

# Social Signal Interpretation XML Tutorial

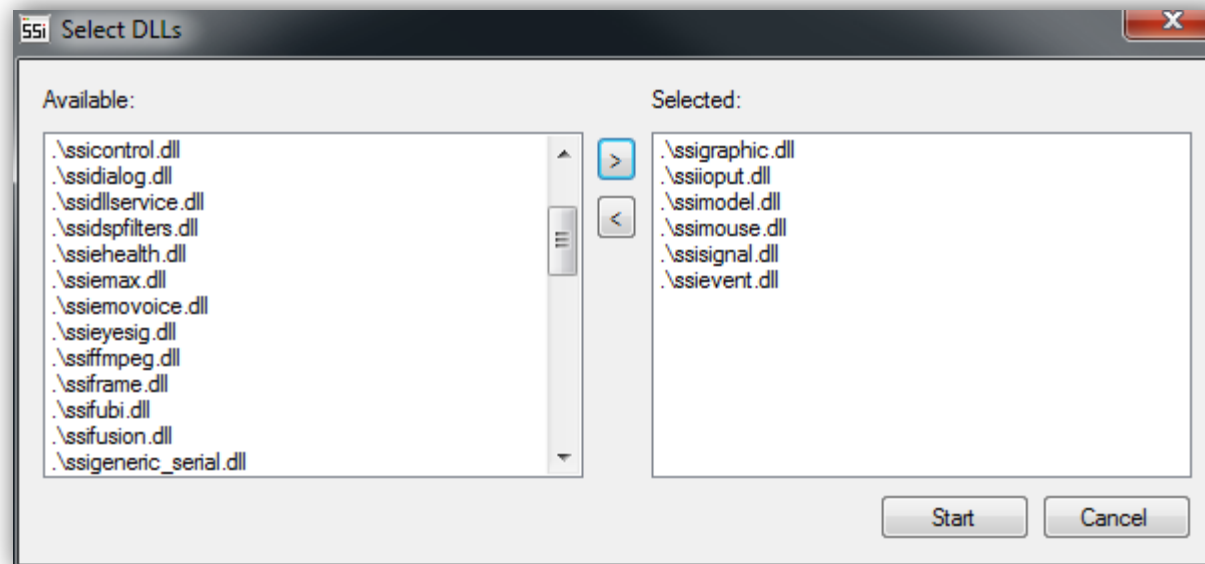
- Johannes Wagner <wagner@openssi.net>
- (last update: 12.01.16)

<http://openssi.net>

# XML Editor

1. Start `bin/<Win32|x64>/<vc100|vc120>/xmledit.exe`
2. Select DLLs you want to use (press 'strg' to select multiple)

`ssievent.dll`, `ssigraphic.dll`, `ssioput.dll`,  
`ssimodel.dll`, `ssimouse.dll`, `ssisignal.dll`

