**Data Input and Output in Microsoft Azure Microsoft Machine Learning**

**Data Input and Output**

This lists the modules you can import and export data and models in Machine Learning Studio (classic).

In addition to using modules, you can directly upload and download datasets from local files on your computer or network. For more information, see Upload existing data into a Machine Learning experiment.

Here are some of the sources you can use to import and export data and models in Machine Learning Studio (classic):

* Get data from sources in the cloud, such as Azure SQL Database, Azure SQL Data Warehouse, Azure Storage, and Azure Cosmos DB. You can also import data provided as a public web URL, get data from Hadoop by using a Hive query, or query an on-premises SQL server.
* Load a collection of images from Azure Blob storage to use in image classification tasks.
* Extract the data from zip files that you uploaded to Machine Learning. You can use the datasets in experiments.
* Create small datasets by typing in the Machine Learning Studio (classic) UI. This can be handy for creating small test datasets.
* Save your results or intermediate data to Azure Table storage, Blob storage, an SQL database, or a Hive query.
* Get a trained model from a URL or Blob storage, and then use it in an experiment.

**Enter Data Manually**

This describes how to use the Enter Data Manually module in Machine Learning Studio (classic), to create a small dataset by typing values. The dataset can have multiple columns.

This module can be helpful in scenarios such as these:

* Generating a small set of values for testing
* Creating a short list of labels
* Entering values for use in Apply Math Operation
* Specifying replacement values for use in Replace Discrete Values
* Typing a list of column names to insert in a dataset

**How to use Enter Data Manually**

* Add the Enter Data Manually module to your experiment. You can find this module in the Data Input and Output category in Machine Learning Studio (classic).
* For DataFormat, select one of the following options. These options determine how the data that you provide should be parsed. The requirements for each format differ greatly, so be sure to read the related topics.

1. ARFF. The attribute-relation file format, used by Weka. For more information, see Convert to ARFF.
2. CSV. Comma-separated values format. For more information, see Convert to CSV.
3. SVMLight. A format used by Vowpal Wabbit and other machine learning frameworks. For more information, see Convert to SVMLight.
4. TSV. Tab-separated values format. For more information, see Convert to TSV.

If you choose a format and do not provide data that meets the format specifications, a run-time error occurs.

* Click inside the Data text box to start entering data. The following formats require special attention:

1. CSV:

To create multiple columns, paste in comma-separated text, or type multiple columns using commas between fields.

If you select the HasHeader option, you can use the first row of values as the column heading.

If you deselect this option, the column names, Col1, Col2, and so forth are used. You can add or change column names later using Edit Metadata.

1. TSV:

To create multiple columns, paste in tab-separated text, or type multiple columns using tabs between fields.

If you select the HasHeader option, you can use the first row of values as the column heading.

If you deselect this option, the column names, Col1, Col2 and so forth are used. You can add or change column names later using Edit Metadata.

1. ARFF:

Paste in an existing ARFF format file. If you are typing values directly, be sure to add the optional header and required attribute fields at the beginning of the data.

For example, the following header and attribute rows could be added to a simple list. The column heading would be SampleText.

Code:

*% Title: SampleText.ARFF*

*% Source: Enter Data module*

*@ATTRIBUTE SampleText STRING*

*@DATA*

*\<type first data row here>*

SVMLight: Type or paste in values using the SVMLight format

For example, the following sample represents the first couple of lines of the Blood Donation dataset, in SVMight format

Code:

*# features are [Recency], [Frequency], [Monetary], [Time]*

*1 1:2 2:50 3:12500 4:98*

*1 1:0 2:13 3:3250 4:28*

When you run the Enter Data Manually module, these lines are converted to a dataset of columns and index values as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Col1 | Col2 | Col3 | Col4 | Labels |
| 0.00016 | 0.004 | 0.999961 | 0.00784 | 1 |
| 0 | 0.004 | 0.999955 | 0.008615 | 1 |

* Press ENTER after each row, to start a new line.

