


Use Automated Machine Learning

Azure Machine Learning includes an *automated machine learning* capability that leverages the scalability of cloud compute to automatically try multiple pre-processing techniques and model-training algorithms in parallel to find the best performing supervised machine learning model for your data.

In this exercise, you'll use the visual interface for automated machine learning in Azure Machine Learning studio

 **Note:** You can also use automated machine learning through the Azure Machine Learning SDK.

Before you start

If you have not already done so, complete the [Create an Azure Machine Learning Workspace](#) exercise to create an Azure Machine Learning workspace and compute instance, and clone the notebooks required for this exercise.

Configure compute resources

To use automated machine learning, you require compute on which to run the model training experiment.

1. Sign into [Azure Machine Learning studio](#) with the Microsoft credentials associated with your Azure subscription, and select your Azure Machine Learning workspace.
2. In Azure Machine Learning studio, view the **Compute** page; and on the **Compute instances** tab, start your compute instance if it is not already running. You will use this compute instance to test your trained model.
3. While the compute instance is starting, switch to the **Compute clusters** tab, and add a new compute cluster with the following settings. You'll run the automated machine learning experiment on this cluster to take advantage of the ability to distribute the training runs across multiple compute nodes:
 - **Location:** *The same location as your workspace*
 - **Virtual Machine priority:** Dedicated
 - **Virtual Machine type:** CPU
 - **Virtual Machine size:** Standard_DS11_v2
 - **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

Create a dataset

Now that you have some compute resources that you can use to process data, you'll need a way to store and ingest the data to be processed.

1. View the comma-separated data at <https://aka.ms/diabetes-data> in your web browser. Then save this as a local file named **diabetes.csv** (it doesn't matter where you save it).
2. 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.
3. Create a new dataset from local files, using the following settings:
 - **Basic Info:**
 - **Name:** diabetes dataset
 - **Dataset type:** Tabular
 - **Description:** Diabetes data
 - **Datastore and file selection:**
 - **Select or create a datastore:** Currently selected datastore
 - **Select files for your dataset:** Browse to the **diabetes.csv** file you downloaded.
 - **Upload path:** *Leave the default selection*

- **Skip data validation:** Not selected
 - **Settings and preview:**
 - **File format:** Delimited
 - **Delimiter:** Comma
 - **Encoding:** UTF-8
 - **Column headers:** Only first file has headers
 - **Skip rows:** None
 - **Schema:**
 - Include all columns other than **Path**
 - Review the automatically detected types
 - **Confirm details:**
 - Do not profile the dataset after creation
4. After the dataset has been created, open it and view the **Explore** page to see a sample of the data. This data represents details from patients who have been tested for diabetes, and you will use it to train a model that predicts the likelihood of a patient testing positive for diabetes based on clinical measurements.

 **Note:** You can optionally generate a *profile* of the dataset to see more statistical details.

Run an automated machine learning experiment

In Azure Machine Learning, operations that you run are called *experiments*. Follow the steps below to run an experiment that uses automated machine learning to train a classification model that predicts diabetes diagnoses.

1. In Azure Machine Learning studio, view the **Automated ML** page (under **Author**).
2. Create a new Automated ML run with the following settings:
 - **Select dataset:**
 - **Dataset:** diabetes dataset
 - **Configure run:**
 - **New experiment name:** mslearn-automl-diabetes
 - **Target column:** Diabetic (*this is the label the model will be trained to predict*)
 - **Select compute type:** Compute cluster
 - **Select Azure ML compute cluster:** *the compute cluster you created previously*
 - **Task type and settings:**
 - **Task type:** Classification
 - Select **View additional configuration settings** to open **Additional configurations:**
 - **Primary metric:** Select **AUC_Weighted** (*more about this metric later!*)
 - **Explain best model:** Selected - *this option causes automated machine learning to calculate feature importance for the best model; making it possible to determine the influence of each feature on the predicted label.*
 - **Blocked algorithms:** Leave the default setting - *all algorithms can potentially be used when training*
 - **Exit criterion:**
 - **Training job time (hours):** 0.5 - *this causes the experiment to end after a maximum of 30 minutes.*
 - **Metric score threshold:** 0.90 - *this causes the experiment to end if a model achieves a weighted AUC metric of 90% or higher.*
 - Select **View featurization settings** to open **Featurization:**
 - **Enable featurization:** Selected - *this causes Azure Machine Learning to automatically preprocess the features before training.*

- **Select the validation and test type:**
 - **Validation type:** Train-validation split
 - **Percentage validation of data:** 30
 - **Test dataset:** No test dataset required
- 3. When you finish submitting the automated ML run details, it will start automatically. You can observe the status of the run in the **Properties** pane.
- 4. When the run status changes to *Running*, view the **Models** tab and observe as each possible combination of training algorithm and pre-processing steps is tried and the performance of the resulting model is evaluated. The page will automatically refresh periodically, but you can also select ↺ **Refresh**. It may take ten minutes or so before models start to appear, as the cluster nodes need to be initialized and the data featurization process completed before training can begin. Now might be a good time for a coffee break!
- 5. Wait for the experiment to finish.

