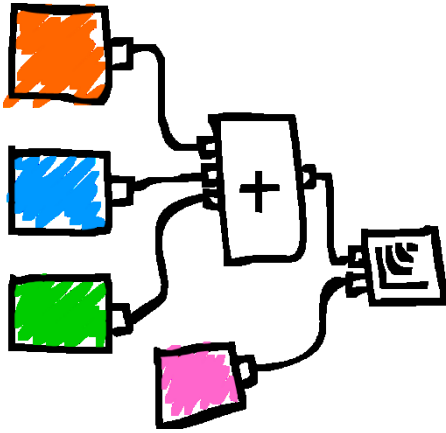


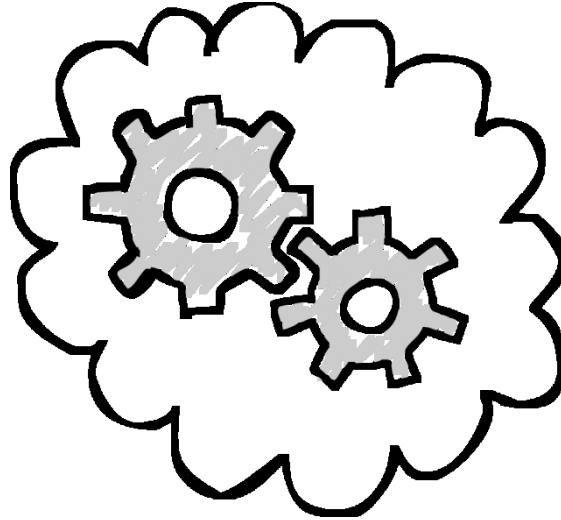
# ClowdFlows

Janez Kranjc

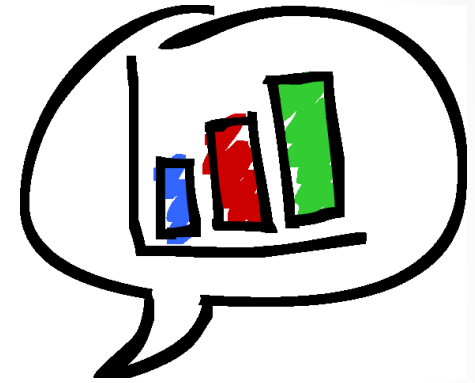
# What is CloudFlows



**CONSTRUCT**  
a workflow in  
the browser



**EXECUTE**  
on the cloud



**SHARE**  
your experiments  
and results 😊

# What is ClowdFlows

The screenshot displays the ClowdFlows workflow editor interface. The browser address bar shows `clowdflows.org/workflows/`. A yellow banner at the top contains the text: "Hello! Welcome to ClowdFlows. Start by clicking on widgets in the treeview on the left side!".

**Left Panel (Widget Treeview):**

- Local services
  - Bio3graph
  - Decision Support
  - Files
  - ILP
  - Integers
  - MySQL
  - NLP
  - Noise Handling
  - Objects
  - Orange
  - Performance Evaluation
  - ScikitAlgorithms
  - Streaming
  - Streaming Visualizations
    - Add neutral zone
    - Filter tweets by language
    - Remove words from tweets
    - RSS Reader
    - Simulate stream from Gamasystem csv
    - Simulate stream from text file
    - Sliding Window
    - Split positive and negative tweets
    - Tweet Sentiment Analysis
    - Twitter
  - Strings
  - Testing
  - Visual performance evaluation (ViperCharts)
  - Weka
- Subprocess widgets
- WSDL Imports
- Import webservice

**Main Canvas (Workflow Diagram):**

The workflow is titled "Snowden sentiment analysis" and consists of the following steps:

- Twitter** (Widget: Itw)
- Filter tweets by language** (Widget: Itw)
- Tweet Sentiment Analysis** (Widget: Itw)
- Sliding Window** (Widget: Ist)
- Sliding Window** (Widget: Ist)
- Sliding Window** (Widget: Ist)
- Split positive and negative tweets** (Widget: Itw, with sub-widgets ptw and ntw)
- Display tweets** (Widget: Itw)
- Positive tweets** (Widget: Itw)
- Positive Word Cloud** (Widget: Itw)
- Negative tweets** (Widget: Itw)
- Negative word cloud** (Widget: Itw)
- Sentiment graph** (Widget: Itw)

Each widget in the diagram has a green checkmark below it, indicating it is active or successful.

