataset**.
2. View the data, noting that the **Species** column has been removed, there are no missing values, and the values for all four features have been normalized to a common scale.
3. Close the normalized data result visualization.

Now that you have selected and prepared the features you want to use from the dataset, you're ready to use them to train a clustering model.

Create and run a training pipeline

Completed100 XP

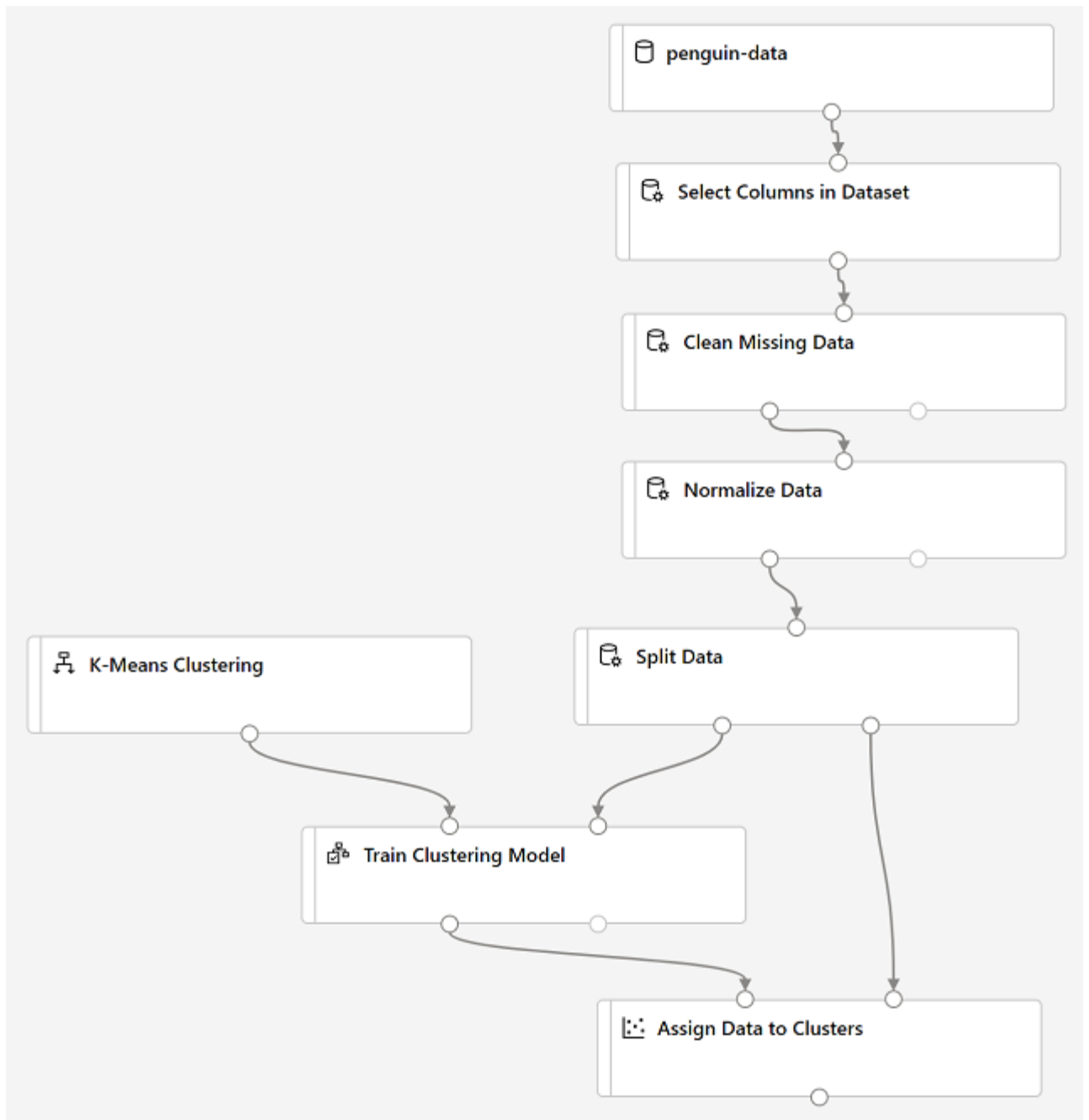
- 8 minutes

After you've used data transformations to prepare the data, you can use it to train a machine learning model.

