Provision an Azure AI Language resource

Configure role-based access for your user

Upload sample articles

Create a custom text classification project

Label your data

Train your model

Evaluate your model

Deploy your model

Prepare to develop an app in Cloud Shell

Configure your application

Add code to classify documents

Clean up

Custom text classification

Azure AI Language provides several NLP capabilities, including the key phrase identification, text summarization, and sentiment analysis. The Language service also provides custom features like custom question answering and custom text classification.

To test the custom text classification of the Azure AI Language service, you'll configure the model using Language Studio then use a Python application to test it.

While this exercise is based on Python, you can develop text classification applications using multiple language-specific SDKs; including:

- Azure AI Text Analytics client library for Python
- Azure AI Text Analytics client library for .NET
- Azure AI Text Analytics client library for JavaScript

This exercise takes approximately **35** minutes.

Provision an *Azure AI Language* resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI Language service** resource. Additionally, use custom text classification, you need to enable the **Custom text classification & extraction** feature.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Select **Create a resource**.
3. In the search field, search for **Language service**. Then, in the results, select **Create** under **Language Service**.
4. Select the box that includes **Custom text classification**. Then select **Continue to create your resource**.
5. Create a resource with the following settings:
    - **Subscription**: *Your Azure subscription*.
    - **Resource group**: *Select or create a resource group*.
    - **Region**: *Choose from one of the following regions**
        - Australia East
        - Central India
        - East US
        - East US 2
        - North Europe
        - South Central US
        - Switzerland North
        - UK South
        - West Europe
        - West US 2
        - West US 3
    - **Name**: *Enter a unique name*.
    - **Pricing tier**: Select **F0** (*free*), or **S** (*standard*) if F is not available.
    - **Storage account**: New storage account
        - **Storage account name**: *Enter a unique name*.
        - **Storage account type**: Standard LRS
    - **Responsible AI notice**: Selected.
6. Select **Review + create,** then select **Create** to provision the resource.

7. Wait for deployment to complete, and then go to the resource group.

8. Find the storage account you created, select it, and verify the *Account kind* is **StorageV2**. If it's v1, upgrade your storage account kind on that resource page.

Configure role-based access for your user

> **NOTE**: If you skip this step, you'll get a 403 error when trying to connect to your custom project. It's important that your current user has this role to access storage account blob data, even if you're the owner of the storage account.

1. Go to your storage account page in the Azure portal.
2. Select **Access Control (IAM)** in the left navigation menu.
3. Select **Add** to Add Role Assignments, and choose the **Storage Blob Data Owner** role on the storage account.
4. Within **Assign access to**, select **User, group, or service principal**.
5. Select **Select members**.
6. Select your User. You can search for user names in the **Select** field.

Upload sample articles

Once you've created the Azure AI Language service and storage account, you'll need to upload example articles to train your model later.

1. In a new browser tab, download sample articles from `https://aka.ms/classification-articles` and extract the files to a folder of your choice.

2. In the Azure portal, navigate to the storage account you created, and select it.

3. In your storage account select **Configuration**, located below **Settings**. In the Configuration screen enable the option to **Allow Blob anonymous access** then select **Save**.

4. Select **Containers** in the left menu, located below **Data storage**. On the screen that appears, select **+ Container**. Give the container the name `articles`, and set **Anonymous access level** to **Container (anonymous read access for containers and blobs)**.

   > **NOTE**: When you configure a storage account for a real solution, be careful to assign the appropriate access level. To learn more about each access level, see the Azure Storage documentation.

5. After you've created the container, select it then select the **Upload** button. Select **Browse for files** to browse for the sample articles you downloaded. Then select **Upload**.

Create a custom text classification project

After configuration is complete, create a custom text classification project. This project provides a working place to build, train, and deploy your model.

> **NOTE**: This lab utilizes **Language Studio**, but you can also create, build, train, and deploy your model through the REST API.

1. In a new browser tab, open the Azure AI Language Studio portal at `https://language.cognitive.azure.com/` and sign in using the Microsoft account associated with your Azure subscription.

2. If prompted to choose a Language resource, select the following settings:

   - **Azure Directory**: The Azure directory containing your subscription.
   - **Azure subscription**: Your Azure subscription.
   - **Resource type**: Language.

- **Language resource**: The Azure AI Language resource you created previously.