**Bottom Console:**

Welcome to ClowdFlows. This is the console where success and error messages are logged.

# What is ClowdFlows

- A platform for:
  - composition,
  - execution,
  - and sharing of **interactive data mining workflows**
- Most important features:
  - A web based user interface for building workflows
  - Cloud-based architecture, service-oriented architecture
  - Big roster of workflow components
  - Real-time processing module

# Building scientific workflows

- visual programming paradigm
- implemented in

- Weka,

Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. 3. edn. Morgan Kaufmann, Amsterdam (2011)

- Orange,

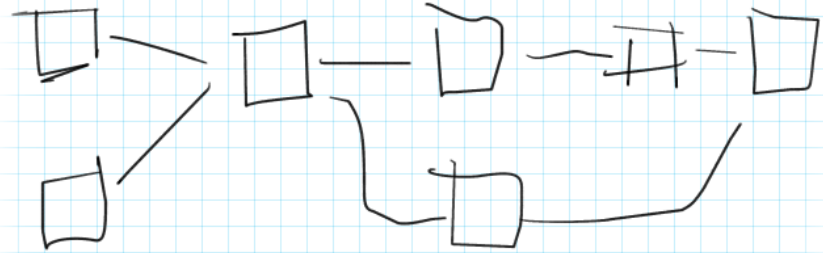
Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From experimental machine learning to interactive data mining. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: PKDD. Volume 3202 of Lecture Notes in Computer Science., Springer (2004) 537-539

- KNIME,

Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R., eds.: GfKI. Studies in Classification, Data Analysis, and Knowledge Organization, Springer (2007) 319-326

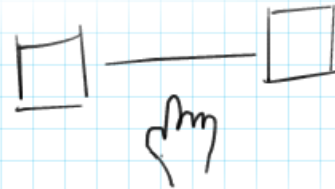
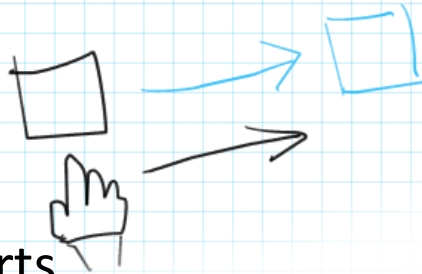
- RapidMiner

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In Ungar, L., Craven, M., Gunopulos, D., Eliassi-Rad, T., eds.: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (August 2006) 935-940



# Building scientific workflows

- consists of simple operations on workflow elements
  - drag
  - drop
  - connect
- suitable for non-experts
- good for representing complex procedures



