**Data Transformation in Microsoft Azure Machine Learning**

**Data Transformation**

This lists the modules that are provided in Machine Learning Studio (classic) for data transformation. For machine learning, data transformation entails some very general tasks, such as joining datasets or changing column names. But, it also includes many tasks that are specific to machine learning, such as normalization, binning and grouping, and inference of missing values.

Modules for data transformation are grouped into the following task-based categories:

* Creating filters for digital signal processing: Digital signal filters can be applied to numeric data to support machine learning tasks such as image recognition, voice recognition, and waveform analysis.
* Generating and using count-based features: Count-based featurization modules help you develop compact features to use in machine learning.
* General data manipulation and preparation: Merging datasets, cleaning missing values, grouping and summarizing data, changing column names and data types, or indicating which column is a label or a feature.
* Sampling and splitting datasets: Divide your data into training and test sets, split datasets by percentage or by a filter condition, or perform sampling.
* Scaling and reducing data: Prepare numerical data for analysis by applying normalization or by scaling. Bin data into groups, remove or replace outliers, or perform principal component analysis (PCA).

**Data Transformation – Filter**

Filters typically are applied to data in the data processing stage or the preprocessing stage. Filters enhance the clarity of the signal that's used for machine learning. For example, you can use the filter modules in Machine Learning Studio (classic) for these processing tasks:

* Clean up waveforms that are used for speech recognition.
* Detect trends or remove seasonal effects in noisy sales or economic data.
* Analyze patterns or artifacts in telemetry signals.

These modules provide easy configuration of filters by using well-researched algorithms to mathematically transform waveform data. You can also create a custom filter if you have already determined the correct coefficients to apply to your data.

If you need to do tasks such as excluding data from a dataset on a row-by-row basis, removing missing values, or reducing the size of a dataset, use these modules instead:

* Clean Missing Data: Remove missing values, or replace missing values with placeholders.
* Partition and Sample: Divide or filter your dataset by using criteria such as a range of dates, a specific value, or regular expressions.
* Clip Values: Set a range of values, and keep only the values within that range.

**Filters in digital signal processing**

Just like you can attach a filter to a camera to compensate for lighting or to create special effects, you can apply a filter to the data that you use for machine learning. Filters can help improve the clarity of a signal, capture interesting characteristics, or reduce noise.

The ideal filter would eliminate all noise and have uniform sensitivity for the desired signal. But, designing even a pretty good filter might take many iterations or combinations of techniques. If you succeed in designing an effective filter, consider saving the filter so that you can reuse it when you transform new data.

In general, filtering is based on the principles of waveform analysis. When you design a filter, you look for ways to suppress or amplify parts of the signal, to expose underlying trends, to reduce noise and interference, or to identify data values that otherwise might not be perceived.

Various techniques are applied to decompose individual trends or waveform components that create actual data values. The series of values can be analyzed by using trigonometric functions to identify and isolate individual waveforms. (This is true whether it's an econometric series or the composite frequencies of audio signals.) Filters can then be applied to these waveforms to eliminate noise, amplify some waves, or remove targeted components.

When filtering is applied to a noisy series to isolate different components, you can specify which frequencies to remove or strengthen by specifying the band of frequencies to work with.

**Digital filters in Machine Learning Studio (classic)**

The following types of filters are supported in Machine Learning Studio (classic):

* *Filters based on waveform decomposition*. Examples include finite impulse response (FIR) and infinite impulse response (IIR) filters. These filters work by removing specific components from an overall series. You can then view and investigate the simplified waveform.
* *Filters based on moving averages or median values.* These filters smooth out variations in a data series by averaging across windows of time. The windows can be fixed or sliding, and can have different shapes. For example, a triangular window peaks at the current data point (weights the current value stronger) and tails off before and after the data point (weights preceding and following values less strongly).
* *User-defined or custom filters*. If you already know the transformations that should be applied to a data series, you can create a user-defined filter. You provide the numeric coefficients that are applied to transform the data series. A custom filter can emulate an FIR or IIR filter. However, with a custom filter, you have more control over the values to apply at each point in the series.