Review the best model

After the experiment has finished; you can review the best performing model that was generated (note that in this case, we used exit criteria to stop the experiment - so the “best” model found by the experiment may not be the best possible model, just the best one found within the time and metric constraints allowed for this exercise!).

1. On the **Details** tab of the automated machine learning run, note the best model summary.
2. Select the **Algorithm name** for the best model to view the child-run that produced it.

The best model is identified based on the evaluation metric you specified (*AUC_Weighted*). To calculate this metric, the training process used some of the data to train the model, and applied a technique called *cross-validation* to iteratively test the trained model with data it wasn't trained with and compare the predicted value with the actual known value. From these comparisons, a *confusion matrix* of true-positives, false-positives, true-negatives, and false-negatives is tabulated and additional classification metrics calculated - including a Receiving Operator Curve (ROC) chart that compares the True-Positive rate and False-Positive rate. The area under this curve (AUC) is a common metric used to evaluate classification performance.


3. Next to the *AUC_Weighted* value, select **View all other metrics** to see values of other possible evaluation metrics for a classification model.
4. Select the **Metrics** tab and review the performance metrics you can view for the model. These include a **confusion_matrix** visualization showing the confusion matrix for the validated model, and an **accuracy_table** visualization that includes the ROC chart.
5. Select the **Explanations** tab, select an **Explanation ID**, and then view the **Aggregate Importance** page. This shows the extent to which each feature in the dataset influences the label prediction.

Deploy a predictive service

After you've used automated machine learning to train some models, you can deploy the best performing model as a service for client applications to use.

Note: In Azure Machine Learning, you can deploy a service as an Azure Container Instances (ACI) or to an Azure Kubernetes Service (AKS) cluster. For production scenarios, an AKS deployment is recommended, for which you must create an *inference cluster* compute target. In this exercise, you'll use an ACI service, which is a suitable deployment target for testing, and does not require you to create an inference cluster.

1. Select the **Details** tab for the run that produced the best model.
2. From the **Deploy** option, use the **Deploy to web service** button to deploy the model with the following settings:
 - **Name:** auto-predict-diabetes
 - **Description:** Predict diabetes
 - **Compute type:** Azure Container Instance
 - **Enable authentication:** Selected
 - **Use custom deployment assets:** Unselected

3. Wait for the deployment to start - this may take a few seconds. Then, on the **Model** tab, in the **Model summary** section, observe the **Deploy status** for the **auto-predict-diabetes** service, which should be **Running**. Wait for this status to change to **Successful**. You may need to select ↻ **Refresh** periodically.
NOTE This can take a while - be patient!
4. In Azure Machine Learning studio, view the **Endpoints** page and select the **auto-predict-diabetes** real-time endpoint. Then select the **Consume** tab and note the following information there. You need this information to connect to your deployed service from a client application.
 - The REST endpoint for your service
 - the Primary Key for your service
5. Note that you can use the  link next to these values to copy them to the clipboard.

Test the deployed service


Now that you've deployed a service, you can test it using some simple code.

1. With the **Consume** page for the **auto-predict-diabetes** 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.
2. In the **Notebooks** page, under **My files**, browse to the **/users/your-user-name/mslearn-dp100** folder where you cloned the notebook repository, and open the **Get AutoML Prediction** notebook.
3. When the notebook has opened, ensure that the compute instance you created previously is selected in the **Compute** box, and that it has a status of **Running**.
4. In the notebook, replace the **ENDPOINT** and **PRIMARY_KEY** placeholders with the values for your service, which you can copy from the **Consume** tab on the page for your endpoint.
5. Run the code cell and view the output returned by your web service.

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 **auto-predict-diabetes** 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**.

 **Note:** Stopping your compute ensures your subscription won't be charged for compute resources. You will however be charged a small amount for data storage as long as the Azure Machine Learning workspace exists in your subscription. If you have finished exploring Azure Machine Learning, you can delete the Azure Machine Learning workspace and associated resources. However, if you plan to complete any other labs in this series, you will need to repeat the [Create an Azure Machine Learning Workspace](#) exercise to create the workspace and prepare the environment first.