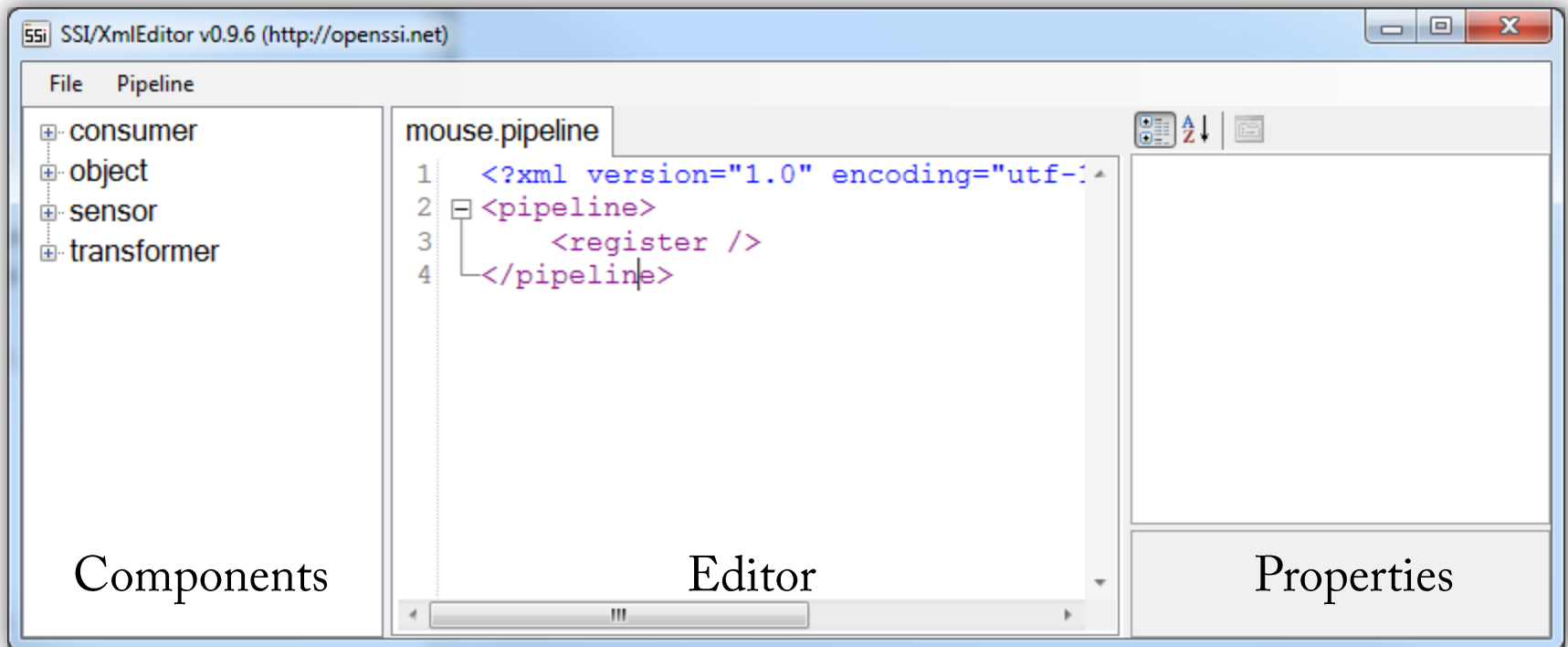


Not selected

Selected

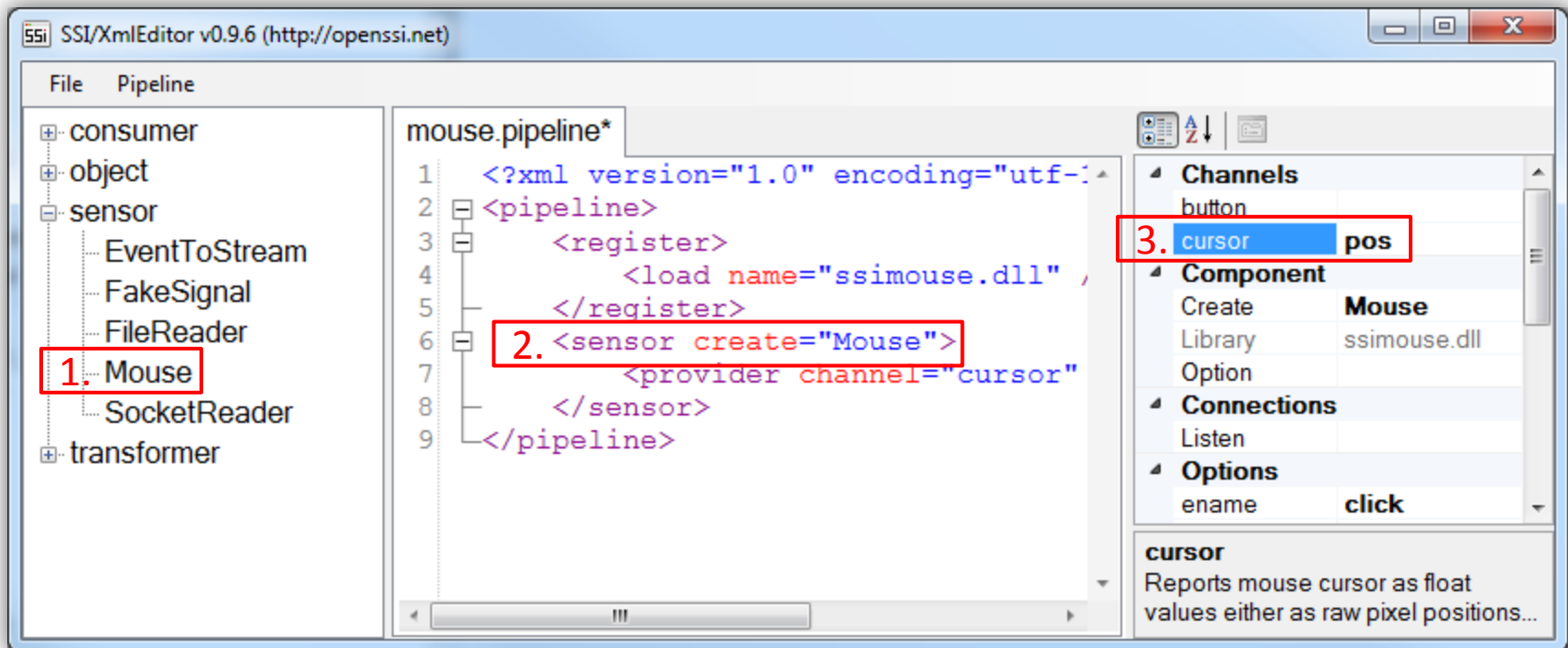
# Pipeline

1. Start `bin/<Win32|x64>/<vc100|vc120>/xmledit.exe`
2. Create a new pipeline (File>New) and save it (File->Save)



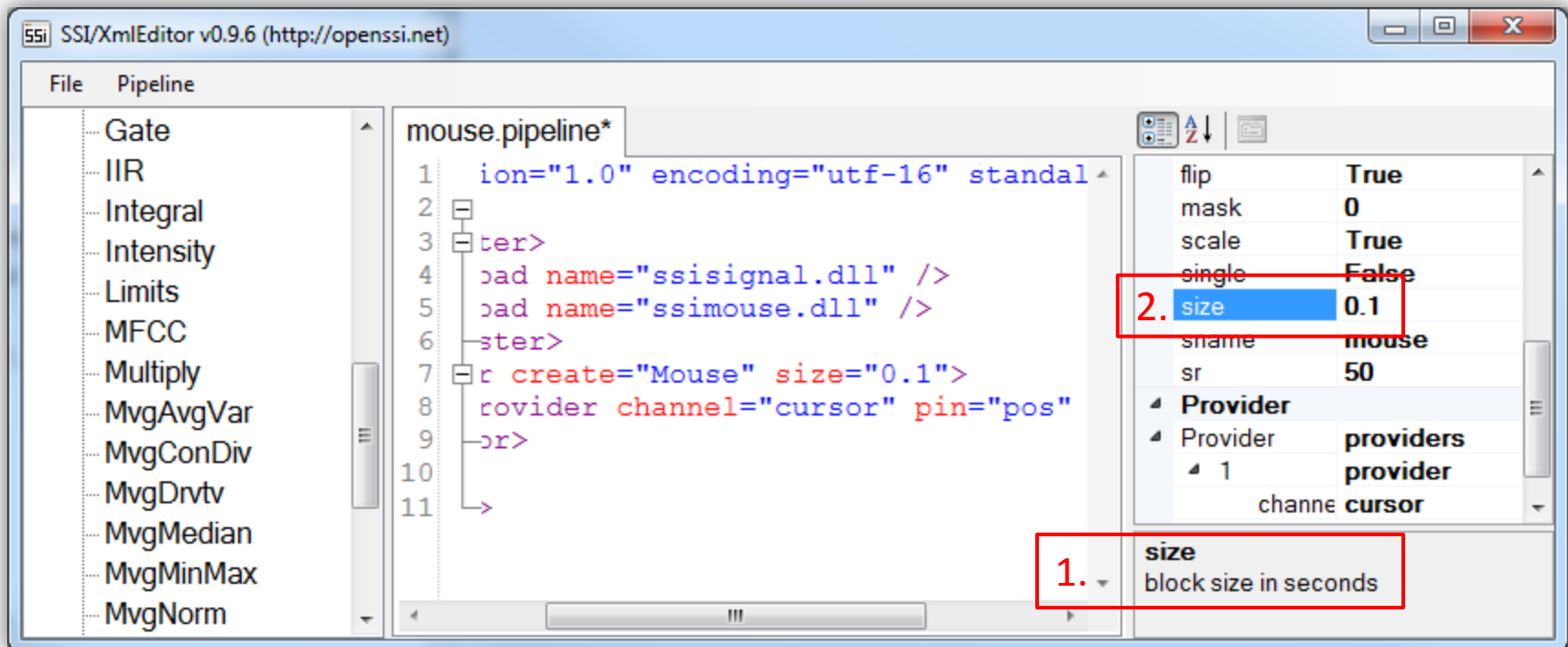
# Cursor Provider

1. Expand the **sensor**-tree and insert a **Mouse** (double click)
2. Place cursor in line `<sensor...` to display properties
3. Below **Channels** assign pin-name to **cursor** (e.g. **pos**)



