* [Condition Monitoring (Neural Net/C5.0)](http://127.0.0.1:54857/help/topic/com.ibm.spss.modeler.tutorial/clementine/excmoverview.htm)

**Condition Monitoring (Neural Net/C5.0)**

This example concerns monitoring status information from a machine and the problem of recognizing and predicting fault states. The data is created from a fictitious simulation and consists of a number of concatenated series measured over time. Each record is a snapshot report on the machine in terms of the following:

* *Time*. An integer.
* *Power*. An integer.
* *Temperature*. An integer.
* *Pressure*. 0 if normal, 1 for a momentary pressure warning.
* *Uptime*. Time since last serviced.
* *Status*. Normally 0, changes to error code on error (101, 202, or 303).
* *Outcome*. The error code that appears in this time series, or 0 if no error occurs. (These codes are available only with the benefit of hindsight.)

This example uses the streams named *condplot.str* and *condlearn.str*, which reference the data files named *COND1n* and *COND2n*. These files are available from the*Demos* directory of any IBM® SPSS® Modeler installation. This can be accessed from the IBM SPSS Modeler program group on the Windows Start menu. The *condplot.str* and *condlearn.str* files are in the *streams*directory.

For each time series, there is a series of records from a period of normal operation followed by a period leading to the fault, as shown in the following table:

| **Time** | **Power** | **Temperature** | **Pressure** | **Uptime** | **Status** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1059 | 259 | 0 | 404 | 0 | 0 |
| 1 | 1059 | 259 | 0 | 404 | 0 | 0 |
|  |  |  | ... |  |  |  |
| 51 | 1059 | 259 | 0 | 404 | 0 | 0 |
| 52 | 1059 | 259 | 0 | 404 | 0 | 0 |
| 53 | 1007 | 259 | 0 | 404 | 0 | 303 |
| 54 | 998 | 259 | 0 | 404 | 0 | 303 |
|  |  |  | ... |  |  |  |
| 89 | 839 | 259 | 0 | 404 | 0 | 303 |
| 90 | 834 | 259 | 0 | 404 | 303 | 303 |
| 0 | 965 | 251 | 0 | 209 | 0 | 0 |
| 1 | 965 | 251 | 0 | 209 | 0 | 0 |
|  |  |  | ... |  |  |  |
| 51 | 965 | 251 | 0 | 209 | 0 | 0 |
| 52 | 965 | 251 | 0 | 209 | 0 | 0 |
| 53 | 938 | 251 | 0 | 209 | 0 | 101 |
| 54 | 936 | 251 | 0 | 209 | 0 | 101 |
|  |  |  | ... |  |  |  |
| 208 | 644 | 251 | 0 | 209 | 0 | 101 |
| 209 | 640 | 251 | 0 | 209 | 101 | 101 |

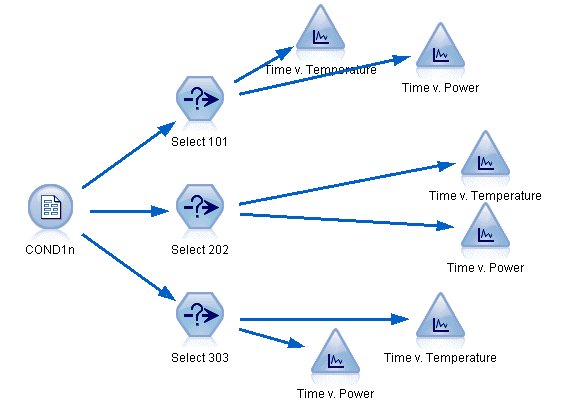
The following process is common to most data mining projects:

* Examine the data to determine which attributes may be relevant to the prediction or recognition of the states of interest.
* Retain those attributes (if already present), or derive and add them to the data, if necessary.
* Use the resultant data to train rules and neural nets.
* Test the trained systems using independent test data.

# Examining the Data

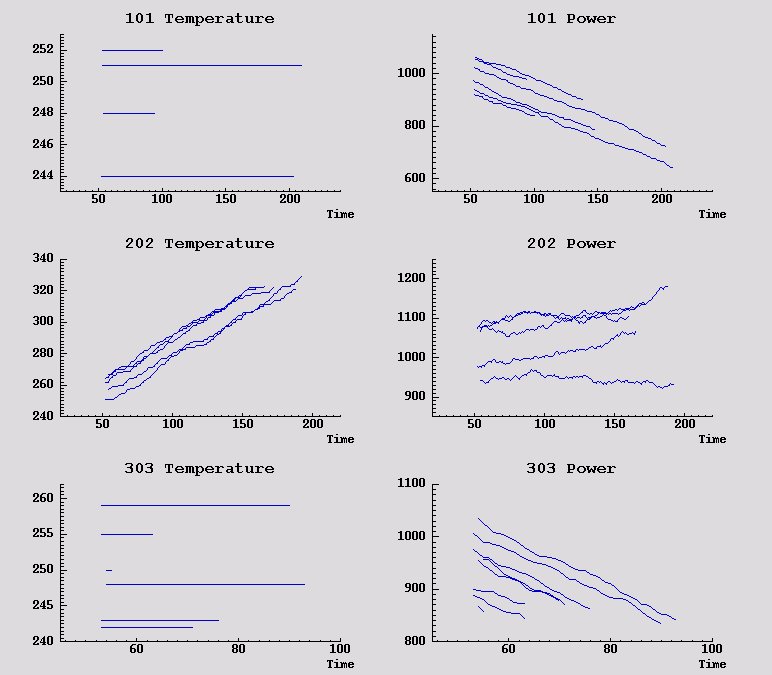
The file condplot.str illustrates the first part of the process. It contains a stream that plots a number of graphs. If the time series of temperature or power contains visible patterns, you could differentiate between impending error conditions or possibly predict their occurrence. For both temperature and power, the stream below plots the time series associated with the three different error codes on separate graphs, yielding six graphs. Select nodes separate the data associated with the different error codes.