Add training modules

To train a clustering model, you need to apply a clustering algorithm to the data, using only the features that you have selected for clustering. You'll train the model with a subset of the data, and use the rest to test the trained model.

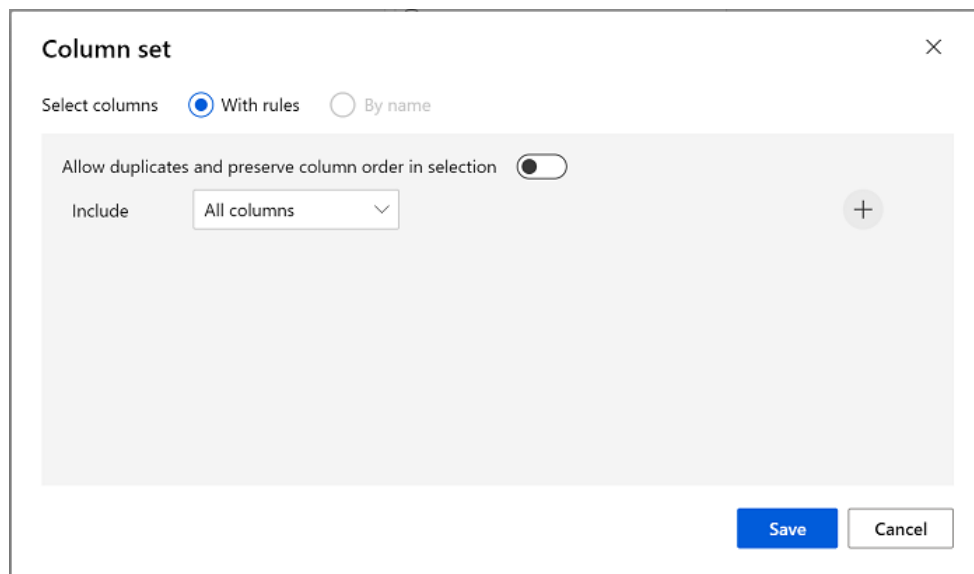
In this exercise, you're going to extend the **Train Penguin Clustering** pipeline as shown here:



Follow the steps below, using the image above for reference as you add and configure the required modules.

1. Open the **Train Penguin Clustering** pipeline, if it's not already open.
2. In the pane on the left, in the **Data Transformations** section, drag a **Split Data** module onto the canvas under the **Normalize Data** module. Then connect the left output of the **Normalize Data** module to the input of the **Split Data** module.
3. Select the **Split Data** module, and configure its settings as follows:
 - **Splitting mode:** Split Rows
 - **Fraction of rows in the first output dataset:** 0.7

- **Random seed:** 123
 - **Stratified split:** False
4. Expand the **Model Training** section in the pane on the left, and drag a **Train Clustering Model** module to the canvas, under the **Split Data** module. Then connect the *Result dataset1* (left) output of the **Split Data** module to the *Dataset* (right) input of the **Train Clustering Model** module.
 5. The clustering model should assign clusters to the data items by using all of the features you selected from the original dataset. Select the **Train Clustering Model** module and in its settings pane, on the **Parameters** tab, select **Edit Columns** and use the **With rules** option to include all columns; like this:



6. The model we're training will use the features to group the data into clusters, so we need to train the model using a *clustering* algorithm. Expand the **Machine Learning Algorithms** section, and under **Clustering**, drag a **K-Means Clustering** module to the canvas, to the left of the **penguin-data** dataset and above the **Train Clustering Model** module. Then connect its output to the **Untrained model** (left) input of the **Train Clustering Model** module.
7. The *K-Means* algorithm groups items into the number of clusters you specify - a value referred to as **K**. Select the **K-Means Clustering** module and in its settings pane, on the **Parameters** tab, set the **Number of centroids** parameter to **3**.