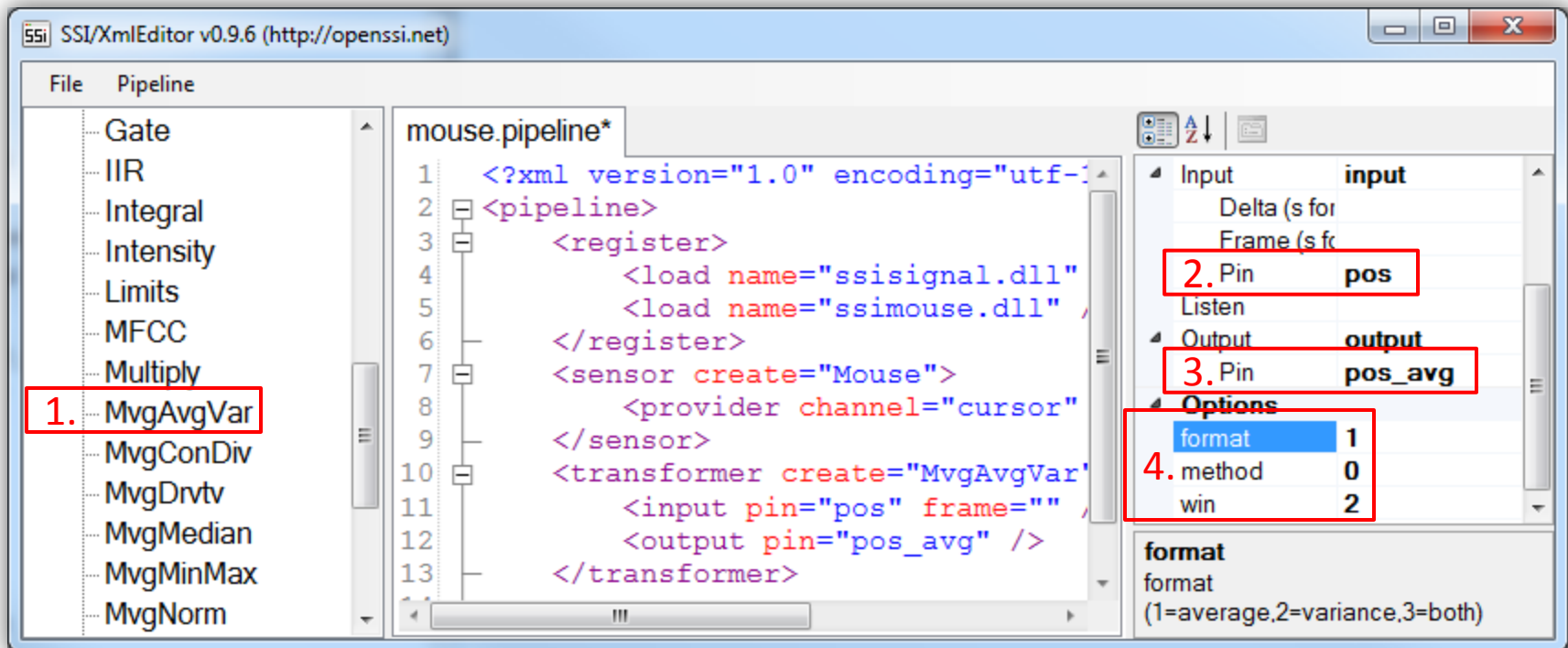
# Change Options

1. When you click on an option in the property panel you'll find additional information at the bottom of the panel
2. Set option `size` to `0.1`



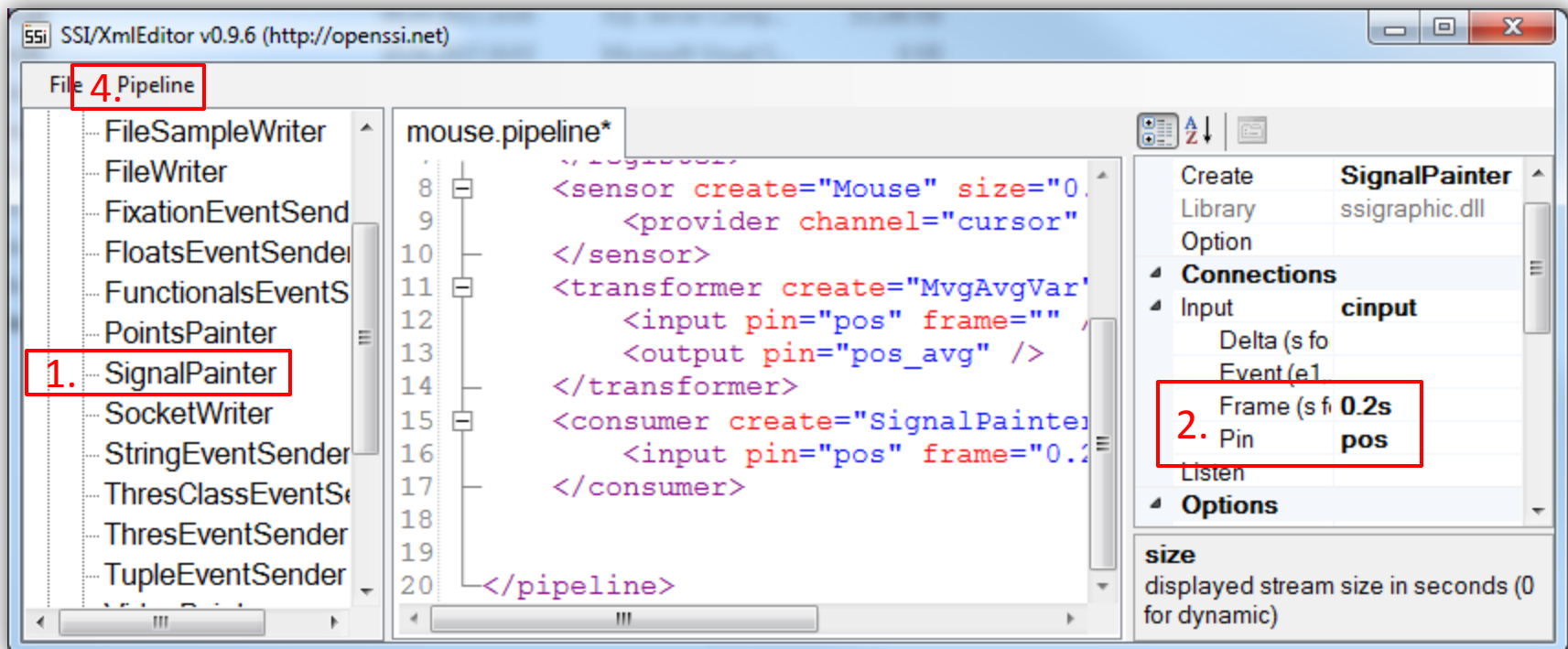
# Smoothing Transformer

1. Place cursor in a new line below `</sensor>` and insert a `MvgAvgVar` (expand `transformer`-tree)
2. Set input pin to (`pos`) and assign an output pin (`pos_avg`)
3. Set frame size as samples per second (e.g. `"10"`) or as seconds (e.g. `"0.2s"`)
4. Set option `format` to `1` and option `win` to `2`