Be sure to press ENTER after the final row.

If you press ENTER multiple times to add multiple empty trailing rows, the final empty row is removed trimmed, but other empty rows are treated as missing values.

If you create rows with missing values, you can always filter them out later.

* Right-click the module and select Run to parse the data and load it into your workspace as a dataset.

To view the dataset, click the output port and select Visualize.

**Export Data**

This describes how to use the Export Data module in Machine Learning Studio (classic), to save results, intermediate data, and working data from your experiments into cloud storage destinations outside Machine Learning Studio (classic).

This module supports exporting or saving your data to the following cloud data services:

* Export to Hive Query: Write data to a Hive table in an HDInsight Hadoop cluster.
* Export to Azure SQL Database: Save data to Azure SQL Database or to Azure SQL Data Warehouse.
* Export to Azure Table: Save data to the table storage service in Azure. Table storage is good for storing large amounts of data. It provides a tabular format that is scalable, inexpensive, and highly available.
* Export to Azure Blob Storage: Saves data to the Blob service in Azure. This option is useful for images, unstructured text, or binary data. Data in the Blob service can be shared publicly or saved in secured application data stores.

**How to configure Export Data**

* Add the Export Data module to your experiment in Studio (classic). You can find this module in the Input and Output category.
* Connect Export Data to the module that contain the data you want to export.
* Double-click Export Data to open the Properties pane.
* For Data destination, select the type of cloud storage where you'll save your data. If you make any changes to this option, all other properties are reset. So be sure to choose this option first!
* Provide an account name and authentication method required to access the specified storage account.

Depending on the storage type and whether the account is secured, you might need to provide the account name, file type, access key, or container name. For sources that do not require authentication, generally it is sufficient to know the URL.

For examples of each type, see the following topics:

1. Export to Hive Query
2. Export to Azure SQL Database
3. Export to Azure Table
4. Export to Azure Blob Storage

* The option, Use cached results, lets you repeat the experiment without rewriting the same results each time.If you deselect this option, results are written to storage each time the experiment is run, regardless of whether the output data has changed.If you select this option, Export Data uses cached data, if available. New results are generated only when there is an upstream change that would affect the results.
* Run the experiment.

**Export to Hive Query**

This article describes how to use the Export data to Hive option in the Export Data module in Machine Learning Studio (classic). This option is useful when you are working with very large datasets, and want to save your machine learning experiment data to a Hadoop cluster or HDInsight distributed storage. You might also want to export intermediate results or other data to Hadoop so that you can process it using a MapReduce job.

**How to export data to Hive**

* Add the Export Data module to your experiment. You can find this module in the Data Input and Output category in Machine Learning Studio (classic).Connect the module to the dataset you want to export.
* For Data source, select Hive Query.
* For Hive table name type the name of the Hive table in which to store the dataset.
* In the HCatalog server URI text box, type the fully qualified name of your cluster.
* In the Hadoop user account name text box, paste in the Hadoop user account that you used when you provisioned the cluster.
* In the Hadoop user account password text box, type the credentials that you used when you provisioned the cluster.
* For Location of output data, select the option that indicates where the data should be stored: HDFS, or Azure.

If the data is in the Hadoop distributed file system (HDFS), it must be accessible via the same account and password that you just entered.

If the data is in Azure, provide the location and credentials of the storage account.

* If you selected the HDFS option, for HDFS server URI, specify the HDInsight cluster name without the https:// prefix.
* If you selected the Azure option, provide the storage account name, and the credentials the module can use to connect to storage.

1. Azure storage account name: Type the name of the Azure account.

Forexample, if the full URL of the storage account is https://myshared.blob.core.windows.net, you would type myshared.

1. Azure storage key: Copy and paste the key that is provided for accessing the storage account.
2. Azure container name: Specify the default container for the cluster. For tips son how to figure out the default container, see the Technical notes section.