**Filter terminology**

The following list includes simple definitions of terms that are used in the parameters and properties of filters:

* **Passband:** The range of frequencies that can pass through a filter without being attenuated or weakened.
* **Stopband:** A range of frequencies between specified limits through which signals are not passed. You define the stopband by setting cut-off frequencies.
* **High pass**: Let only high frequencies through.
* **Low pass:** Accept only frequencies below a specified cut-off value.
* **Corner:** Defines the boundary between the stopband and passband frequencies. Typically, you have the option to decide whether the corner is included in or excluded from the band. A first-order filter causes gradual attenuation until the corner frequency. After that, the filter causes exponential attenuation. Higher-order filters (such as Butterworth and Chebyshev filters) have steeper slopes after the corner frequency. Higher-order filters attenuate the values in the stopband much more rapidly and fully.
* **Bandstop filter** (also called a band reject filter or a notch filter): Has only one stopband. You define the stopband by specifying two frequencies: the high cut-off frequency and the low cut-off frequency. A bandpass filter typically has two stopbands: one on either side of the desired component.
* **Ripple:** A small, unwanted variation that occurs periodically. In Machine Learning, you can specify the amount of ripple to tolerate as part of the parameters in the IIR filter design.

**FIR Filter**

To define a kind of filter called a finite impulse response (FIR) filter. FIR filters have many applications in signal processing and are most commonly used in applications that require a linear-phase response. For example, a filter might be applied to images used in healthcare to sharpen the overall image, eliminate noise, or focus on an imaged object. After you have defined a digital signal processing filter, you can apply the filter to data by connecting a dataset and the filter to the Apply Filter module. You can also save the filter for re-use with similar datasets.

**How to configure FIR Filter**

* Add the FIR Filter module to your experiment. You can find this module under Data Transformation, in the Filter category.
* For Order, type an integer value that defines the number of active elements used to affect the filter's response. The order of the filter represents the length of the filter window.For a FIR filter, the minimum order is 4.
* For Window, select the shape of the data to which the filter will be applied. Machine Learning supports the following types of windowing functions for use in finite impulse response filters:

1. Hamming: The generalized Hamming window provides a type of weighted averaging, which is commonly used in image processing and computer vision.
2. Blackman: A Blackman window applies a smoothly tapered curve function to the signal. The Blackman window has better stopband attenuation than other window types.
3. Rectangular: A rectangular window applies a consistent value inside the specified interval and applies no value elsewhere. The simplest rectangular window might replace n values in a data sequence with zeros, which makes it appear as though the signal suddenly turns on and off.A rectangular window is also known as a boxcar or Dirichlet window.
4. Triangular: A triangular window applies filter coefficients in a step-wise fashion. The current value appears at the peak of the triangle, and then it declines with preceding or following values.
5. None: In some applications it is preferable not to use any windowing functions. For example, if the signal you are analyzing already represents a window or burst, applying a window function could deteriorate the signal-to-noise ratio.

* For Filter type, select an option that defines how the filter is applied. You can specify that the filter exclude the target values, alter the values, reject the values, or pass them through.

1. Lowpass: "Low pass" means that the filter passes through lower values, and removes the higher values. For example, you might use this to remove high-frequency noise and data peaks from a signal.This filter type has a smoothing effect on the data.
2. Highpass: "High pass" means that the filter passes through higher values, and removes lower values. You might use this to remove low frequency data, such as a bias or offset, from a signal.This filter type preserves sudden changes and peaks in a signal.
3. Bandpass: "Band pass" means that it passes thorugh the specified band of values, and remove others. You might use this filter to preserve the data from a signal with frequency characteristics at the intersection between the highpass and lowpass filters.Bandpass filters are created by combining a highpass and a lowpass filter. The highpass filter cutoff frequency represents the lower cutoff, and the lowpass filter frequency represents the higher cutoff.This filter type is good at removing a bias and smoothing a signal.
4. Bandstop: "Band stop" means that it blocks specified sigals. In other words, it removes data from a signal with frequency characteristics that are rejected by the low pass and the highpass filters.This filter type is good at preserving the signal bias and sudden changes.