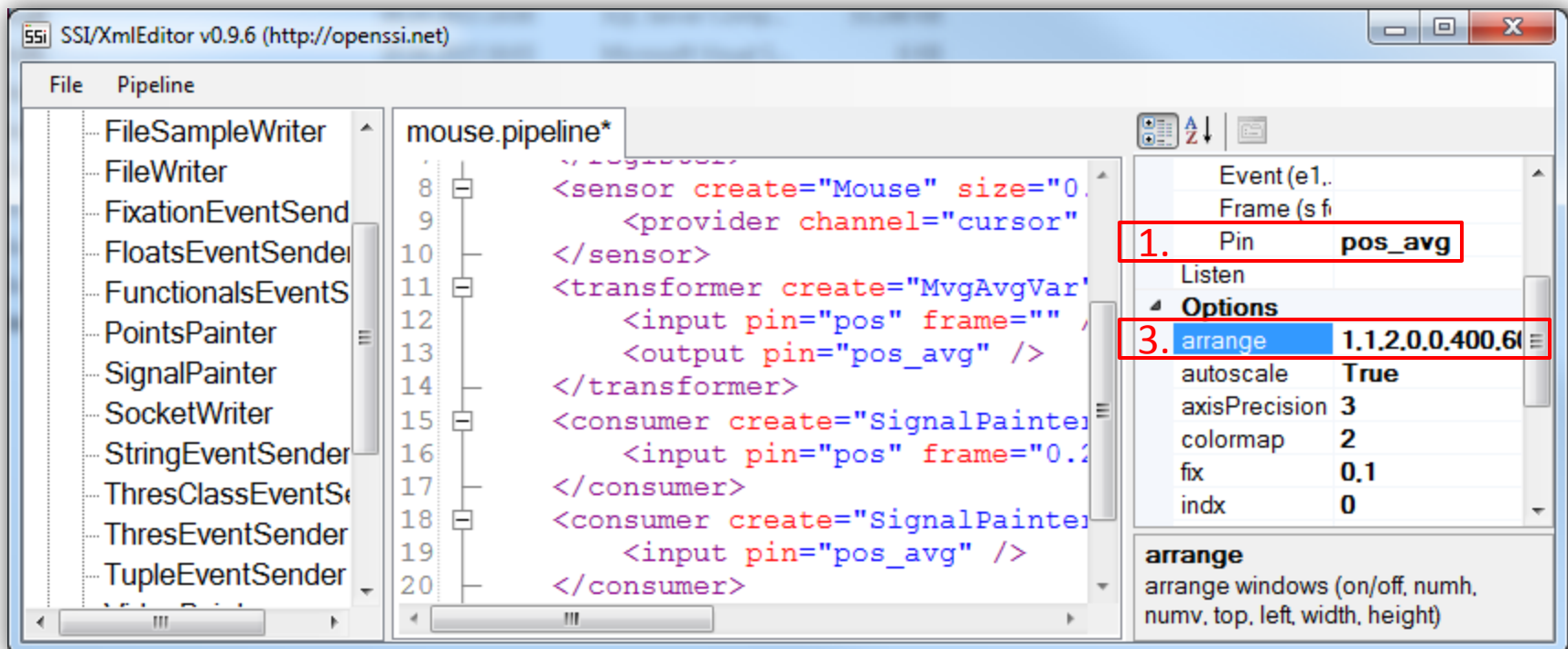
# Plot Consumer

1. Place cursor in a new line below `</transformer>` and add a **SignalPainter** (expand **consumer**-tree)
2. Set input pin to **pos** and choose frame size (e.g. **0.2s**)
3. Set option **size** to **10.0**
4. Press **F5** to run pipeline



# Another Plot Consumer

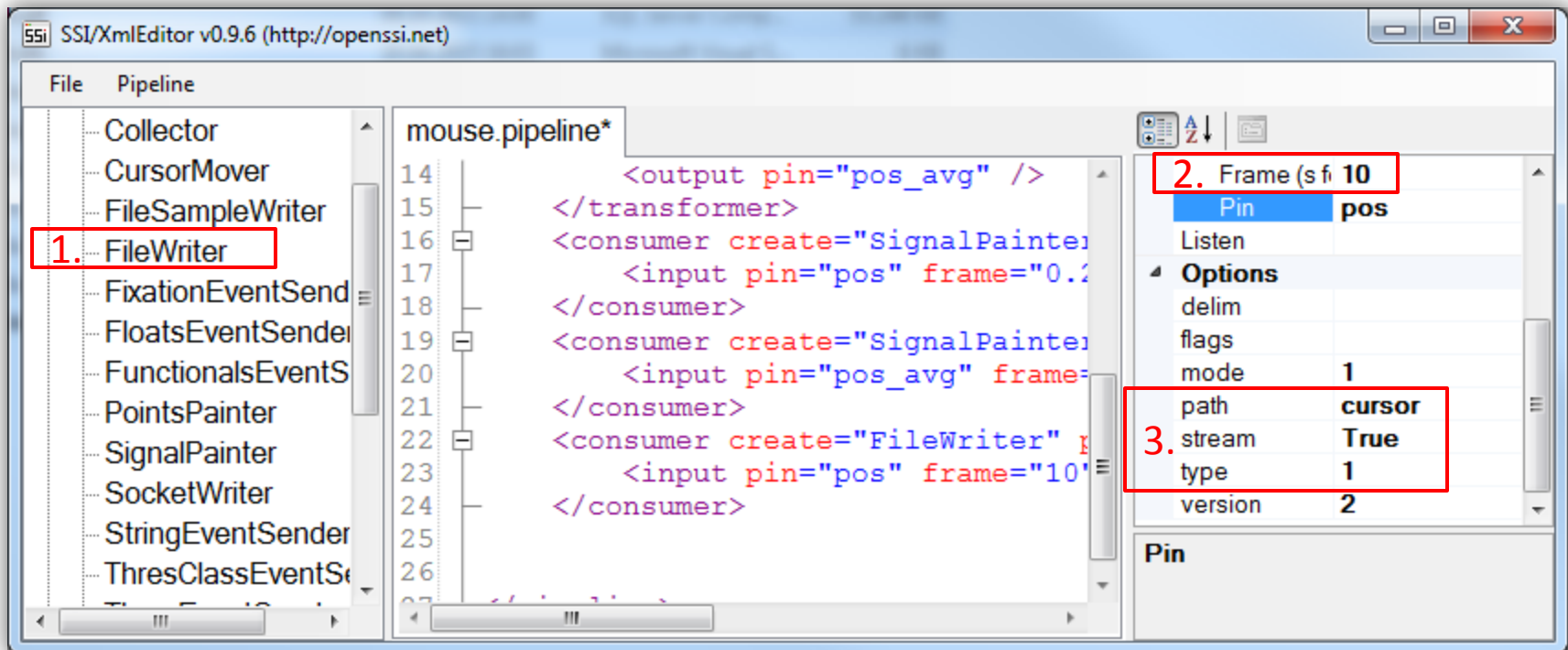
1. To compare the raw **cursor** signal with the filtered version insert another **SignalPainter** consumer and connect it with input pin **cursor\_avg** (and same frame size as before)
2. Set option **size** to **10.0**
3. Set option **arrange** to **1,1,2,0,0,400,600**