* Use cached results: Select this option if you want to avoid rewriting the Hive table each time you run the experiment. If there are no other changes to module parameters, the experiment writes the Hive table only the first time the module is run, or when there are changes to the data.
* If you want to write the Hive table each time the experiment is run, deselect the Use cached results option.
* Run the experiment.

Export to Azure SQL Database

This describes how to use the Export to Azure SQL Database option in the Export Data module in Machine Learning Studio (classic). This option is useful when you want to export data from your machine learning experiment to an Azure SQL Database or Azure SQL Data Warehouse.

Export to a SQL database is useful in many machine learning scenarios: for example, you might want to store intermediate results, save scores, or persist tables of engineered features. Although storing data in an Azure SQL Database or Azure SQL Data Warehouse can be more expensive than using tables or blobs in Azure, there are no transaction fees against SQL databases. Moreover, database storage is ideal for quickly writing smaller amounts of frequently used information, for sharing data between experiments, or for reporting results, predictions, and metrics.On the other hand, there might be limits on the amount of data that you can store in a database, depending on your subscription type. You should also consider using a database and account that is in the same region as your machine learning workspace.

To export data, you provide the instance name and database name where the data is stored, and run the module using an account that has write permissions. You must also specify the table name, and map the columns from your experiment to columns in the table.

**How to export data to an Azure SQL Database**

* Add the Export Data module to your experiment in Studio (classic). You can find this module in the Data Input and Output category.
* Connect Export data to the module that produces the data that you want to export.
* For Data destination, select Azure SQL Database. This option supports Azure SQL Data Warehouse as well.
* Indicate the name of the server and database in Azure SQL Database or Azure SQL Data Warehouse.

1. Database server name: Type the server name as generated by Azure. Typically it has the form <generated\_identifier>.database.windows.net.
2. Database name: Type the name of an existing database on the server you just specified. The Export Data module cannot create a database.
3. Server user account name: Type the user name for an account that has access permissions for the database.
4. Server user account password: Provide the password for the specified user account.

* Specify the columns to export, and if you want to rename the columns.

1. Comma-separated list of columns to be saved: Type the names of the columns from the experiment that you want to write to the database.
2. Data table name: Type the name of the table to store the data in.

For Azure SQL Database, if the table does not exist, a new table is created.

For Azure SQL Data Warehouse, the table must already exist and have the correct schema, so be sure to create it in advance.

1. Comma-separated list of datatable columns: Type the names of the columns as you wish them to appear in the destination table.

For Azure SQL Database, you can change the column names, but you must keep the columns in the same order that you listed the columns for export, in Comma-separated list of columns to be saved.

For Azure SQL Data Warehouse, the columns names must match those already in the destination table schema.

* Number of rows written per SQL Azure operation: This option specifies how many rows should be written to the destination table in each batch.

By default, the value is set to 50, which is the default batch size for Azure SQL Database. However, you should increase this value if you have a large number of rows to write.

For Azure SQL Data Warehouse, we recommend that you set this value to 1.If you use a larger batch size, the size of the command string that is sent to Azure SQL Data Warehouse can exceed the allowed string length, causing an error.

* Use cached results: Select this option to avoid writing new results each time the experiment is run. If there are no other changes to module parameters, the experiment writes the data only the first time the module is run. However, a new write is always performed if any parameters have been changed in Export Data that would change the results.
* Run the experiment.

**Export to Azure Table**

**How to export data to an Azure table**

* Add the Export Data module to your experiment. You can find this module in the Data Input and Output category in Studio (classic).
* Connect it to the module that produces the data that you want to export to Azure table storage.
* Specify whether you want to export data to a public shared resource or to a private storage account that requires login credentials, by setting the Authentication type option.