* Depending on the type of filter you chose, you must set one or more cutoff values.

Use the High cutoff and Low cutoff options to define an upper and/or a lower threshold for values. One or both of these options are required to specify which values are rejected or passed through. A bandstop or bandpass filter requires that you set both high and low cutoff values. Other filter types, such as the lowpass filter, require that you set only the low cutoff value.

* Select the Scale option if scaling should be applied to coefficients; otherwise leave blank.
* Connect the filter to Apply Filter, and connect a dataset.

Use the column selector to specify which columns the filter should be applied to. By default, the Apply Filter module will use the filter for all selected numeric columns.

* Run the experiment.

No computations are performed until you connect a dataset to the Apply Filter module and run the experiment. At that point, the specified transformation is applied to the selected numeric columns.

**Implementation details**

FIR filters have these characteristics:

* FIR filters do not have feedback; that is, they use the previous filter outputs.
* FIR filters are more stable, because the impulse response will always return to 0.
* FIR filters require a higher order to achieve the same selectivity as infinite impulse response (IIR) filters.
* Like other filters, the FIR filter can be designed with a specific cutoff frequency that preserves or rejects frequencies that compose the signal.

**Calculating coefficients over the filter window**

The window type determines the trade-off between selectivity (width of the transition band in which frequencies are neither fully accepted nor rejected) and suppression (the total attenuation of frequencies to be rejected). The windowing function is applied to the ideal filter response to force the frequency response to zero outside of the window. Coefficients are selected by sampling the frequency response within the window.

The number of coefficients returned by the FIR Filter module is equal to the filter order plus one. The coefficient values are determined by filter parameters and by the windowing method, and are symmetric to guarantee a linear phase response

**Scaling of coefficients**

The FIR Filter module returns filter coefficients, or tap weights, for the created filter.

The coefficients are determined by the filter, based on the parameters you enter (such as the order). If you want to specify custom coefficients, use the User-Defined Filter module.

When Scale is set to True, filter coefficients will be normalized, such that the magnitude response of the filter at the center frequency of the passband is 0. The implementation of normalization in Machine Learning Studio (classic) is the same as in the fir1 function in MATLAB.

Typically, in the window design method, you design an ideal infinite impulse response (IIR) filter. The window function is applied to the waveform in the time domain, and multiplies the infinite impulse response by the window function. This results in the frequency response of the IIR filter being convolved with the frequency response of the window function. However, in the case of FIR filters, the input and filter coefficients (or tap weights) are convolved when applying the filter.

**Selectivity and stop band attenuation**

The following table compares selectivity with stop band attenuation for a FIR filter with length n by using different windowing methods:

|  |  |  |
| --- | --- | --- |
| Window Type | Transition Region | Minimum Stopband Attenuation |
| Rectangular | 0.041n | 21dB |
| Triangle | 0.11n | 26dB |
| Hann | 0.12n | 44dB |
| Hamming | 0.23n | 53dB |
| Blackman | 0.2n | 75dB |

**Module parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Range** | **Type** | **Default** | **Description** |
| Order | >=4 | Integer | 5 | Specify the filter order |
| Window | Any | WindowType |  | Specify the type of window to apply |
| Filter type | Any | FilterType | LowPass | Select the type of filter to create |
| Low Cutoff | [double.Epsilon  ;.9999999] | Float | 0.3 | Set the low cutoff frequency |
| High Cutoff | [double.Epsilon  ;.9999999] | Float | 0.7 | Set the high cutoff frequency |
| Scale | Any | Boolean | True | If true, filter coefficients will be normalized |

**IIR Filter**

To create an infinite impulse response (IIR) filter.Filters are an important tool in digital signal processing, and are used to improve the results of image or voice recognition. In general, a filter is a transfer function that takes an input signal and creates an output signal based on the filter characteristics. For more general information about the user of filters in digital signal processing.

An IIR filter is a particular type of filter; typical uses of an IIR filter would be to simplify cyclical data that includes random noise over a steadily increasing or decreasing trend. The IIR filter you create with this module defines a set of constants (or coefficients) that alter the signal that is passed through. The word infinite in the name refers to the feedback between the outputs and the series values.