Note

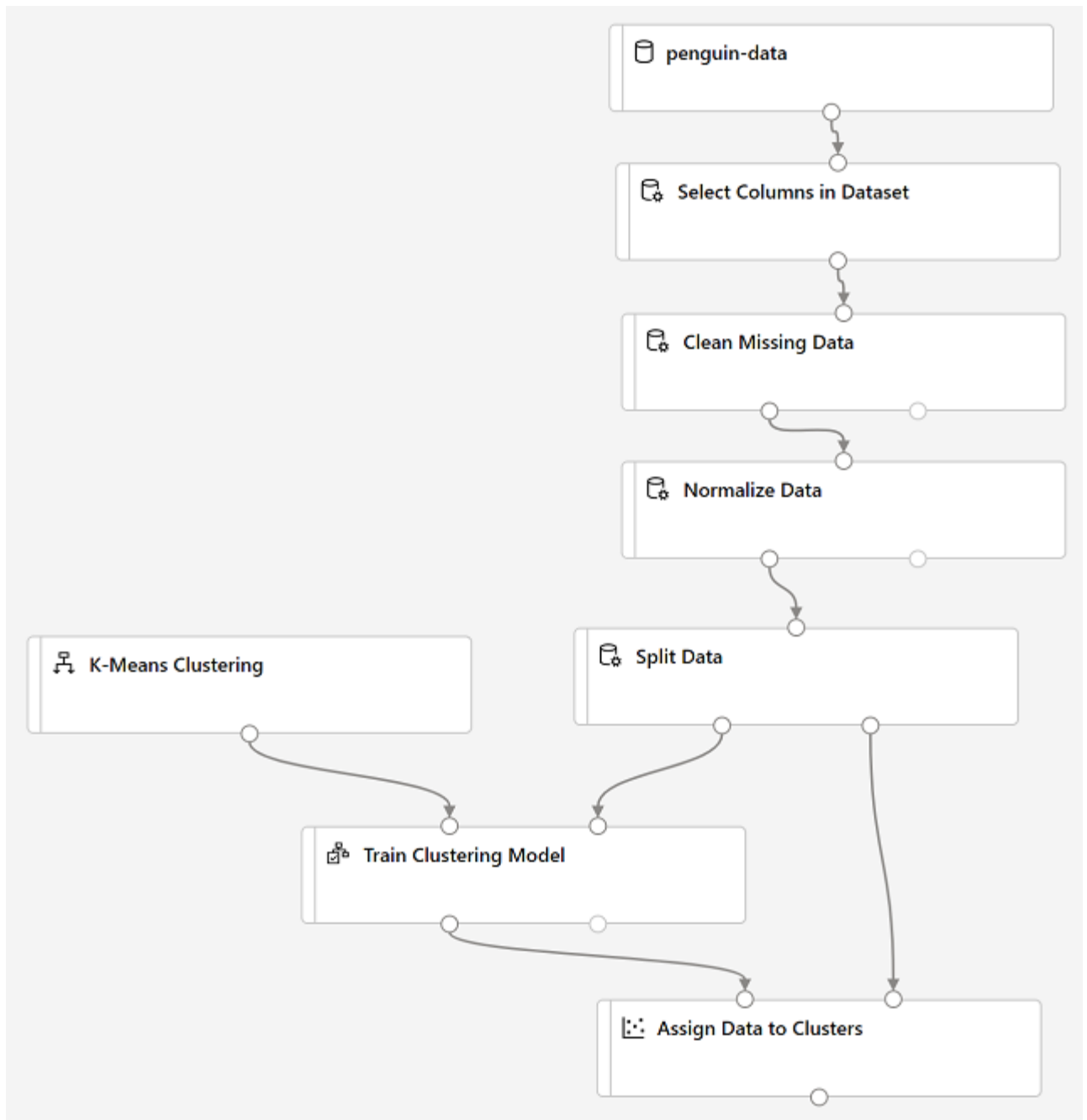
You can think of data observations, like the penguin measurements, as being multidimensional vectors. The K-Means algorithm works by:

1. initializing K coordinates as randomly selected points called *centroids* in n -dimensional space (where n is the number of dimensions in the feature vectors).
 2. Plotting the feature vectors as points in the same space, and assigning each point to its closest centroid.
 3. Moving the centroids to the middle of the points allocated to it (based on the *mean* distance).
 4. Reassigning the points to their closest centroid after the move.
 5. Repeating steps 3 and 4 until the cluster allocations stabilize or the specified number of iterations has completed.
8. After using 70% of the data to train the clustering model, you can use the remaining 30% to test it by using the model to assign the data to clusters. Expand the **Model Scoring & Evaluation** section and drag an **Assign Data to Clusters** module to the canvas, below the **Train Clustering Model** module. Then connect the **Trained model** (left) output of the **Train Clustering Model** module to the **Trained model** (left) input of the **Assign Data to Clusters** module; and connect the **Results dataset2** (right) output of the **Split Data** module to the **Dataset** (right) input of the **Assign Data to Clusters** module.