If you are not prompted to choose a language resource, it may be because you have multiple Language resources in your subscription; in which case:

a. On the bar at the top if the page, select the **Settings (⚙)** button.
b. On the **Settings** page, view the **Resources** tab.
c. Select the language resource you just created, and click **Switch resource**.
d. At the top of the page, click **Language Studio** to return to the Language Studio home page

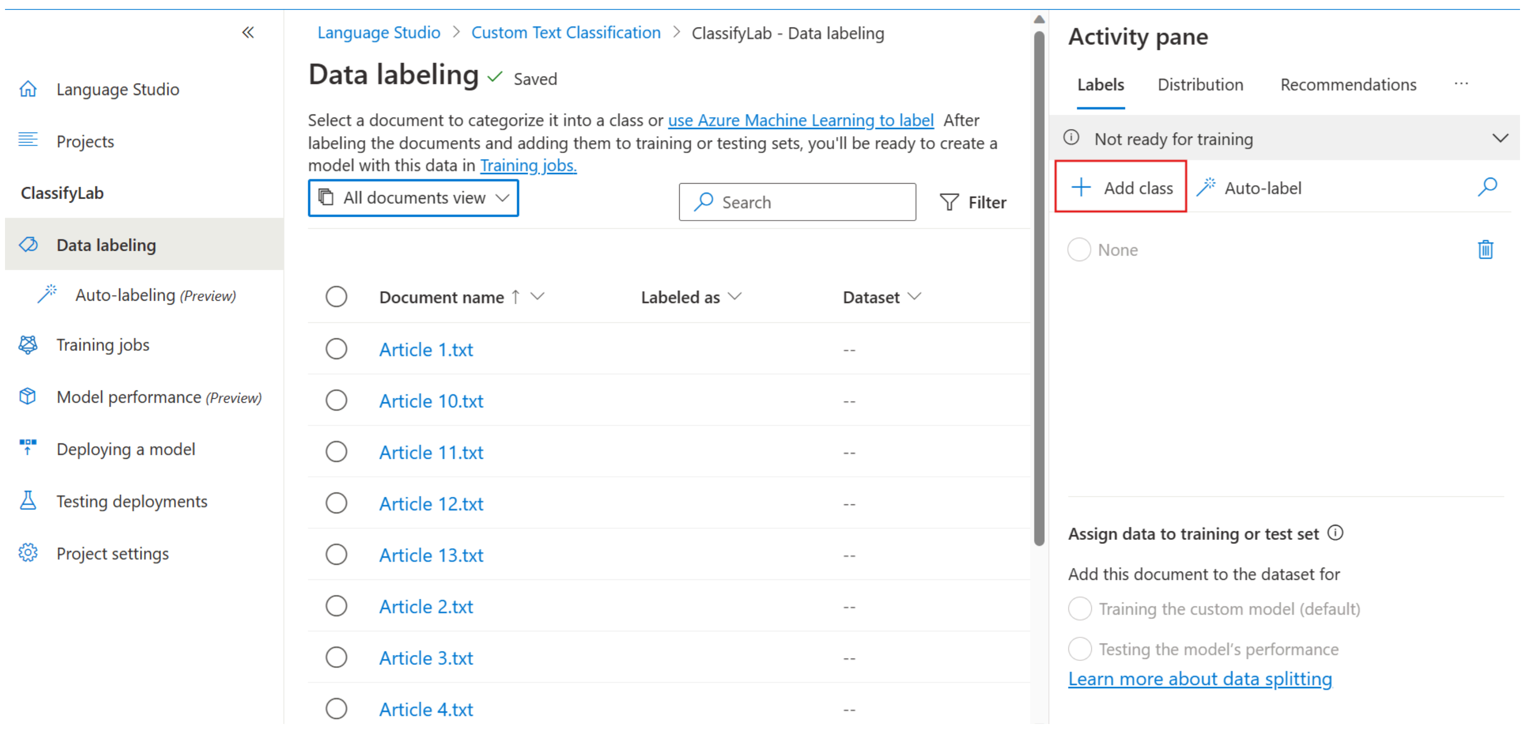
3. At the top of the portal, in the **Create new** menu, select **Custom text classification**.
4. The **Connect storage** page appears. All values will already have been filled. So select **Next**.
5. On the **Select project type** page, select **Single label classification**. Then select **Next**.
6. On the **Enter basic information** pane, set the following:
   - **Name**: `ClassifyLab`
   - **Text primary language**: English (US)
   - **Description**: `Custom text lab`
7. Select **Next**.
8. On the **Choose container** page, set the **Blob store container** dropdown to your *articles* container.
9. Select the **No, I need to label my files as part of this project** option. Then select **Next**.
10. Select **Create project**.