After you have defined a filter that meets your needs, you can apply the filter to data by connecting a dataset and the filter to the Apply Filter module.

**How to configure IIR Filter**

* Add the IIR Filter module to your experiment. You can find this module under Data Transformation, in the Filter category.
* For Order, type an integer value that defines the number of active elements used to affect the filter's response. The order of the filter represents the length of the filter window. For an IIR filter, the minimum order is 4.
* For Filter kind, choose the algorithm that is used to compute filter coefficients. The filter kind designates the mathematical transfer function that controls frequency response and frequency suppression. Machine Learning supports these kinds of filters commonly used in digital signal processing:

1. **Butterworth:** A Butterworth filter is also called a maximally flat magnitude filter because it constrains the response (change in signal) in the passband and the stopband.
2. **Chebyshev Type 1:** Chebyshev filters are intended to minimize the errors between the idealized and the actual filter characteristics over the range of the filter. Type 1 Chebyshev filters leave more ripple in the passband.
3. **Chebyshev Type 2:** Type 2 Chebyshev filters have the same general characteristics as Type 1 Chebyshev filters, but they leave more ripple in the stopband.

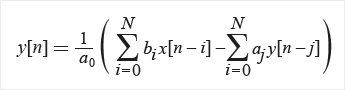
* For Filter type, select an option that defines how the filter will affect the values in the input signal. You can specify that the filter exclude values above or below a cutoff point, or specify that the filter either reject or pass through values in a specified frequency range.

1. LowPass: Allows low frequency values (below the cutoff value) to pass and attenuates other values.
2. HighPass: Allows high frequency values (above the cutoff value) to pass and attenuates other values.
3. Bandpass: Allows signals in the range that is specified by the low and high cutoff values to pass and attenuates other values.
4. BandStop: Allows signals outside the range specified by the low and high cutoff values to pass and attenuates values within the range.

* Specify the high or low cutoff values, or both, as a value between 0 and 1, representing a normalized frequency. For High cutoff, type a value that represents the upper-frequency boundary. For Low cutoff, type a value that represents the lower frequency boundary.
* For Ripple, specify the amount of ripple to tolerate when you define your filter. Ripple refers to a small variation that occurs periodically. Ripple is usually considered an unwanted effect, but you can compensate for ripple by adjusting other filter parameters, such as the filter length. Not all filters produce ripple.
* Add the Apply Filter module to your experiment, and connect the filter you designed, and the dataset containg the values you want to modify.Use the column selector to specify which columns of the dataset to which the filter should be applied. By default, the Apply Filter module will use the filter for all selected numeric columns.
* Run the experiment to apply the transformation.

**Implementation details**

An IIR filter returns feed forward and feed backward coefficients, which are represented by way of a transfer function. Here is an example representation:



Where:

* N: filter order
* bi: feed forward filter coefficients
* ai: feed backward filter coefficients
* x[n]: the input signal
* y[n]: the output signal

**Module parameters**

| **Name** | **Range** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| Order | [4;13] | Integer | 5 | Specify the filter order |
| Filter kind | Any | IIRFilterKind |  | Select the kind of IIR filter to create |
| Filter type | Any | FilterType |  | Select the filter band type |
| Low cutoff | [double.Epsilon;.9999999] | Float | 0.3 | Set the low cutoff value |
| High cutoff | [double.Epsilon;.9999999] | Float | 0.7 | Set the high cutoff value |
| Ripple | >=0.0 | Float | 0.5 | Specify the amount of ripple in the filter |

**Median Filter**

Creates a median filter used to smooth data for trend analysis

To define a *median filter* for applying to a series of values that represent a digital input signal or image.Median filters are widely used in image recognition to reduce noise so that features can more easily be detected.

After you have defined a filter transformation that meets your needs by using the Median Filter module, you can apply the filter to data by connecting a dataset and the filter to the Apply Filter module.

**How to configure Median Filter**

* Add the Median Filter to your experiment. You can find this module under Data Transformation, in the Filter category.
* For Length, type an integer value that defines the total size of the window across which the filter is applied. This is also called the filter mask.

