**Data Format in Machine Learning Studio**

**Data Format Conversions**

This lists the modules provided in Machine Learning Studio (classic) for converting data among various file formats used in machine learning.

*The supported formats include:*

* The dataset format that's used throughout Machine Learning.
* The ARFF format that's used by Weka. Weka is an open-source Java-based set of machine learning algorithms.
* The SVMLight format. The SVMLight format was developed for the SVMlight framework for machine learning. It can also be used by Vowpal Wabbit.
* The tab-separated (TSV) and comma-separated (CSV) flat file formats that are supported by most relational databases. These formats are also widely supported by R and Python.
* When you convert data to these formats, you can more easily move results and data between different machine learning frameworks or storage mechanisms.

**Common data conversion scenarios**

You typically use the data conversion modules if you need to move data from a Machine Learning experiment to another machine learning tool or platform. You also can use the modules to export data from Machine Learning in a format that can be used by a database or other tools. For example:

|  |  |
| --- | --- |
| **Task** | **Use this** |
| * You need to save an intermediate dataset to use in Excel, or to import to a database. | * Use the CSV module or the TSV module to prepare the data in the correct format. Then, either download the data or save it to Azure Storage. |
| * You want to reuse data from your experiment in R or Python code. | * Use the CSV module or the TSV module to prepare the data. Then, right-click the converted dataset to get the Python code that you need to access the dataset. |
| * You are porting your experiment and data between Weka and Machine Learning. | * Use the ARFF module to prepare the data. Then, download the results. |
| * You need to prepare data in the SVMlight framework. | * Use the Convert to SVMLight module to prepare the data. Then, download the resulting data. |
| * Create data to use with Vowpal Wabbit. | * Use the SVMLight format. Then, modify the files as described in the article. Save the file in Azure Blob storage to use with a Vowpal Wabbit module in Machine Learning. |
| * Data is not in a tabular format. | * Coerce it to a dataset format by using the Convert to Dataset module. |
| * Import data from my computer into Machine Learning. | * Upload datasets in CSV format as described in Import your training data into Machine Learning Studio (classic). |
| * Import data from a cloud data source, including Hadoop or Azure. | * Use the Import Data module. |
| * Save machine learning datasets to Azure Blob storage, a Hadoop cluster, or other cloud-based storage. | * Use the Export Data module. |
| * Change the data type of columns or cast columns to a different format or type. | * In Machine Learning, use the Edit Metadata or Apply SQL Transformation modules. If you are proficient with R or Python, try the Execute Python Script or Execute R Script modules. |
| * Round, group, or normalize numerical data. | * Use the Apply Math Operation, Group Data into Bins, or Normalize Data modules. |

**List of modules**

The Data Format Conversions category includes these modules:

* *Convert to ARFF:* Converts data input to the attribute relation file format that's used by the Weka toolset.
* *Convert to CSV:* Converts a dataset to a comma-separated values format.
* *Convert to Dataset:* Converts data input to the internal dataset format that's used by Machine Learning.
* *Convert to SVMLight*: Converts data input to the format that's used by the SVMlight framework.
* Convert to TSV: Converts data input to the tab-delimited format.

**Convert to ARFF**

This describes how to use the Convert to ARFF module in Machine Learning Studio (classic), to convert datasets and results from the attribute-relation file format used by the Weka toolset. This format is known as ARFF.The ARFF data specification for Weka supports multiple machine-learning tasks, including data preprocessing, classification, and feature selection. In this format, data is organized by entities and their attributes and is contained in a single text file.

**How to use Convert to ARFF**

* Add the Convert to ARFF module to your experiment. You can find this module in the Data Format Conversions category in Machine Learning Studio (classic).
* Connect it to any module that outputs a dataset.
* Run the experiment, or click the Convert to ARFF module, and click Run selected.

**Results**

To create a copy of the data in a local folder, double-click the output of Convert to ARFF, and select the Download option.If you do not specify a folder, a default file name is applied and the file is saved in the local Downloads library.

**Example of ARFF format**

This section provides an example of how a typical dataset would look when converted to ARFF.

Typically an ARFF data file is comprised of two sections: a header that defines the data source and schema, and the data section, which contains the actual entities and their attributes.

**ARFF header**

* The header for an ARFF file defines the list of the attributes (in columns) and their data types. The header can also contain multiple comment lines that describe the data source or any other notes.

*% Source: Iris dataset, UCI % 0 = Iris-setosa, 1= Iris-virginica @RELATION iris @ATTRIBUTE sepal\_length NUMERIC @ATTRIBUTE sepal\_width NUMERIC @ATTRIBUTE petal\_length NUMERIC @ATTRIBUTE petal\_width NUMERIC @ATTRIBUTE class {0, 1} ARFF data*

* The data section consists of comma-separated values and looks very much like a CSV file without column headings.