1. Public (SAS URL): Choose this option if the account supports access via SAS URL. In the Table SAS URI field, type or paste the full URI that defines the account and the public blob.
2. The SAS URL is a time bound access URL that you can generate by using an Azure storage utility. In a page accessible via SAS URL, data can be stored using only these formats: CSV, TSV, and ARFF.
3. Account: Choose this option if your data is in a private account. You must also supply credentials including the account name and the key.

* If you want to export your data to secured, private storage, provide the credentials needed for accessing the account:

1. Table account name: Type or paste the name of the account that contains the blob you want to access. For example, if the full URL of the storage account is https://myshared.table.core.windows.net, you would type myshared.
2. Table account key: Paste the access key that is associated with the storage account.
3. Table name: Type the name of the specific table that you want to read.

* Specify which columns to save to the table store, and which columns to use in defining the table schema, by using the column properties.

1. Partition key: Choose the column that should be used for partitioning the saved dataset for the table in Azure Storage. Tables in Azure are partitioned to support load balancing across storage nodes. All table entities are organized by partition; therefore, the PartitionKey property is required for all table operations.
2. Azure table row key: Choose the column that should be used for the RowKey property. The RowKey property is a system property that is required for every entity in a table. Along with the PartitionKey property, it forms a unique index for every row in the table.
3. Azure table origin columns: Select any additional columns from the dataset that you want to save to the Azure table. You must also include the columns selected for PartitionKey and RowKey.

* Azure table write mode: Indicate how you want the Export Data to behave when data already exists in the Azure table.

1. Insert: The Insert Entity operation inserts a new entity with a unique primary key, which is formed from a combination of the PartitionKey and the RowKey properties.
2. Merge: The Merge Entity operation updates an existing entity by updating the entity's properties. This operation does not replace the existing entity.
3. Replace: The Update Entity operation replaces the contents of the given entity in a table.
4. InsertOrReplace: The InsertOrReplace Entity operation inserts the entity if the entity does not exist. If the entity exists, it replaces the existing one.
5. InsertOrMerge: The InsertOrMerge Entity operation inserts the entity if the entity does not exist. If the entity exists, it merges the provided entity properties with the already existing ones.

* Use cached results: Indicate whether you want the data to be refreshed each time the experiment is run.

If you select this option, the Export Data module saves data to the specified table the first time the experiment is run, and thereafter not perform writes, unless there are upstream changes.

If you deselect this option, the data is written to the destination each time the experiment is run, regardless of whether the data is the same or not.

* Run the experiment.

**Import Data**

Loads data from external sources on the web; and from various forms of cloud-based storage in Azure such as tables, blobs, and SQL data. This describes how to use the Import Data module in Machine Learning Studio (classic), to load data into a machine learning experiment from existing cloud data services.The module now features a wizard to help you choose a storage option and select from among existing subscriptions and accounts to quickly configure all options.

After you define the data you want and connect to the source, Import Data infers the data type of each column based on the values it contains, and loads the data into your Machine Learning Studio (classic) workspace. The output of Import Data is a dataset that can be used with any experiment.If your source data changes, you can refresh the dataset and add new data by re-running Import Data. However, if you don't want to re-read from the source each time you run the experiment, select the Use cached results option to TRUE. When this option is selected, the module checks whether the experiment has run previously using the same source and same input options. If a previous run is found, the data in the cache is used, instead of re-loading the data from the source.

**Data sources**

The Import Data module supports the following data sources. Click the links for detailed instructions and examples of using each data source.

If you are not sure how or where you should store your data, see this guide to common data scenarios in the data science process: Scenarios for advanced analytics in Machine Learning.bases; and from on-premises SQL Server databases

**How to use Import Data**

* Add the Import Data module to your experiment. You can find this module in the Data Input and Output category in Studio (classic).
* Click Launch Data Import Wizard to configure the data source using a wizard.

The wizard gets the account name and credentials, and help you configure other options. If you are editing an existing configuration, it loads the current values first.