The value should be an odd, positive-valued integer. If you specify an even number, the mask size is reduced by one.

By default the mask begins at the current value and creates a window centered on the current value.

For example, if you type 5 as the Length or window size, the median value is computed across a sliding window consisting of 5 values centered on the current value. If you type 4, the mask is reduced to 3 values, centered on the index value.

* Connect the filter to Apply Filter, and connect a dataset.

Use the column selector to specify which columns of the dataset to which the filter should be applied. By default, the Apply Filter module will use the filter for all selected numeric columns.

* Run the experiment. The following operations are applied to the selected columns:

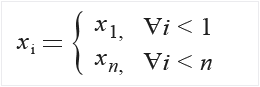
1. For each set of values included in the window or mask, the filter algorithm computes the median.
2. The current (or index) value is replaced with the median value.

**Implementation details**

Each entry in the output signal is equal to the median of the entries in a subset (mask) of the input signal, and centered at the corresponding index. The mask size should be an odd, positive-valued integer.

If you provide this method with an even-valued mask size, it is reduced by one. For example, given m=2q+1, the filter is defined as: yi = median[{xi-q,…, xi+q}]

Values beyond the borders of the input signal are assumed to equal the value at the border. That is, if *n* is the length of the input signal:

****

**Moving Average Filter**

To calculate a series of one-sided or two-sided averages over a dataset, using a window length that you specify.

After you have defined a filter that meets your needs, you can apply it to selected columns in a dataset by connecting it to the Apply Filter module. The module does all the calculations and replaces values within numerical columns with corresponding moving averages.

You can use the resulting moving average for plotting and visualization, as a new smooth baseline for modeling, for calculating variances against calculations for similar periods, and so on.

**Understanding and using moving averages**

This type of average helps you reveal and forecast useful temporal patterns in retrospective and real-time data. The simplest type of moving average starts at some sample of the series, and uses the average of that position plus the previous n positions instead of the actual value. (You can define n as you like.) The longer the period n across which the average is computed, the less variance you will have among values. Also, as you increase the number of values used, the less effect any single value has on the resulting average.

A moving average can be one-sided or two-sided. In a one-sided average, only values preceding the index value are used. In a two-sided average, past and future values are used.

For scenarios in which you are reading streaming data, cumulative and weighted moving averages are particularly useful. A cumulative moving average takes into account the points preceding the current period.

You can weight all data points equally when computing the average, or you can ensure that values nearer the current data point are weighted more strongly. In a weighted moving average, all weights must sum to 1.

In an exponential moving average, the averages consist of a head and a tail, which can be weighted. A lightly weighted tail means that the tail follows the head quite closely, so the average behaves like a moving average on a short weighting period. When tail weights are heavier, the average behaves more like a longer simple moving average.

**How to configure Moving Average Filter**

* Add the Moving Average Filter module to your experiment. You can find this module under Data Transformation, in the Filter category.
* For Length, type a positive whole number value that defines the total size of the window across which the filter is applied. This is also called the filter mask. For a moving average, the length of the filter determines how many values are averaged in the sliding window.

Longer filters are also called higher order filters, and provide a larger window of calculation and a closer approximation of the trend line.

Shorter or lower order filters use a smaller window of calculation and more closely resemble the original data.

* For Type, choose the type of moving average to apply.

Machine Learning Studio (classic) supports the following types of moving average calculations:

Simple: A simple moving average (SMA) is calculated as an unweighted rolling mean.

1. Triangular: Triangular moving averages (TMA) are averaged twice for a smoother trend line. The word triangular is derived from the shape of the weights that are applied to the data, which emphasizes central values.
2. Exponential Simple: An exponential moving average (EMA) gives more weight to the most recent data. The weighting drops off exponentially.
3. Exponential: A modified exponential moving average calculates a running moving average, where calculating the moving average at any one point considers the previously computed moving average at all preceding points. This method yields a smoother trend line.
4. Cumulative: Given a single point and a current moving average, the cumulative moving average (CMA) calculates the moving average at the current point.