Run the training pipeline

Now you're ready to run the training pipeline and train the model.

1. Ensure your pipeline looks like this:



2. Select **Submit**, and run the pipeline using the existing experiment named **mslearn-penguin-training** on your compute cluster.
3. Wait for the experiment run to finish. This may take 5 minutes or more.
4. When the experiment run has finished, select the **Assign Data to Clusters** module and in its settings pane, on the **Outputs + Logs** tab, under **Data outputs** in the **Results dataset** section, use the **Visualize** icon to view the results.
5. Scroll to the right, and note the **Assignments** column, which contains the cluster (0, 1, or 2) to which each row is assigned. There are also new columns indicating the distance from the point representing this row to the centers of

each of the clusters - the cluster to which the point is closest is the one to which it is assigned.

6. Close the **Assign Data to Clusters** visualization.

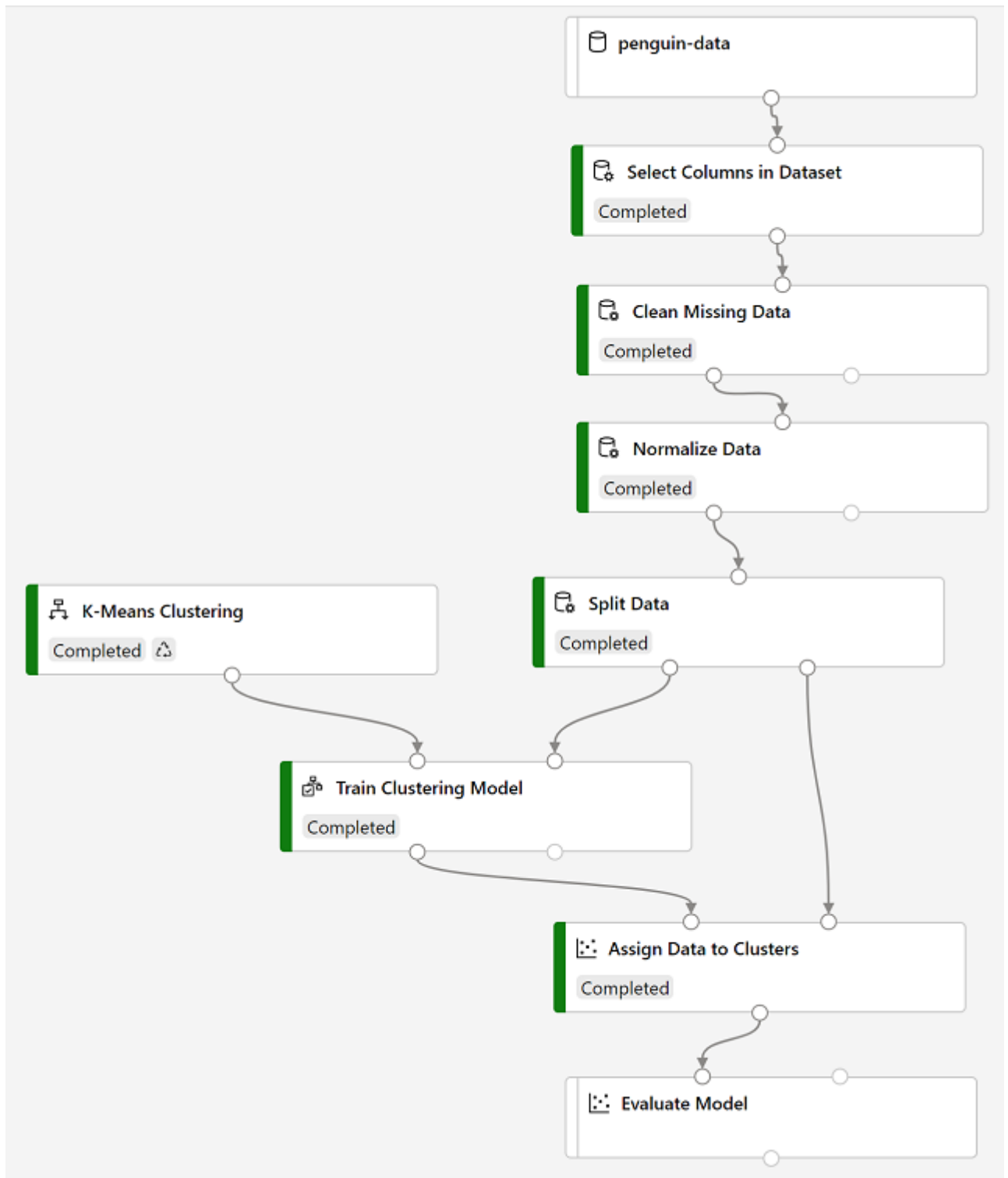
The model is predicting clusters for the penguin observations, but how reliable are its predictions? To assess that, you need to evaluate the model.