* If you do not want to use the wizard, click Data source, and choose the type of cloud-based storage you are reading from.

Additional settings depend on the type of storage you choose, and whether the storage is secured or not. You might need to provide the account name, file type, or credentials. Some sources do not require authentication; for others, you might need to know the account name, a key, or container name.

* Select the Use cached results option if you want to cache the dataset for re-use on successive runs. Assuming there have been no other changes to module parameters, the experiment loads the data only the first time the module is run, and thereafter uses a cached version of the dataset.Deselect this option if you need to reload the data each time you run the experiment.
* Run the experiment.

When Import Data loads the data into Studio (classic), it infers the data type of each column based on the values it contains, either numerical or categorical.

1. If a header is present, the header is used to name the columns of the output dataset.
2. If there are no existing column headers in the data, new column names are generated using the format col1, col2,… ,coln.

**Results**

When import completes, click the output dataset and select Visualize to see if the data was imported successfully.

If you want to save the data for re-use, rather than importing a new set of data each time the experiment is run, right-click the output and select Save as Dataset. Choose a name for the dataset. The saved dataset preserves the data at the time of saving, and data is not updated when the experiment is re-run, even if the dataset in the experiment changes. This can be handy for taking snapshots of data.

After importing the data, it might need some additional preparations for modeling and analysis:

* Generate statistical summaries of the data, using Summarize Data or Compute Elementary Statistics.
* Use Edit Metadata to change column names, to handle a column as a different data type, or to indicate that some columns are labels or features.
* Use Select Columns in Dataset to select a subset of columns to transform or use in modeling. The transformed or removed columns can easily be rejoined to the original dataset by using the Add Columns module or the Join Data module.
* Use Partition and Sample to divide the dataset, perform sampling, or get the top n rows.
* Use Apply SQL Transformation to aggregate data, filter, or transform using SQL statements.
* Use these modules to clean up text columns and generate new text features:

1. Preprocess Text
2. Extract N-Gram Features from Text
3. Named Entity Recognition
4. Execute Python Script, to implement custom NLP based on nltk.

**Load Trained Model**

This describes how to use the Load Trained Model module in Machine Learning Studio (classic), to load an already trained model for use in an experiment.This module requires an existing trained model. Typically, you create and then train the model in a different experiment, and then save the model either to your workspace, or to one of the supported cloud storage options.Then, you use the Load Trained model module to get the trained model and run it in a new experiment.

**How to use Load Trained Model**

To use an existing model to make predictions for new data.

* The model must have previously been trained and then saved in the iLearner format.
* The model must be accessible either by URL or in Azure blob storage.

**Save a trained model**

You can save models by using the Studio (classic) interface, or using an experiment that runs as a web service.

**Save a model using a web service**

* Create an experiment that does the training or retraining of the model as a web service
* Publish that experiment as a Web service.
* When you call the BES endpoint of the training web service, the Web service saves a trained model using the iLearner interface and saves the file in the Azure blob storage account that you specify.

**Save a model in Studio (classic)**

* Run the experiment that builds and trains the model.
* When training is complete, right-click the module that was used for training, select Trained model, and then click Save as trained model.
* By default, models are saved to your Studio (classic) workspace. You can view them using the Studio (classic) UI.

The following modules can create a saved model that uses the required iLearner interface:

* Train Model
* Train Clustering Model
* Train Anomaly Detection Model
* Tune Model Hyperparameters
* Sweep Clustering

**Load the model into a new experiment**

* Add the Load Trained Model module to your experiment in Studio (classic).
* For Data source, indicate the location of the trained model, using one of the following options:

1. Web URL via HTTP: Provide a URL that points to the experiment and the file representing the trained model. In Machine Learning, trained models are by default saved in the ILearner format.
2. Azure Blob Storage: Select this option only if you exported the trained model to Azure storage. You must then provide the account name and account key, and the path to the container, directory, or blob.