*Figure 1. Condplot stream*



The results of this stream are shown in this figure.

*Figure 2. Temperature and power over time*



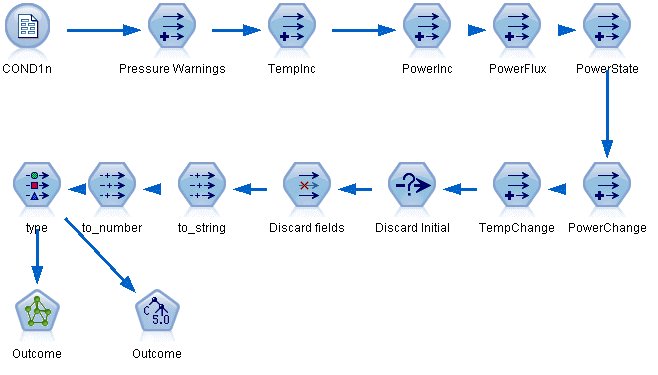
The graphs clearly display patterns distinguishing 202 errors from 101 and 303 errors. The 202 errors show rising temperature and fluctuating power over time; the other errors do not. However, patterns distinguishing 101 from 303 errors are less clear. Both errors show even temperature and a drop in power, but the drop in power seems steeper for 303 errors.

Based on these graphs, it appears that the presence and rate of change for both temperature and power, as well as the presence and degree of fluctuation, are relevant to predicting and distinguishing faults. These attributes should therefore be added to the data before applying the learning systems.

**Data Preparation**

Based on the results of exploring the data, the stream *condlearn.str* derives the relevant data and learns to predict faults.

*Figure 1. Condlearn stream*



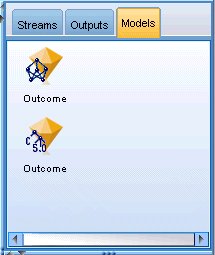
The stream uses a number of Derive nodes to prepare the data for modeling.

* **Variable File node**. Reads data file *COND1n*.
* **Derive Pressure Warnings**. Counts the number of momentary pressure warnings. Reset when time returns to 0.
* **Derive TempInc**. Calculates momentary rate of temperature change using @DIFF1.
* **Derive PowerInc**. Calculates momentary rate of power change using @DIFF1.
* **Derive PowerFlux**. A flag, true if power varied in opposite directions in the last record and this one; that is, for a power peak or trough.
* **Derive PowerState**. A state that starts as *Stable* and switches to *Fluctuating* when two successive power fluxes are detected. Switches back to *Stable* only when there hasn't been a power flux for five time intervals or when *Time* is reset.
* **PowerChange**. Average of *PowerInc* over the last five time intervals.
* **TempChange**. Average of *TempInc* over the last five time intervals.
* **Discard Initial (select)**. Discards the first record of each time series to avoid large (incorrect) jumps in *Power* and *Temperature* at boundaries.
* **Discard fields**. Cuts records down to *Uptime*, *Status*, *Outcome*, *Pressure Warnings*, *PowerState*, *PowerChange*, and *TempChange*.
* **Type**. Defines the role of *Outcome* as **Target** (the field to predict). In addition, defines the measurement level of *Outcome* as **Nominal**, *Pressure Warnings* as **Continuous**, and *PowerState* as **Flag**.

# Learning

Running the stream in condlearn.str trains the C5.0 rule and neural network (net). The network may take some time to train, but training can be interrupted early to save a net that produces reasonable results. Once the learning is complete, the Models tab at the upper right of the managers window flashes to alert you that two new nuggets were created: one represents the neural net and one represents the rule.

*Figure 1. Models manager with model nuggets*



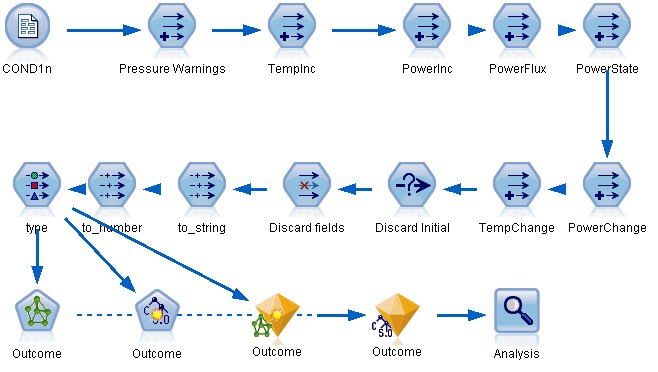
The model nuggets are also added to the existing stream, enabling us to test the system or export the results of the model. In this example, we will test the results of the model.

# Testing

The model nuggets are added to the stream, both of them connected to the Type node.

1. Reposition the nuggets as shown, so that the Type node connects to the neural net nugget, which connects to the C5.0 nugget.
2. Attach an Analysis node to the C5.0 nugget.
3. Edit the original source node to read the file COND2n (instead of COND1n), as COND2n contains unseen test data.

*Figure 1. Testing the trained network*



1. Open the Analysis node and click Run.

Doing so yields figures reflecting the accuracy of the trained network and rule.