Evaluate a clustering model

Evaluating a clustering model is made difficult by the fact that there are no previously known *true* values for the cluster assignments. A successful clustering model is one that achieves a good level of separation between the items in each cluster, so we need metrics to help us measure that separation.

Add an Evaluate Model module

1. Open the **Train Penguin Clustering** pipeline you created in the previous unit if it's not already open.
2. In the pane on the left, in the **Model Scoring & Evaluation** section, drag an **Evaluate Model** module to the canvas, under the **Assign Data to Clusters** module, and connect the output of the **Assign Data to Clusters** module to the **Scored dataset** (left) input of the **Evaluate Model** module.
3. Ensure your pipeline looks like this:



4. Select **Submit**, and run the pipeline using the existing **mslearn-penguin-training** experiment.
5. Wait for the experiment run to finish.
6. When the experiment run has finished, select the **Evaluate Model** module and in the settings pane, on the **Outputs + Logs** tab, under **Data outputs** in the **Evaluation results** section, use the **Visualize** icon to view the performance metrics. These metrics can help data scientists assess how well the model

separates the clusters. They include a row of metrics for each cluster, and a summary row for a combined evaluation. The metrics in each row are:

- **Average Distance to Other Center:** This indicates how close, on average, each point in the cluster is to the centroids of all other clusters.
- **Average Distance to Cluster Center:** This indicates how close, on average, each point in the cluster is to the centroid of the cluster.
- **Number of Points:** The number of points assigned to the cluster.
- **Maximal Distance to Cluster Center:** The maximum of the distances between each point and the centroid of that point's cluster. If this number is high, the cluster may be widely dispersed. This statistic in combination with the **Average Distance to Cluster Center** helps you determine the cluster's *spread*.

7. Close the **Evaluate Model result visualization** window.

Now that you have a working clustering model, you can use it to assign new data to clusters in an *inference pipeline*.

Create an inference pipeline

After creating and running a pipeline to train the clustering model, you can create an *inference pipeline* that uses the model to assign new data observations to clusters. This will form the basis for a predictive service that you can publish for applications to use.