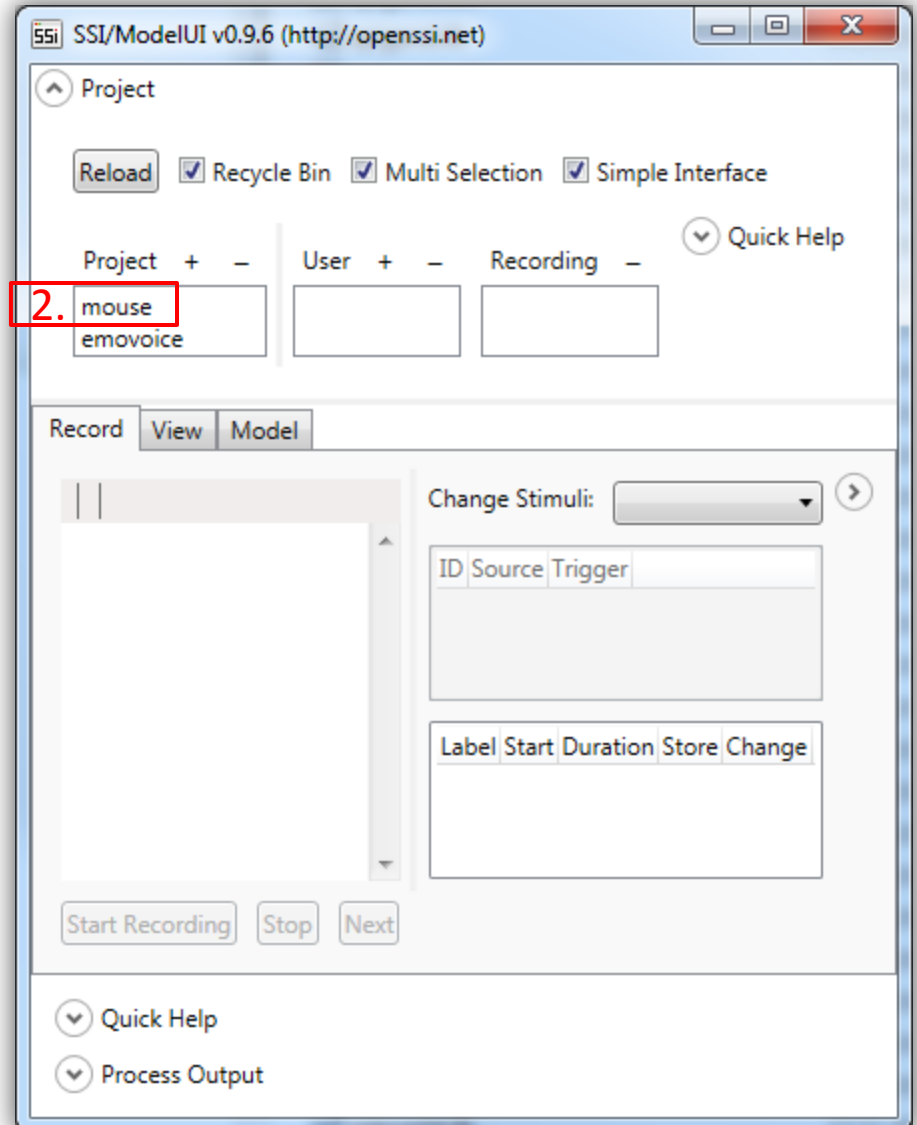
# File Consumer

1. Add a **FileWriter** (consumer) and connect it to **pos**
2. Set frame size (e.g. **10**)
3. Set a **path** and change file **mode** to **1** (=ASCII)
4. When you run the pipeline cursor position will be stored to disk (two files **cursor.stream** and **cursor.stream~** will be created)