> **Tip**: If you get an error about not being authorized to perform this operation, you'll need to add a role assignment. To fix this, we add the role "Storage Blob Data Contributor" on the storage account for the user running the lab. More details can be found on the documentation page

Label your data

Now that your project is created, you need to label, or tag, your data to train your model how to classify text.

1. On the left, select **Data labeling**, if not already selected. You'll see a list of the files you uploaded to your storage account.
2. On the right side, in the **Activity** pane, select **+ Add class**. The articles in this lab fall into four classes you'll need to create: `Classifieds`, `Sports`, `News`, and `Entertainment`.



Language Studio > Custom Text Classification > ClassifyLab - Data labeling

Data labeling ✓ Saved

Select a document to categorize it into a class or use Azure Machine Learning to label After labeling the documents and adding them to training or testing sets, you'll be ready to create a model with this data in Training jobs.

All documents view | Search | Filter

Document name ↑	Labeled as	Dataset
Article 1.txt		--
Article 10.txt		--
Article 11.txt		--
Article 12.txt		--
Article 13.txt		--
Article 2.txt		--
Article 3.txt		--
Article 4.txt		--

Activity pane

Labels | Distribution | Recommendations

Not ready for training

+ Add class | Auto-label

None

Assign data to training or test set

Add this document to the dataset for

Training the custom model (default)

Testing the model's performance

Learn more about data splitting

3. After you've created your four classes, select **Article 1** to start. Here you can read the article, define which class this file is, and which dataset (training or testing) to assign it to.

4. Assign each article the appropriate class and dataset (training or testing) using the **Activity** pane on the right. You can select a label from the list of labels on the right, and set each article to **training** or **testing** using the options at the bottom of the Activity pane. You select **Next document** to move to the next document. For the purposes of this lab, we'll define which are to be used for training the model and testing the model:

Article	Class	Dataset
Article 1	Sports	Training
Article 10	News	Training
Article 11	Entertainment	Testing
Article 12	News	Testing
Article 13	Sports	Testing
Article 2	Sports	Training
Article 3	Classifieds	Training
Article 4	Classifieds	Training
Article 5	Entertainment	Training
Article 6	Entertainment	Training
Article 7	News	Training
Article 8	News	Training
Article 9	Entertainment	Training

> **NOTE** Files in Language Studio are listed alphabetically, which is why the above list is not in sequential order. Make sure you visit both pages of documents when labeling your articles.

5. Select **Save labels** to save your labels.

Train your model

After you've labeled your data, you need to train your model.

1. Select **Training jobs** on the left side menu.
2. Select **Start a training job**.
3. Train a new model named `ClassifyArticles`.
4. Select **Use a manual split of training and testing data**.

> **TIP** In your own classification projects, the Azure AI Language service will automatically split the testing set by percentage which is useful with a large dataset. With smaller datasets, it's important to train with the right class distribution.

5. Select **Train**

> **IMPORTANT** Training your model can sometimes take several minutes. You'll get a notification when it's complete.

Evaluate your model

In real world applications of text classification, it's important to evaluate and improve your model to verify it's performing as you expect.

1. Select **Model performance**, and select your **ClassifyArticles** model. There you can see the scoring of your model, performance metrics, and when it was trained. If the scoring of your model isn't 100%, it means that one of the documents used for testing didn't evaluate to what it was labeled. These failures can help you understand where to improve.
2. Select **Test set details** tab. If there are any errors, this tab allows you to see the articles you indicated for testing and what the model predicted them as and whether that conflicts with their test label. The tab defaults to show incorrect predictions only. You can toggle the **Show mismatches only** option to see all the articles you indicated for testing and what they each of them predicted as.

Deploy your model

When you're satisfied with the training of your model, it's time to deploy it, which allows you to start classifying text through the API.