Create an inference pipeline

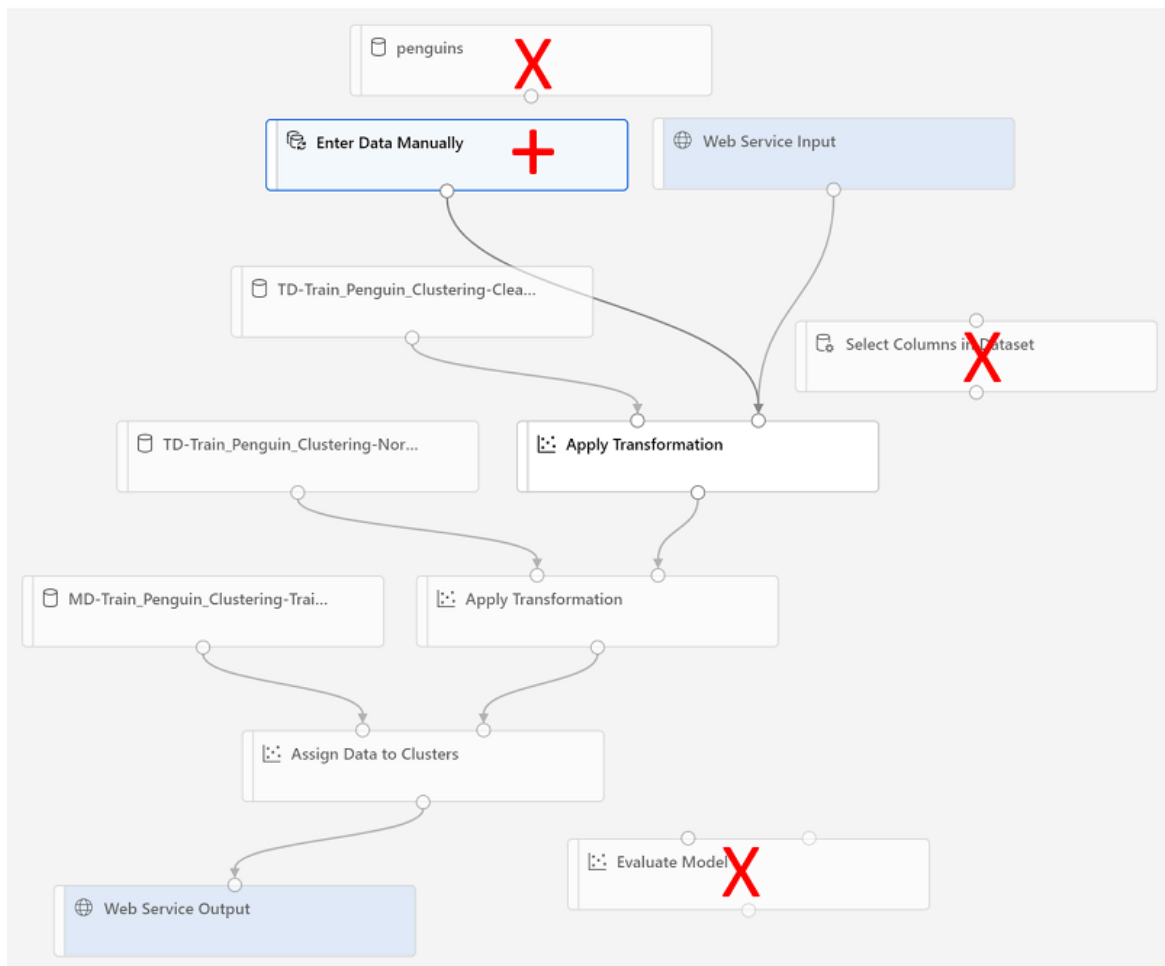
1. In Azure Machine Learning Studio, open the **Train Penguin Clustering** pipeline you created previously.
2. In the **Create inference pipeline** drop-down list, click **Real-time inference pipeline**. After a few seconds, a new version of your pipeline named **Train Penguin Clustering-real time inference** will be opened.

*If the pipeline does not include **Web Service Input** and **Web Service Output** modules, go back to the **Designer** page and then re-open the **Train Penguin Clustering-real time inference** pipeline.*

3. Rename the new pipeline to **Predict Penguin Clusters**, and then review the new pipeline. It contains a web service input for new data to be submitted, and a web service output to return results. The transformations and clustering

model in your training pipeline are encapsulated in this pipeline based on the statistics from your training data, and will be used to transform and score the new data.

You are going to make the following changes to the inference pipeline:



- Replace the **penguin-data** dataset with an **Enter Data Manually** module that does not include the **Species** column.
- Remove the **Select Columns in Dataset** module, which is now redundant.
- Connect the **Web Service Input** and **Enter Data Manually** modules (which represent inputs for data to be clustered) to the first **Apply Transformation** module.
- Remove the **Evaluate Model** module.

Follow the remaining steps below, using the image and information above for reference as you modify the pipeline.