*@DATA 5.1,3.5,1.4,0.2,0*

**Current ARFF version**

Machine Learning Studio (classic) saves ARFF files by using the ARFF 3.0 format.

**Convert to CSV**

This describes how to use the Convert to CSV module in Machine Learning Studio (classic), to convert a dataset from Azure ML into a CSV format that can be downloaded, exported, or shared with R or Python script modules. The CSV format, which stands for "comma-separated values", is a file format used by many external machine learning tools. Although the native dataset format used by Machine Learning is based on the .NET data table and thus can be read by .NET libraries, CSV is a common interchange format when working with open-source languages such as R or Python. Even if you do most of your work in Machine Learning Studio (classic), there are times when you might find it handy to convert your dataset to CSV to use in external tools.

For example:

* Download the CSV file to open it with Excel, or import it into a relational database.
* Save the CSV file to cloud storage and connect to it from Power BI to create visualizations.
* Use the CSV format to prepare data for use in R and Python. Just right-click the output of the module to generate the code needed to access the data directly from Python or a Jupyter notebook.

When you convert a dataset to CSV, the file is saved in your Azure ML workspace. You can use an Azure storage utility to open and use the file directly, or you can right-click the module output and download the CSV file to your computer, or use it in R or Python code.

**How to configure Convert to CSV**

* Add the Convert to CSV module to your experiment. You can find this module in the Data Format Conversions group in Studio (classic).
* Connect it to any module that outputs a dataset.
* Run the experiment, or click the Convert to CSV module, and click Run selected.

**Results**

Double-click the output of Convert to CSV, and select one of these options.

* ***Download:***
  1. Immediately opens a copy of the data in CSV format that you can save to a local folder. If you do not specify a folder, a default file name is applied and the CSV file is saved in the local Downloads library.
  2. If you select Download dataset, you must indicate whether you want to open the dataset, or save it to a local file.
  3. If you select Open, the dataset is loaded using the application that is associated by default with .CSV files: for example, Microsoft Excel.
  4. If you select Download dataset, by default, the file is saved with the name of the module plus a GUID representing the workspace ID. However, you can select the Save As option during download and change the file name or location.
* ***Save as Dataset:*** Saves the CSV file back to the Azure ML workspace as a separate dataset.
* ***Generate Data Access Code:*** Azure ML generates two sets of code for you to access the data, either by using Python or by using R. To access the data, copy the code snippet into your application.
* ***Open in a new Notebook:*** A new Jupyter notebook is created for you and code inserted for reading the data from your workspace, using the language of your choice: Python 2, Python 3, or R with Microsoft R Open.

**Common questions and issues**

This describes some known issues, common questions, and workarounds specific to the Convert to CSV module.

* *Headers must be single rows*

The CSV file format used in Machine Learning supports a single header row. You cannot insert multi-line headers.

* *Custom separators supported on import but not export*

The Convert to CSV module does not support generating alternative column separators, such as the semicolon (;), which is often used in Europe.

However, when you import data from CSV files in external storage, you can specify alternative separators. In the Import Data module, select the CSV with encodings option, and pick a supported encoding.

* *Inaccurate column separation on string data containing commas*

It is a common problem in text processing that just about every character that can be specified as a column separator (tabs, spaces, commas, etc.) can also be found randomly in text fields. Importing text from CSV always requires caution to avoid separating text across unnecessary new columns.

When you try to export a column of string data that contains commas, you might run into problems as well. Machine Learning does not support any special handling or special translation of such data, such as enclosing strings in quotation marks. Also, you cannot use escape characters before a comma to ensure that commas are handled as a literal character.

Therefore, new fields are created in the output file for each comma that is encountered in the string field. To avoid this problem, there are several workarounds:

1. Use the Preprocess Text module to remove punctuation characters from string fields.
2. Use custom R script or Python script to process text and ensure that data can be exported correctly.

* *UTF-8 encoding required :*The Convert to CSV module supports only UTF-8 character encoding. If you need to export data using a different encoding, you can try using the Execute R Script or Execute Python Script modules to generate custom output.
* *Dataset does not have column names*

If the dataset you are exporting to a CSV file does not have column names, we recommend that you use Edit Metadata to add column names before converting it. You cannot add column names as part of the conversion or export process.

* *SYLK: File format is not valid*

If the first column of the dataset that you convert to CSV has the name ID, you might get the following error when you try to open the file in Excel:

"SYLK: File format is not valid."To avoid this error, you must rename the column.