* If you intend to create a Request-Response web service that is based on the current experiment, select the option, Allow to use in RRS. Otherwise, scoring is performed using the Batch Execution Service (BES) option, which is recommended. See the Technical notes section for details.
* Select the Use cached results option if you want to load the trained model from cache, when the cache is available and populated. This option is ignored after the experiment is deployed as a Web service API.

**Unpack Zipped Datasets**

This describes how to use the Unpack Zipped Datasets module in Machine Learning Studio (classic), to upload data and script files in compressed format, and then unzip them for use in an experiment.The purpose of this module is to reduce data transfer times when working with very large datasets by saving and uploading your data files in a compressed format. Generally, zipping files is a good option when your dataset is so large that you want to use compression for the upload, to minimize upload time and associated costs.The module takes as input a dataset in your workspace. The dataset must have been uploaded in a compressed format. The module then decompresses the dataset and adds the data to your workspace.

**How to use Unpack Zipped Datasets**

**Step 1. Prepare files**

Before uploading your file, make sure that the data in the file can be used in Machine Learning:

* Ensure that the data in the file uses UTF-8 encoding.

If the file is small enough, you can open it in Notepad and then save the file in the desired encoding. Many other text editors offer similar functionality. For CSV files, you can use Excel's Save As or Export commands to specify a file format and encoding.

* Verify that the data files use a supported format, such as CSV, TSV, ARFF, or SVMLight.
* Compress the data by adding the data file to a .ZIP or .GZ format archive file. Other archive types are not supported.
* Remove password protection. If any of the files or the compressed folder itself has been encrypted or password-protected, you must unlock or decrypt the file before you upload it. The module cannot detect encrypted data types and does not support dialog boxes for password entry from arbitrary clients.

**Step 2. Upload dataset to your workspace**

Next, upload the zipped dataset to your experiment workspace.

* Click NEW, select DATASET, and select FROM LOCAL FILE.
* Locate the zipped file to upload. When you select the file, the type should automatically be set to Zip file (.zip).

**Step 3. Add zipped dataset to experiment**

After the dataset has uploaded completely, add it to your experiment in zipped format.

* In the left-hand navigation pane of Machine Learning Studio (classic), select Saved Datasets, and then expand My Datasets.
* Locate the zipped dataset that you just uploaded, and drag it to the experiment canvas.

**Step 4. Unpack dataset**

The final step is to unpack the dataset.

* Connect the zipped dataset to the input of the Unpack Zipped Datasets module.
* In Dataset to Unpack, type the name of a single dataset to unpack.

1. If you saved a worksheet with the name Sheet1 as an Excel CSV file named Test.csv, the name of the dataset would be Test.csv, not Sheet1.
2. The name that you type in the Dataset to Unpack text box must be exactly the same as the name of the original file before it was compressed, including the file name extension. For example, if you want to unpack a dataset based on the text file Users.txt, type Users.txt, not Users.
3. If you put multiple files into one compressed folder, you must unpack one dataset at a time.

* For Dataset file format, specify the original format of the dataset: that is, the format before it was zipped.

1. You can upload and unzip datasets that were created using any of these formats: CSV, ARFF, TSV, SvmLight.
2. If this property is left empty, the module identifies the dataset using the source file name.

* Select the option, File has a header row, if the original dataset had a header row. Otherwise, the first row of data is used as the header. If this is not what you want, add a header prior to input. This option applies only to .CSV and .TSV files.
* If the file is compressed, use the Compression file format option to specify the algorithm that was used to compress or expand the file.Currently the .ZIP and GZ (or Gzip) formats are supported.
* Run the experiment

**Results**

* To verify that the data was imported correctly, right-click the Unpacked Zipped Datasets module, and select Visualize .
* To change the name of the dataset, right-click the Unpacked Zipped Datasets module, and select Save as Dataset. At this point you can type a different name.

This option is handy if you are unpacking multiple datasets from a single ZIP file.