# Distributed processing

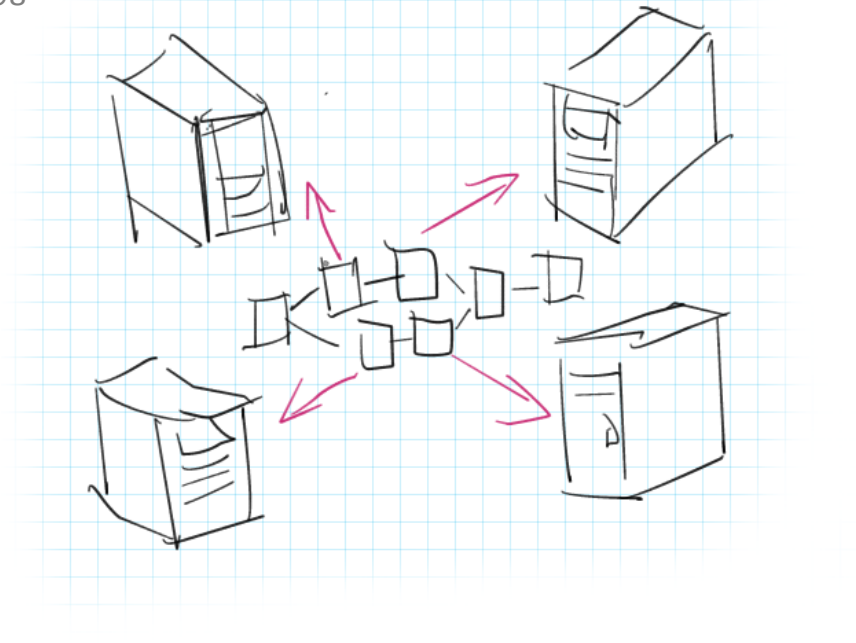
- Using Web Services

- like Taverna

Hull, D., Wolstencroft, K., Stevens, R., Goble, C.A., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34(Web-Server-Issue) (2006) 729-732

- and Orange4WS

Podpečan, V., Zemenova, M., Lavrač, N.: Orange4ws environment for service-oriented data mining. *The Computer Journal* 55(1) (2012) 89-98



# Distributed processing

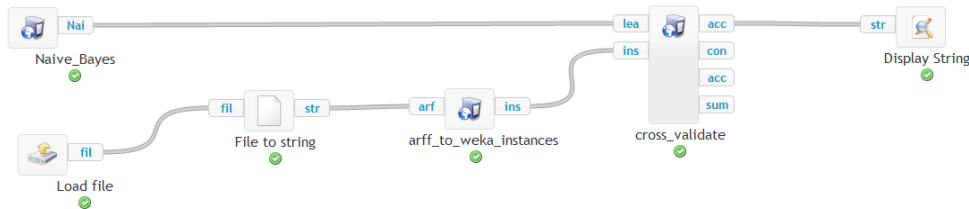
- Service oriented architecture
  - enables parallelization,
  - remote execution,
  - high availability,
  - provides access to large public (and proprietary) databases,
  - enables easy integration of 3rd party components
- T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.



# Sharing of workflows

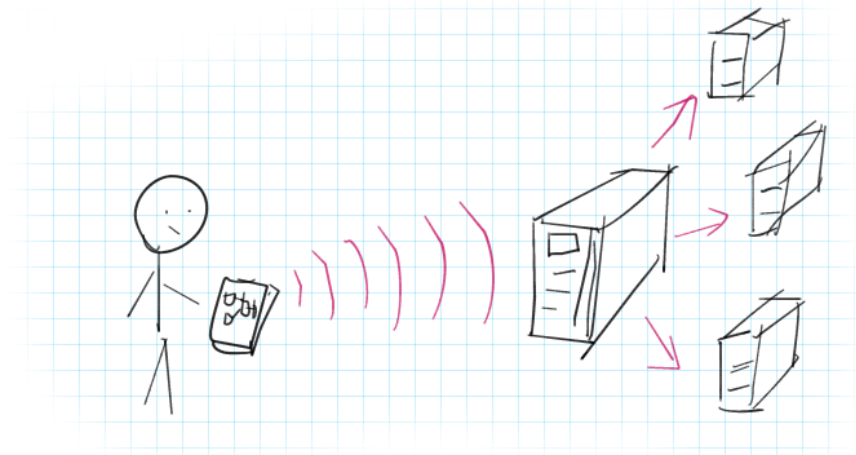
- Allow users to publicly upload their workflows so that they are available to a wider audience
- A link may be published in a research paper
- Like the *myExperiment* website

De Roure, D., Goble, C. and Stevens, R. (2009) The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. *Future Generation Computer Systems* 25, pp. 561-567



# Remote execution (cloud based)

- Executing workflows on different machines than used for construction
- Very useful for execution from mobile devices



# The user interface

The screenshot shows the CloudFlows workflow editor interface. The browser address bar displays `cloudflows.org/workflows/`. A yellow banner at the top contains the text: "Hello! Welcome to CloudFlows. Start by clicking on widgets in the treeview on the left side!".