1. On the left panel, select **Deploying model**.
2. Select **Add deployment**, then enter `articles` in the **Create a new deployment name** field, and select **ClassifyArticles** in the **Model** field.
3. Select **Deploy** to deploy your model.
4. Once your model is deployed, leave that page open. You'll need your project and deployment name in the next step.

Prepare to develop an app in Cloud Shell

To test the custom text classification capabilities of the Azure AI Language service, you'll develop a simple console application in the Azure Cloud Shell.

1. In the Azure Portal, use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal.

   Note: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

2. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

3. In the PowerShell pane, enter the following commands to clone the GitHub repo for this exercise:

   Code Copy

   ```
   rm -r mslearn-ai-language -f
   git clone https://github.com/microsoftlearning/mslearn-ai-language
   ```

   Tip: As you paste commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

4. After the repo has been cloned, navigate to the folder containing the application code files:

   Code Copy

```
cd mslearn-ai-language/Labfiles/04-text-classification/Python/classify-text
```

Configure your application

1. In the command line pane, run the following command to view the code files in the **classify-text** folder:

   Code Copy
   ```
   ls -a -l
   ```

   The files include a configuration file (**.env**) and a code file (**classify-text.py**). The text your application will analyze is in the **articles** subfolder.

2. Create a Python virtual environment and install the Azure AI Language Text Analytics SDK package and other required packages by running the following command:

   Code Copy
   ```
   python -m venv labenv
   ./labenv/bin/Activate.ps1
   pip install -r requirements.txt azure-ai-textanalytics==5.3.0
   ```

3. Enter the following command to edit the application configuration file:

   Code Copy
   ```
   code .env
   ```

   The file is opened in a code editor.

4. Update the configuration values to include the **endpoint** and a **key** from the Azure Language resource you created (available on the **Keys and Endpoint** page for your Azure AI Language resource in the Azure portal).The file should already contain the project and deployment names for your text classification model.

5. After you've replaced the placeholders, within the code editor, use the **CTRL+S** command or **Right-click > Save** to save your changes and then use the **CTRL+Q** command or **Right-click > Quit** to close the code editor while keeping the cloud shell command line open.

Add code to classify documents

1. Enter the following command to edit the application code file:

   Code Copy
   ```
   code classify-text.py
   ```

2. Review the existing code. You will add code to work with the AI Language Text Analytics SDK.

   > ! **Tip**: As you add code to the code file, be sure to maintain the correct indentation.

3. At the top of the code file, under the existing namespace references, find the comment **Import namespaces** and add the following code to import the namespaces you will need to use the Text Analytics SDK:

   Code Copy

```
# import namespaces
from azure.core.credentials import AzureKeyCredential
from azure.ai.textanalytics import TextAnalyticsClient
```

4. In the **main** function, note that code to load the Azure AI Language service endpoint and key and the project and deployment names from the configuration file has already been provided. Then find the comment **Create client using endpoint and key**, and add the following code to create a text analysis client:

Code Copy

```
# Create client using endpoint and key
credential = AzureKeyCredential(ai_key)
ai_client = TextAnalyticsClient(endpoint=ai_endpoint, credential=credential)
```

5. Note that the existing code reads all of the files in the **articles** folder and creates a list containing their contents. Then find the comment **Get Classifications** and add the following code:

Code Copy

```
# Get Classifications
operation = ai_client.begin_single_label_classify(
   batchedDocuments,
   project_name=project_name,
   deployment_name=deployment_name
)

document_results = operation.result()

for doc, classification_result in zip(files, document_results):
   if classification_result.kind == "CustomDocumentClassification":
       classification = classification_result.classifications[0]
       print("{} was classified as '{}' with confidence score {}.".format(
           doc, classification.category, classification.confidence_score)
       )
   elif classification_result.is_error is True:
       print("{} has an error with code '{}' and message '{}'".format(
           doc, classification_result.error.code, classification_result.error.message)
       )
```

6. Save your changes (CTRL+S), then enter the following command to run the program (you maximize the cloud shell pane and resize the panels to see more text in the command line pane):

Code Copy

```
python classify-text.py
```

7. Observe the output. The application should list a classification and confidence score for each text file.

Clean up

When you don't need your project any more, you can delete if from your **Projects** page in Language Studio. You can also remove the Azure AI Language service and associated storage account in the Azure portal.