* Add the dataset that has the values you want to compute a moving average for, and add the Apply Filter module.

Connect the Moving Average Filter to the left-hand input of Apply Filter, and connect the dataset to the right-hand input.

* In the Apply Filter module, use the column selector to specify which columns the filter should be applied to. By default, the filter transformation is applied to all numeric columns, so be sure to exclude any columns that don’t have appropriate data.
* Run the experiment.

For each set of values defined by the filter length parameter, the current (or index) value is replaced with the moving average value.

**Threshold Filter**

To define a filter that restricts numeric values to a specified range.

Threshold filters are commonly used in digital signal processing. A threshold filter examines each value of the input dataset and changes all values that do not meet the boundary conditions. You typically would use this type of filter for the following applications:

* Replace all negatively signed measurements with a value of zero.
* Convert a gray-scale image to black and white areas by defining a numerical boundary value for all pixels.

After you have defined a filter that meets your needs, you can apply the filter to data by connecting a dataset and the filter to the Apply Filter module.

The output of the Apply Filter module is a dataset containing the selected columns, transformed as specified by the Threshold Filter settings. Alternatively, if you select the Indicator option, instead of returning the filter values, a column is returned containing Boolean values that indicates whether the value in each row met the specified filter condition or not. This can be useful when you are testing a new filter.

**How to configure Threshold Filter**

* Add the Threshold Filter module to your experiment. You can find this module under Data Transformation, in the Filter category.
* For Type, specify the type of filter to apply:

1. LessThan: Changes values that are less than the specified level to the boundary level, and passes through all other values.
2. GreaterThan: Changes values that are greater than the specified level to the boundary level, and passes through all other values.
3. MagnitudeLessThan: Changes values less than the specified level to the boundary level but preserves the sign of the original value.
4. MagnitudeGreaterThan: Changes values greater than the specified level to the boundary level but preserves the sign of the original value.
5. InRange: Passes through all values that fall within the specified range, and changes values outside the range to the closest boundary value.
6. OutOfRange: Passes through all values that fall outside the specified range, and changes values inside the range to the closest boundary value.
7. InRangeWithStd: Passes through all values that fall within the specified range of standard deviations, and changes values outside the range to the closest boundary value.
8. OutOfRangeWithStd: Passes through all values that fall outside the specified range of standard deviations, and changes values inside the range to the closest boundary value.

* For Level, type the boundary value to apply in each type of threshold.

1. If you select the LessThan filter, the number you specify defines the lowest value that can be passed through without replacement.
2. If you select the GreaterThan filter, the number you specify defines the greatest value that can be passed through without replacement.
3. If you select the MagnitudeLessThan filter, type a single positive or negative number for Level. Any value that is less than that value is replaced with the level value.
4. If you select the MagnitudeGreaterThan filter, type a single positive or negative number for Level. Any value that is greater than that value is replaced with the level value.
5. If you select the filters, InRange orOutOfRange, specify the upper or lower bounds. For Lower boundary, type the lowest number to include in the range. For Upper boundary, type the highest number to include in the range.
6. If you chose one of the filter types that uses standard deviations (InRangeWithStd, OutOfRangeWithStd), you must specify the Alpha constant. The values of alpha times the deviation is used to calculate the filter result.

* Optionally, select the Indicator option to generate a column that only indicates whether the value would be affected by the filter. If you leave the Indicator unselected, the filter generates the replacement values.
* Connect the filter to Apply Filter, and connect a dataset.

Use the column selector to specify which columns the filter should be applied to. By default, the Apply Filter module applies the filter transformation to all selected numeric columns.

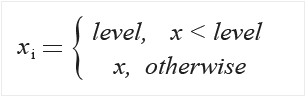
* Run the experiment.

No computations are performed until you connect a dataset to the Apply Filter module and run the experiment. At that point, the specified transformation is applied to the selected numeric columns.