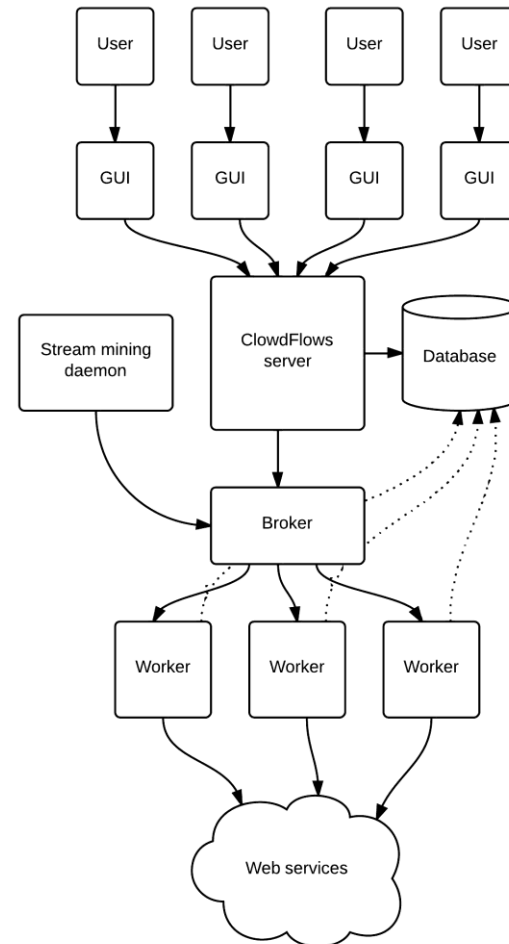
The interface is divided into three main sections:

- Widget Repository:** A treeview on the left side containing various categories of widgets such as "Local services", "Streaming", "Strings", and "Subprocess widgets". A red arrow points from the handwritten label "widget repository" to this treeview.
- Workflow Canvas:** The central workspace where a workflow is being built. The workflow is titled "Snowden sentiment analysis". It consists of several interconnected widgets: "Twitter", "Filter tweets by language", "Tweet Sentiment Analysis", "Sliding Window", "Display tweets", "Split positive and negative tweets", "Positive tweets", "Positive Word Cloud", "Negative tweets", and "Negative word cloud". A red arrow points from the handwritten label "workflow canvas" to this central area.
- Widget:** A specific widget being selected or highlighted in the workflow canvas, indicated by a red circle and a red arrow pointing from the handwritten label "widget".