**Convert to Dataset**

This describes how to use the Convert to Dataset module in Machine Learning Studio (classic), to convert any data that you might need for an experiment to the internal format used by Studio (classic). Conversion is not required in most cases, because Machine Learning implicitly converts data to its native dataset format when any operation is performed on the data. However, saving data to the dataset format is recommended if you have performed some kind of normalization or cleaning on a set of data, and you want to ensure that the changes are used in further experiments.

**How to use Convert to Dataset**

We recommend that you use the Edit Metadata module to prepare the dataset before using Convert to Dataset. You can add or change column names, adjust data types, and so forth.

* Add the Convert to Dataset module to your experiment. You can find this module in the Data Format Conversions category in Machine Learning Studio (classic).
* Connect it to any module that outputs a dataset.

As long as the data is tabular, you can convert it to a dataset. This includes data loaded using Import Data, data created by using Enter Data Manually, data generated by code in custom modules, datasets transformed by using Apply Transformation, or datasets that were generated or modified by using Apply SQL Transformation.

* In the Action dropdown list, indicate if you want to do any cleanup on the data before saving the dataset:

1. None: Use the data as is.
2. SetMissingValue: Specify a placeholder that is inserted in the dataset wherever there is a missing value. The default placeholder is the question mark character (?), but you can use the Custom Missing Value option to type a different value.
3. ReplaceValues: Use this option to specify a single exact value to be replaced with any other exact value. For example, assuming your data contains the string obs used as a placeholder for missing values, you could specify a custom replacement operation using these options:
   * 1. Set Replace to Custom
     2. For Custom value, type the value you want to find. In this case, you would type obs.
     3. For the New value, type the new value to replace the original string with. In this case, you might type ?

Note that the ReplaceValues operation applies only to exact matches. For example, these strings would not be affected: obs., obsolete.

* SparseOutput: Indicates that the dataset is sparse. By creating a sparse data vector, you can ensure that missing values do not affect a sparse data distribution. After choosing this option, you must indicate how missing values and zero values should be handled.

To remove any value other than zero, click the Remove option and type a single value to remove. You can remove missing values, or set a custom value to delete from the vector. Only exact matches will be removed. For example, if you type x in the Remove value text box, the row xx would not be affected.

By default, the option Remove zeroes is set to True, meaning that all zero values are removed when the sparse column is created.

* Run the experiment, or right-click the Convert to Dataset module and select Run selected.

**Results**

To save the resulting dataset with a new name, right-click the output of Convert to Dataset and select Save as Dataset.

**Convert to SVMLight**

This describes how to use the Convert to SVMLight module in Machine Learning Studio (classic), to convert your datasets to the format that is used by SVMLight.The SVM-Light framework was developed by researchers at Cornell University. The SVM-Light library implements Vapnik's Support Vector Machine, but the format has been adopted elsewhere and can be used for many machine learning tasks, including classification and regression.

**How to configure Convert to SVMLight**

The conversion to SVMLight format entails converting each case into a row of data that begins with the label, followed by feature-value pairs expressed as colon-separated numbers. The conversion process does not automatically identify the correct columns, so it is important that you prepare the columns in your dataset before attempting conversion. For more information, see Preparing Data for Conversion.

* Add the Convert to SVMLight module to your experiment. You can find this module in the Data Format Conversions category in Machine Learning Studio (classic).
* Connect the dataset or output that you want to convert to SVMLight format.
* Run the experiment.
* Right-click the output of the module, select Download, and save the data to a local file for modification or for reuse with a program that supports SVMLight.

**Convert to TSV**

This describes how to use the Convert to TSV module in Machine Learning Studio (classic), to convert any dataset from the internal format that is used by all Machine Learning Studio (classic) modules, to a flat file in tab-separated format.

Tab-separated value (TSV) files are compatible with many external tools, including:

* R and Python
* Excel and PowerPivot
* All relational databases

For example, if your experiment has an intermediate dataset that you would like to save for re-use in another tool or would like to call from code, you convert it to the TSV format, and then right-click the converted dataset to get the Python code needed to access the dataset.

**How to use Convert to TSV**

Use the Convert to TSV module whenever you need to download a dataset in tab-delimited format.

* Add the Convert to TSV to your experiment. You can find this module in the Data Format Conversions category in Machine Learning Studio (classic).
* Connect the module to another datset, or to a module that outputs a tabular dataset.
* Run the experiment, or right-click just the Convert to TSV module, and select Run selected.

**Results**

* When the conversion is complete, you can open the dataset, call it from R or Python code, use it in a Jupyter notebook, or save it to a local file.
* If you want to download the dataset, double-click the module output, and indicate whether you want to open or save the dataset.
* If you select Open, the dataset is loaded using whatever tool your computer uses by default to open.TSV files. Typically this is Microsoft Excel.
* If you select Download dataset, by default, the file is saved with the name of the module plus a GUID representing the workspace ID. However, you can select the Save As option during download and change the file name or location.