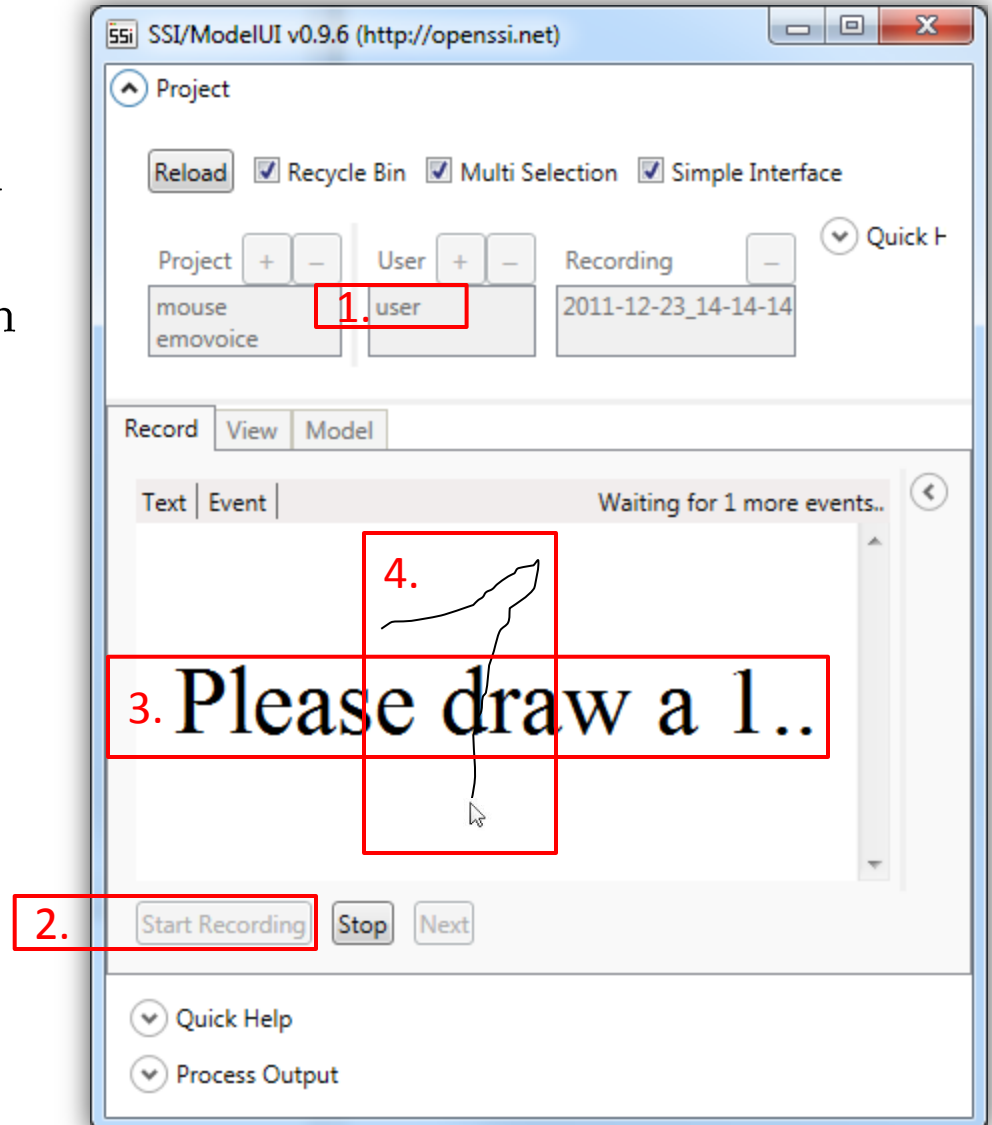
# ModelUI

1. Start [modelui.exe](#)
2. select project [mouse](#)



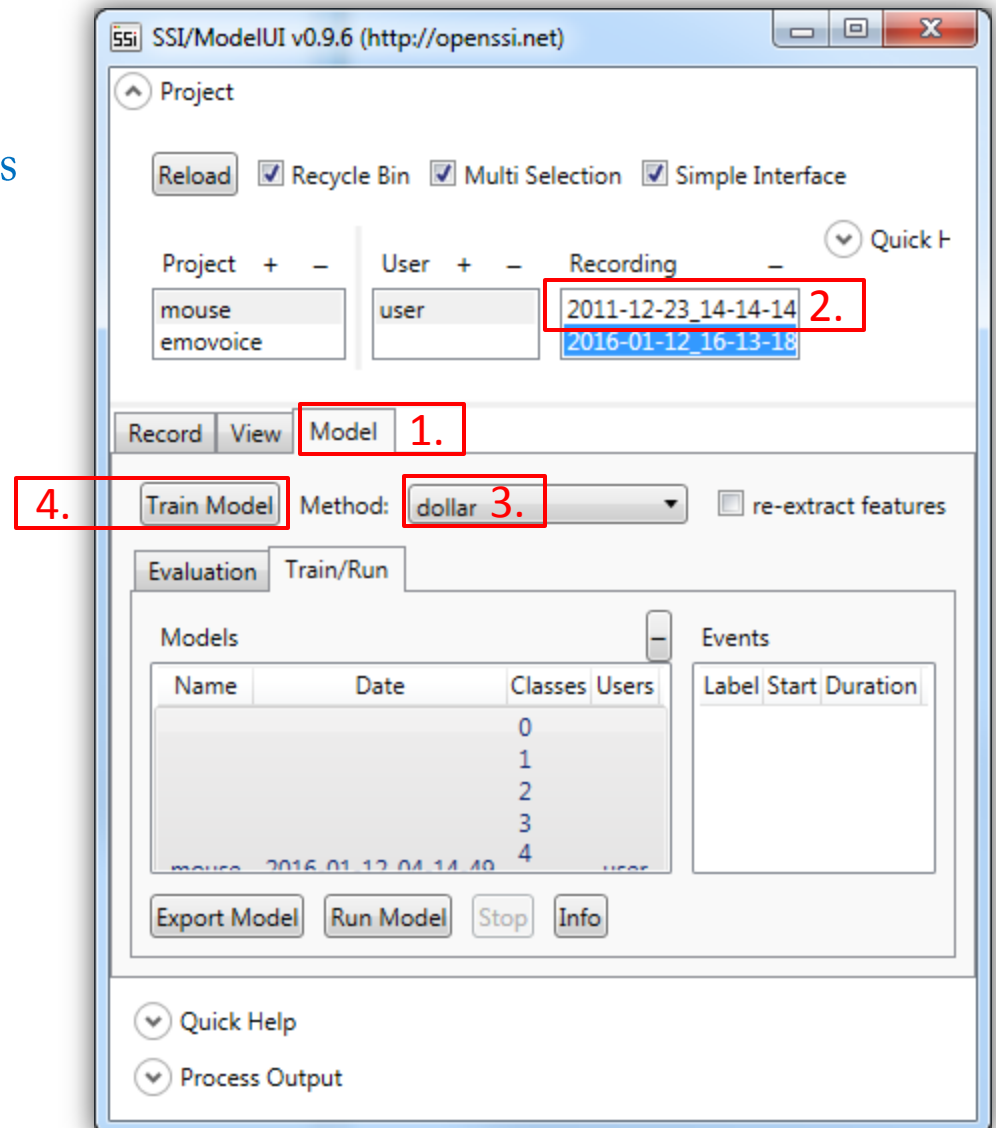
# Record

1. Select **user**
2. Click **Start Recording**
3. Follow instructions on screen
4. Draw inside the GUI while holding the left mouse button pressed