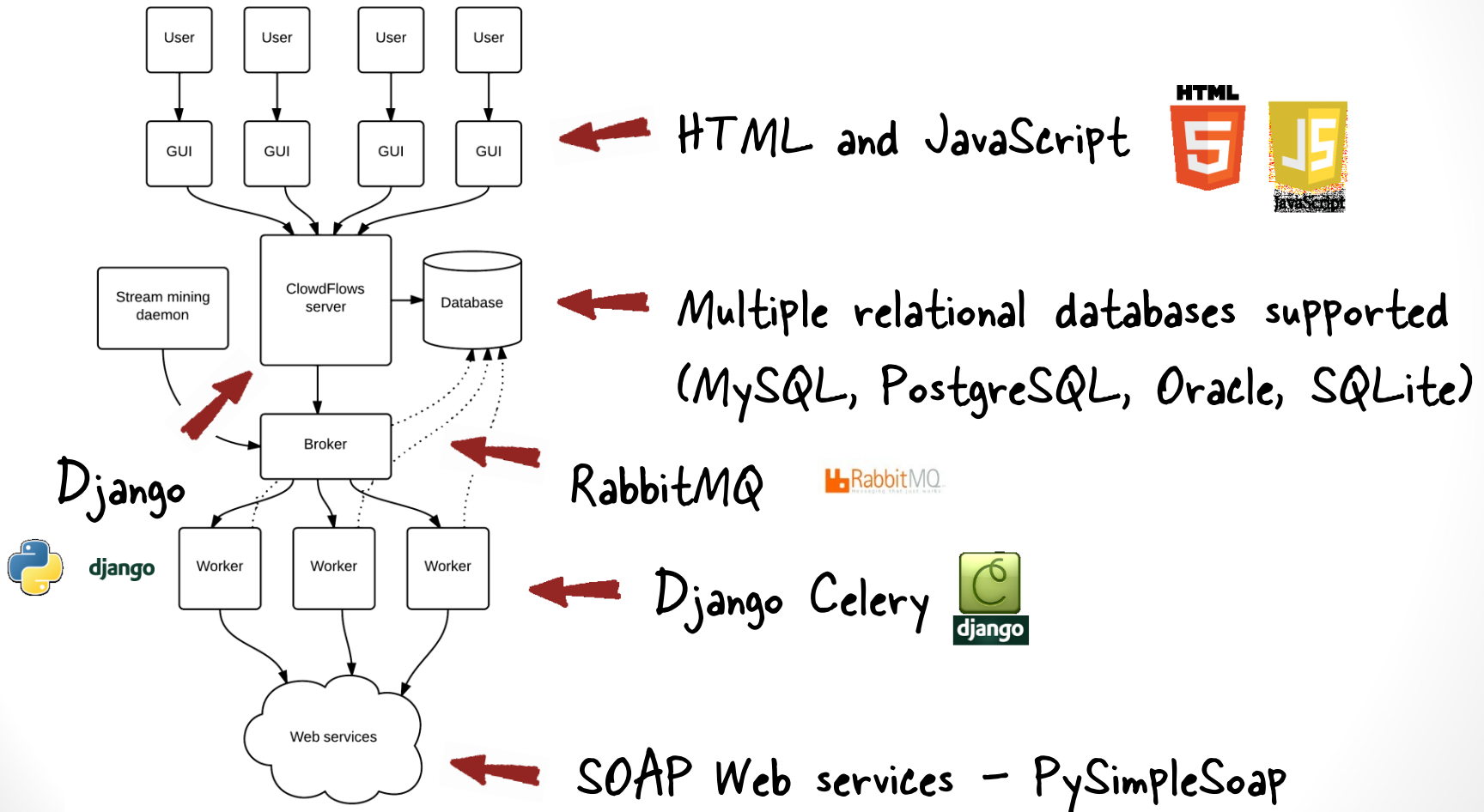
At the bottom of the interface, a console area displays the message: "Welcome to CloudFlows. This is the console where success and error messages are logged."