4. The inference pipeline assumes that new data will match the schema of the original training data, so the **penguin-data** dataset from the training pipeline is included. However, this input data includes a column for the penguin species,

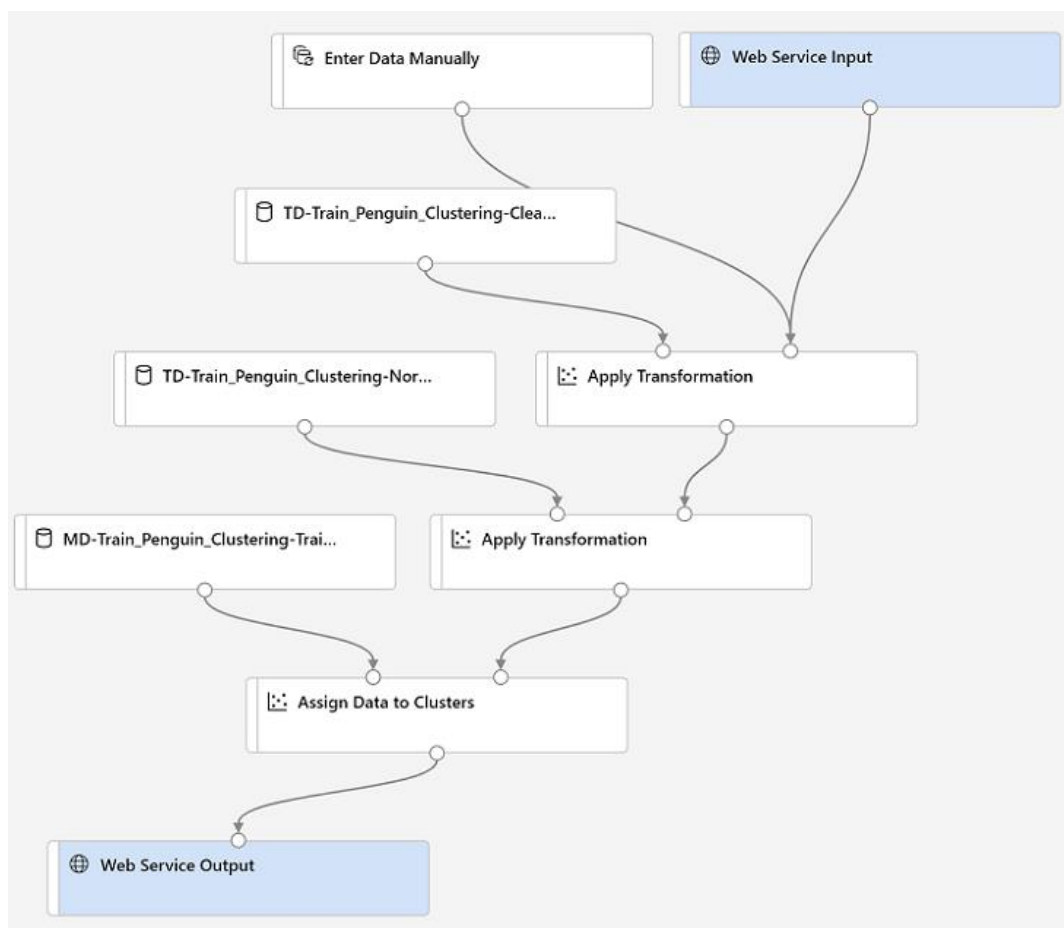
which the model does not use. Delete both the **penguin-data** dataset and the **Select Columns in Dataset** modules, and replace them with an **Enter Data Manually** module from the **Data Input and Output** section. Then modify the settings of the **Enter Data Manually** module to use the following CSV input, which contains feature values for three new penguin observations (including headers):

```

CSVCopy
CulmenLength,CulmenDepth,FlipperLength,BodyMass
39.1,18.7,181,3750
49.1,14.8,220,5150
46.6,17.8,193,3800

```

5. Connect the outputs from both the **Web Service Input** and **Enter Data Manually** modules to the Dataset (right) input of the first **Apply Transformation** module.
6. Delete the **Evaluate Model** module.
7. Verify that your pipeline looks similar to the following:



8. Submit the pipeline as a new experiment named **mslearn-penguin-inference** on your compute cluster. This may take a while!

9. When the pipeline has finished, visualize the **Results dataset** output of the **Assign Data to Clusters** module to see the predicted cluster assignments and metrics for the three penguin observations in the input data.