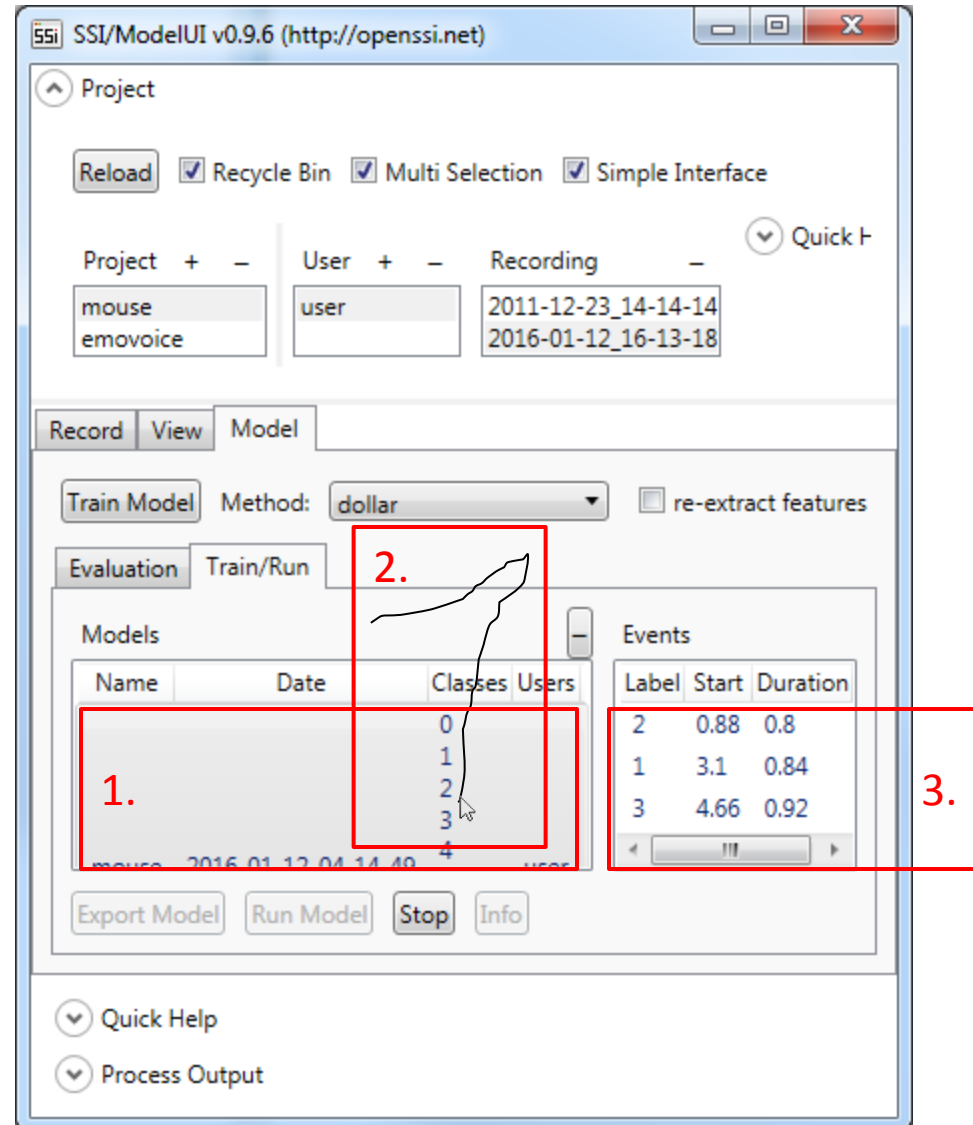
# Train

1. Switch to **Model** panel and select **Train/Run**
2. Select one or more **recordings** (by holding the ctrl key)
3. Select **dollar** as training method
4. Press **Train Model** to train model



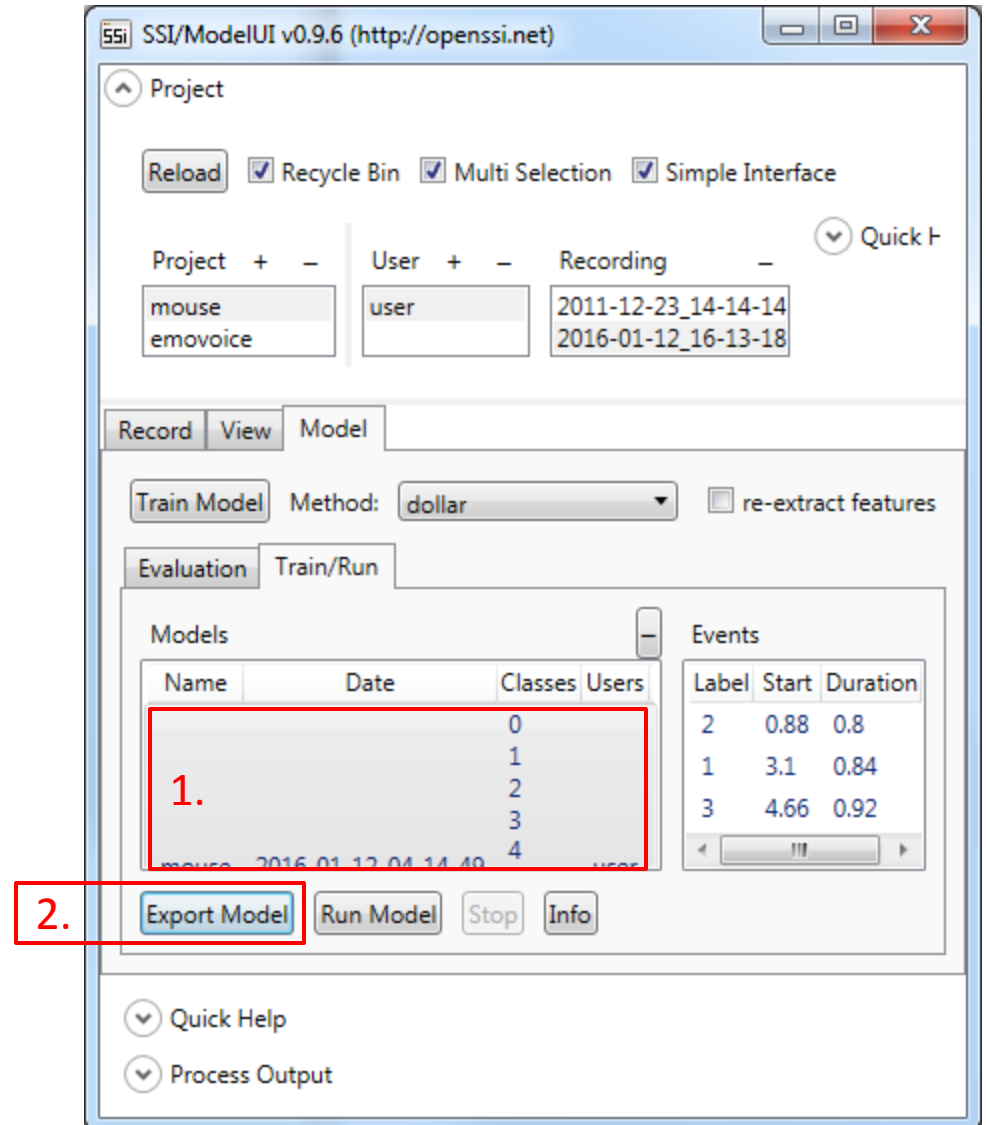
# Run

1. Select the trained **model** and click **Run Selected Model**
2. Draw inside the GUI by holding the left mouse button pressed
3. Classification results are displayed when you release the mouse button