# The architecture

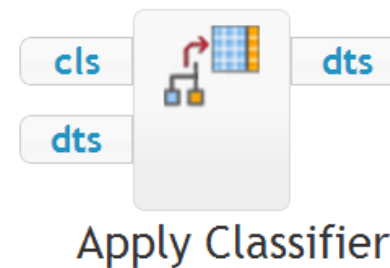
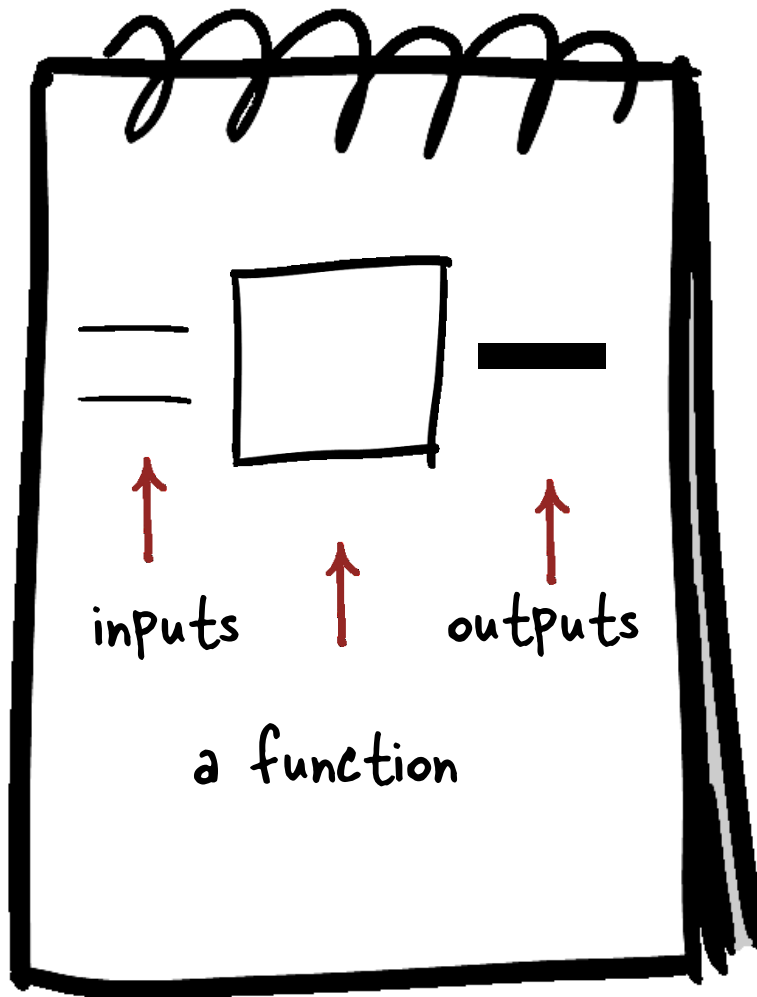
- GUI
  - User constructs workflows by connecting widgets on the canvas
- CloudFlows server
  - Serves the GUI, stores all changes to the database, emits tasks to execute widgets to the broker
- The broker
  - Delegates the tasks to workers.
- The workers
  - Headless instances of the CloudFlows server (they do not serve the user interface)
- Web services
  - Widgets may also be created by importing SOAP web services



# Technologies used



# The widget



```
def scikitAlgorithms_applyClassifier(input_dict):  
    classifier = input_dict['classifier']  
    data = input_dict['data']  
    data["target"] = classifier.predict(data["data"])  
    new_data = (data["data"], classifier.predict(data["data"]))  
    output_dict = {'classes':data}  
    return output_dict
```

# Types of widgets

- Regular widgets
- Visualization widgets
- Interactive widgets
- Special workflow control widgets

# Regular widgets

- Each regular widget is implemented as a Python function that transforms the inputs and parameters into outputs
- Widgets that implement complex procedures can also implement progress bars to notify the user of its progress.



# Visualization widgets

- Extended versions of regular widgets
- Visualization widgets also return HTML and JavaScript that is rendered in the user's browser
- Visualization widgets are regular widgets with the addition of a Python function which control the rendering of a template.

# Example visualization widget

The screenshot displays the Orange3 data mining environment. On the left is a widget palette with categories like Local services, Decision Support, Files, NLP, and Objects. The main workspace shows a workflow titled "Simple decision support workflow (copy)". The workflow consists of the following widgets: "Load Dataset to Orange Data Table" (ds), "Kepner-Tregoe" (odt), "Table viewer" (tab), "Sensitivity analysis" (mdl), and "Decision support visualization" (mdl). The "Decision support visualization" widget is selected, opening a dialog box titled "Decision support visualization visualization".

The dialog box shows the "Input model: Kepner-Tregoe" and has two tabs: "Weights pie chart" and "Weights bar chart". The "Weights bar chart" is active, displaying a horizontal bar chart titled "Weights". The x-axis is labeled "Weight" and ranges from 0 to 60. The y-axis lists three categories: "location", "long-term-prospects", and "salary". The bars have the following values: location (35), long-term-prospects (16), and salary (49).