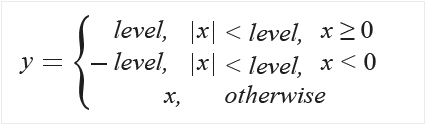
**Implementation details**

The **Threshold Filter** module uses the following methods to define threshold values, depending on the filter type:

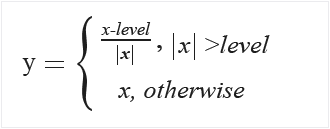
* **LessThan**: The less-than mode is defined as:



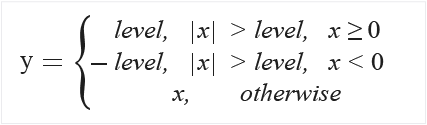
**MagnitudeLessThan**: The less-than-magnitude mode is defined as:



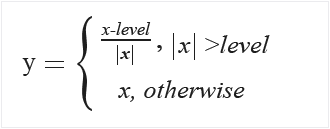
For complex inputs, the magnitude of each element is restricted as shown by this formula:



* **MagnitudeGreaterThan**: The greater than magnitude mode is defined as:



For complex inputs, the magnitude of each element is restricted as shown by this formula:



**User-Defined Filter**

To define a custom filter by using a finite impulse response (FIR) filter or an infinite impulse response (IIR) filter with coefficients that you specify.

A filter is a transfer function that takes an input signal and creates an output signal based on the filter characteristics. For more general information about the user of filters in digital signal processing, see Filter. This module is particularly useful for applying a set of previously derived filter coefficients to your data.

After you have defined a filter that meets your needs, you can apply the filter to data by connecting a dataset and the filter to the Apply Filter module.

**How to configure User-Defined Filter**

* Add the User-Defined Filter module to your experiment in Studio (classic). You can find this module under Data Transformation, in the Filter category.
* In the Properties pane, choose a type of filter: FIR filter, or IIR filter.
* Provide the coefficients to apply in the filter. The requirements for the coefficients differ depending on whether you choose a FIR filter or an IIR filter.

1. For a FIR filter, you specify a vector of feed-forward coefficients. The length of the vector determines the filter's order. A FIR filter is effectively a moving average, so the configuration values apply a moving average to filter a data sequence.
2. For an IIR filter, you apply custom feed-forward and feed-backward coefficients. See the Examples section for some tips.

* Connect the filter to Apply Filter, and connect a dataset.

Use the column selector to specify which columns of the dataset to which the filter should be applied. By default, the Apply Filter module will use the filter for all selected numeric columns.

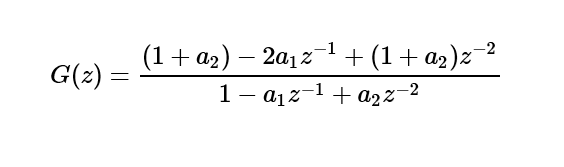
* Run the experiment.

The specified transformations are applied to the selected numeric columns only when you run the experiment using Apply Filter

**IIR filter example: Notch filter**

A good example of an application for a user-defined IIR filter is to define a *notch filter*, also called a *bandstop filter*. The desired notch filter attenuates a -3dB rejection band, *fb*, centered at a notch frequency, fn, with a sampling frequency, fs.

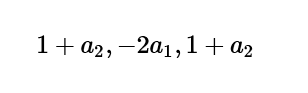
In this case, the digital notch filter can be represented by the following formula:



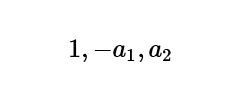
This formula assumes:

custom notch filter

From this formula, we can get the feed-forward coefficient:

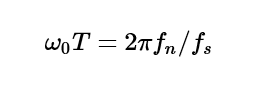


The feed-backward coefficients would be as follows:

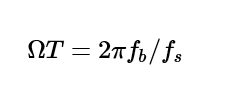


**Example of IIR Filter: Notch filter 2**

The following example shows a notch filter with a notch frequency of fn =1250 Hz and a -3 dB rejection band of fb =100 Hz, with sampling frequency of fs=10 kHz.



Using the following formula, you get a2 = 0.93906244 and a1 = 1.3711242:



From this, you can get the following feed-forward (b) and feed-backward (a) coefficients:

b= 1.9390624, -2.7422484, 1.9390624

a= 1, -1.3711242, 0.9390624