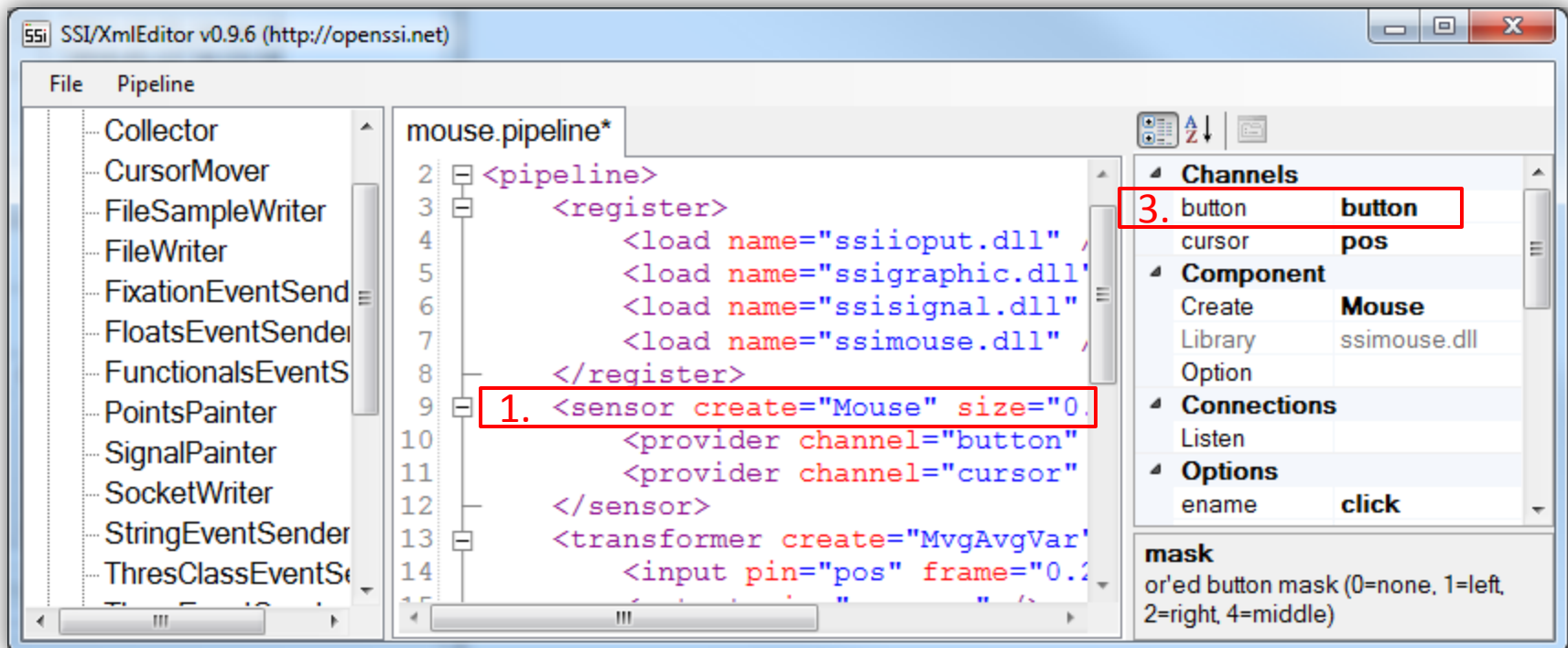
# Export

1. Select the trained model
2. Click **Export Model**
3. Save model in the same folder as the pipeline you have created earlier



# Button Provider

1. Switch back to the editor and place cursor again in the `<sensor...` line
2. Set `mask` of mouse sensor to `1`
3. Assign a name to the `button` channel name (e.g. `button`)



# Button Trigger

1. Add a **ZeroEventSender** (consumer) with input pin **button** and
2. Set frame size to **0.2s**
3. Input an event name by setting option **ename** (e.g. **gesture**) and set option **mindur** to **0.3**

The screenshot shows the SSI/XmlEditor v0.9.6 interface. On the left, a tree view lists various components, with **1. ZeroEventSender** highlighted in a red box. The main editor displays the XML configuration for `mouse.pipeline*`, showing several consumer definitions. The last consumer is `ZeroEventSender` with the following XML snippet:

```
<input pin="button" frame="0.2s" />
<consumer>
```

On the right, the configuration panel for the selected consumer is shown. It includes a table of options:

Option	Value
Delta (s for Event(e1..	
2. Frame (s for Pin	<b>0.2s</b>
Listen	
<b>Options</b>	
eager	False
empty	True
3. ename	<b>gesture</b>
hangin	0
hangout	0

Below the table, the **ename** field is described as "name of event (if sent to event board)".



# Classifier

1. Insert a consumer of type **Classifier** and set input to **pos**
2. Instead of a frame size set **gesture@** as **Event** name
3. Set **trainer** to the previously trained model (**mouse**) and **console** to **True**
4. Detected gestures will now be displayed on the console when you run the pipeline

The screenshot shows the SSI/XML Editor v0.9.6 interface. On the left, a list of consumers is shown, with **1. Classifier** highlighted by a red box. The main area displays the XML code for a pipeline named **mouse.pipeline**. The code includes several consumer definitions, with the last one being **Classifier**. On the right, the configuration for the **Classifier** consumer is shown. It includes the **Event (e1.. gesture@)** name, the **Pin** set to **pos**, and the **Options** section where **console** is set to **True** and **trainer** is set to **mouse**. These three items are highlighted with red boxes and numbered 1, 2, and 3 respectively. The **console** option is also described as **output classification to console**.

```
<?xml version="1.0" encoding="UTF-8" ?>
<pipeline name="mouse.pipeline">
  <consumer create="SignalPainter"
    <input pin="pos" frame="0.2" />
  </consumer>
  <consumer create="SignalPainter"
    <input pin="pos_avg" frame="0.2" />
  </consumer>
  <consumer create="FileWriter"
    <input pin="pos" frame="10" />
  </consumer>
  <consumer create="ZeroEventSender"
    <input pin="button" frame="1" />
  </consumer>
  <consumer create="Classifier"
    <input pin="pos" listen="gesture@" />
  </consumer>
</pipeline>
```

<b>2. Event (e1.. gesture@)</b>	
Frame (s f)	
Pin	<b>pos</b>
Listen	
<b>Options</b>	
<b>console</b>	<b>True</b>
ename	
merge	<b>False</b>
pthres	<b>0</b>
sname	
<b>3. trainer</b>	<b>mouse</b>

**console**  
output classification to console