Below the weights chart, there are two more tabs: "Alternatives column chart" and "Attribute values chart". The "Attribute values chart" is active, displaying a horizontal bar chart titled "Values". The x-axis is labeled "Value" and ranges from 0 to 0.6. The y-axis lists the same three categories: "location", "long-term-prospects", and "salary". The legend indicates four series: offerA (blue), offerB (red), offerC (green), and offerD (purple). The approximate values for each series are as follows:

| Category            | offerA | offerB | offerC | offerD |
|---------------------|--------|--------|--------|--------|
| location            | 0.15   | 0.30   | 0.45   | 0.20   |
| long-term-prospects | 0.48   | 0.05   | 0.28   | 0.18   |
| salary              | 0.05   | 0.45   | 0.30   | 0.20   |

The dialog box also includes a "Close" button at the bottom right.

# Interactive widgets

- Requires execution prior to prompting the user
- A widget can also be a combination of interactive and visualization widget

# Example interactive widget

The screenshot displays the Orange3 interface. On the left is a widget palette with categories: Ensemble, Object viewer, Pickle object, Unpickle object, Orange, Classification (including C4.5 Tree Learner, Classification Tree, CH2 Rule Learner, k-Nearest Neighbours, Logistic Regression, Lookup Learner, Majority Learner, Naive Bayes, Random Forest, Rule Induction, Support Vector Machine, Support Vector Machine Easy), Evaluation (Apply Classifier, Build Classifier), Utilities (Add Class Noise, Alter table, Load Dataset to Orange Data Table, Orange Data Table to ARFF file, Orange Data Table to ARFF String, Orange Data Table to CSV file, Orange Data Table to TAB file, Select Attributes, Select Data, Table viewer, UCI Dataset to Orange Data Table), Performance Evaluation (Aggregate Detection Results, Evaluate Detection Algorithms, Evaluate Repeated Detection, Evaluation Results to Table, Performance Chart, VIPER: Visual Performance Evaluation), and Strings.

The main workspace shows a workflow titled "Simple decision support workflow (copy)". It contains two widgets: "Load Dataset to Orange Data Table" (labeled "ds") and "Alter table" (labeled "tab"). The "Alter table" widget is currently active, displaying an "Alter table interaction" dialog box. This dialog shows a table with 4 entries, with the first entry selected. The table has columns: location, salary, long-term-prospects, and label [meta].

| location | salary | long-term-prospects | label [meta] |
|----------|--------|---------------------|--------------|
| 1.000    | 4.000  | 4.000               | offerD       |
| 4.000    | 1.000  | 10.000              | offerA       |
| 6.000    | 9.000  | 1.000               | offerB       |
| 9.000    | 6.000  | 6.000               | offerC       |

The dialog also includes a search bar, a "Show 10 entries" dropdown, and navigation buttons: "First", "Previous", "1", "Next", "Last". An "Apply" button is located at the bottom right of the dialog.

# Workflow control widgets

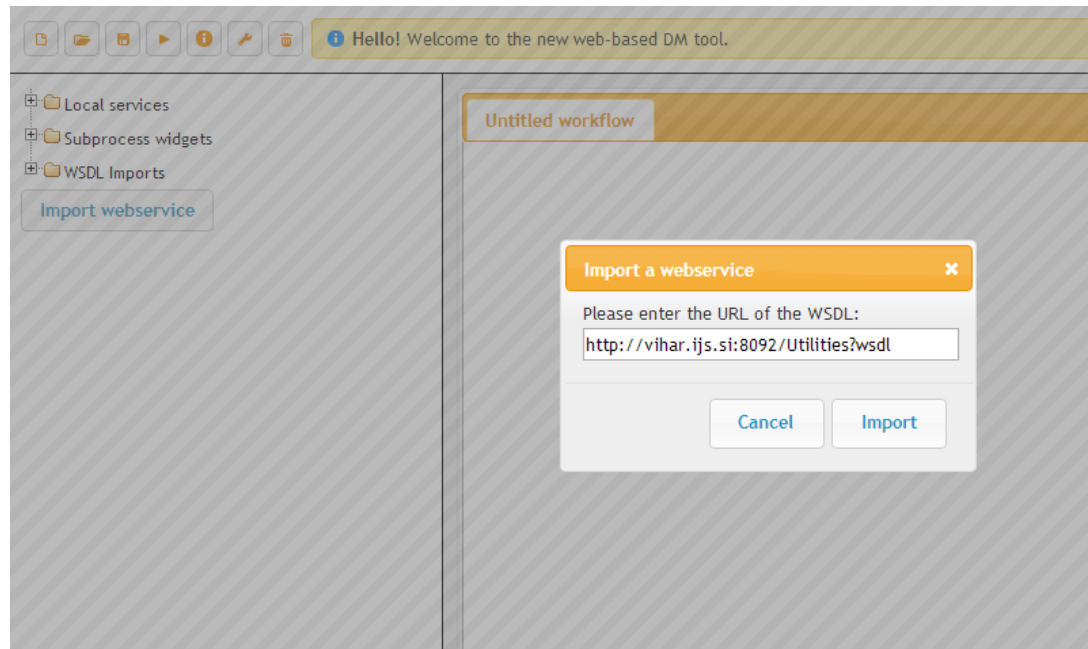
- Sub-workflow widget
- Input widget
- Output widget
- For loops (and cross validation)