Your inference pipeline assigns penguin observations to clusters based on their features. Now you're ready to publish the pipeline so that client applications can use it.

Deploy a predictive service

After you've created and tested an inference pipeline for real-time inferencing, you can publish it as a service for client applications to use.

Note: In this exercise, you'll deploy the web service to to an Azure Container Instance (ACI). This type of compute is created dynamically, and is useful for development and testing. For production, you should create an *inference cluster*, which provide an Azure Kubernetes Service (AKS) cluster that provides better scalability and security.




Deploy a service

1. View the **Predict Penguin Clusters** inference pipeline you created in the previous unit.
2. At the top right, select **Deploy**, and deploy a new real-time endpoint, using the following settings:
 - **Name:** predict-penguin-clusters
 - **Description:** Cluster penguins.
 - **Compute type:** Azure Container Instance
3. Wait for the web service to be deployed - this can take several minutes. The deployment status is shown at the top left of the designer interface.

Test the service

Now you can test your deployed service from a client application - in this case, you'll use the code in the cell below to simulate a client application.

1. On the **Endpoints** page, open the **predict-penguin-clusters** real-time endpoint.

2. When the **predict-penguin-clusters** endpoint opens, view the **Consume** tab and note the following information there. You need this to connect to your deployed service from a client application.
 - The REST endpoint for your service
 - the Primary Key for your service
3. Note that you can use the  link next to these values to copy them to the clipboard.
4. With the **Consume** page for the **predict-penguin-clusters** service page open in your browser, open a new browser tab and open a second instance of [Azure Machine Learning studio](#). Then in the new tab, view the **Notebooks** page (under **Author**).
5. In the **Notebooks** page, under **My files**, use the  button to create a new file with the following settings:
 - **File location:** Users/*your user name*
 - **File name:** Test-Penguins
 - **File type:** Notebook
 - **Overwrite if already exists:** Selected
6. When the new notebook has been created, ensure that the compute instance you created previously is selected in the **Compute** box, and that it has a status of **Running**.
7. Use the  button to collapse the file explorer pane and give you more room to focus on the **Test-Penguins.ipynb** notebook tab.
8. In the rectangular cell that has been created in the notebook, paste the following code:

```
PythonCopy
endpoint = 'YOUR_ENDPOINT' #Replace with your endpoint
key = 'YOUR_KEY' #Replace with your key

import urllib.request
import json
import os

data = {
    "Inputs": {
        "WebServiceInput0":
            [
                {
                    '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')))
```

Note

Don't worry too much about the details of the code. It just defines features for a penguin, and uses the **predict-penguin-clusters** service you created to predict a cluster assignment.

9. Switch to the browser tab containing the **Consume** page for the **predict-penguin-clusters** service, and copy the REST endpoint for your service. The switch back to the tab containing the notebook and paste the key into the code, replacing YOUR_ENDPOINT.
10. Switch to the browser tab containing the **Consume** page for the **predict-penguin-clusters** service, and copy the Primary Key for your service. The switch back to the tab containing the notebook and paste the key into the code, replacing YOUR_KEY.
11. Save the notebook, Then use the ► button next to the cell to run the code.
12. Verify that predicted cluster is returned.


Summary

In this module, you learned how to use Azure Machine Learning designer to train and publish a clustering model.

Clean-up

The web service you created is hosted in an *Azure Container Instance*. If you don't intend to experiment with it further, you should delete the endpoint to avoid

accruing unnecessary Azure usage. You should also stop the compute instance until you need it again.

1. In [Azure Machine Learning studio](#), on the **Endpoints** tab, select the **predict-penguin-clusters** endpoint. Then select **Delete** () and confirm that you want to delete the endpoint.
2. On the **Compute** page, on the **Compute Instances** tab, select your compute instance and then select **Stop**.

If you have finished exploring Azure Machine Learning, you can delete the resource group containing your Azure Machine Learning workspace from your Azure subscription:

1. In the [Azure portal](#), in the **Resource groups** page, open the resource group you specified when creating your Azure Machine Learning workspace.
2. Click **Delete resource group**, type the resource group name to confirm you want to delete it, and select **Delete**.