# The workflow execution engine

- JavaScript engine
  - Useful for monitoring
- Python engine
  - faster

# Expanding the widget repository

- With Web services



# Expanding the widget repository

- With Web services

The screenshot displays a web-based workflow design tool. At the top, a yellow status bar reads "Hello! Welcome to the new web-based DM tool." Below this, a navigation pane on the left contains a tree view with "Local services", "Subprocess widgets", and "WSDL Imports", along with an "Import webservice" button. The main workspace shows an "Untitled workflow" canvas. Four dialog boxes are open, each for configuring an input designation:

- weka\_instances\_to\_arff input designation:** The "instances" field is set to "Input".
- print\_model input designation:** The "model" field is set to "Input".
- print\_tree input designation:** The "tree\_model" field is set to "Input".
- arff\_to\_weka\_instances input designation:** The "arff" field is set to "Input" and the "class\_index" field is also set to "Input".

Each dialog box includes "Close" and "Apply" buttons.



# Expanding the widget repository

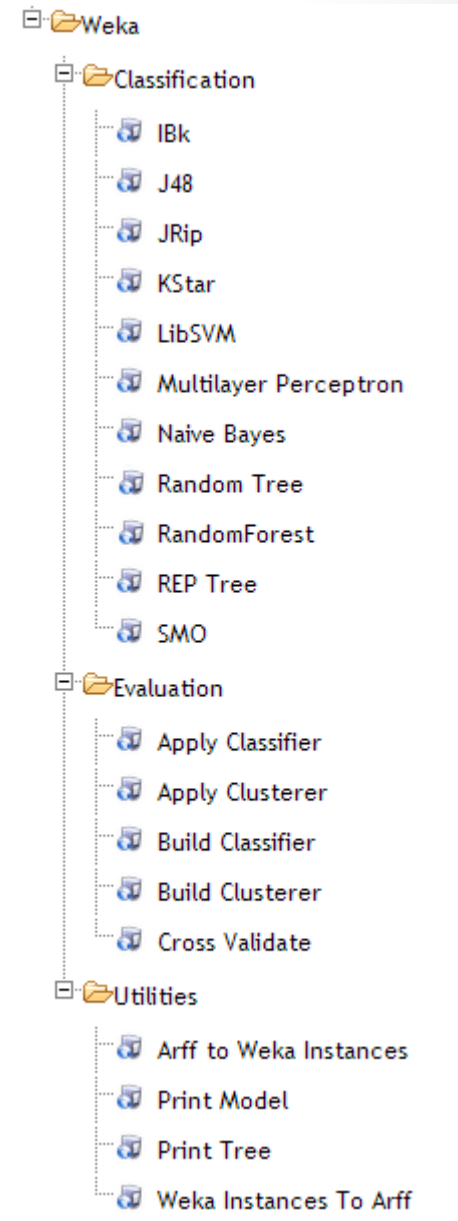
- By adding new Python functions directly to the source code
  - More powerful

# Packages

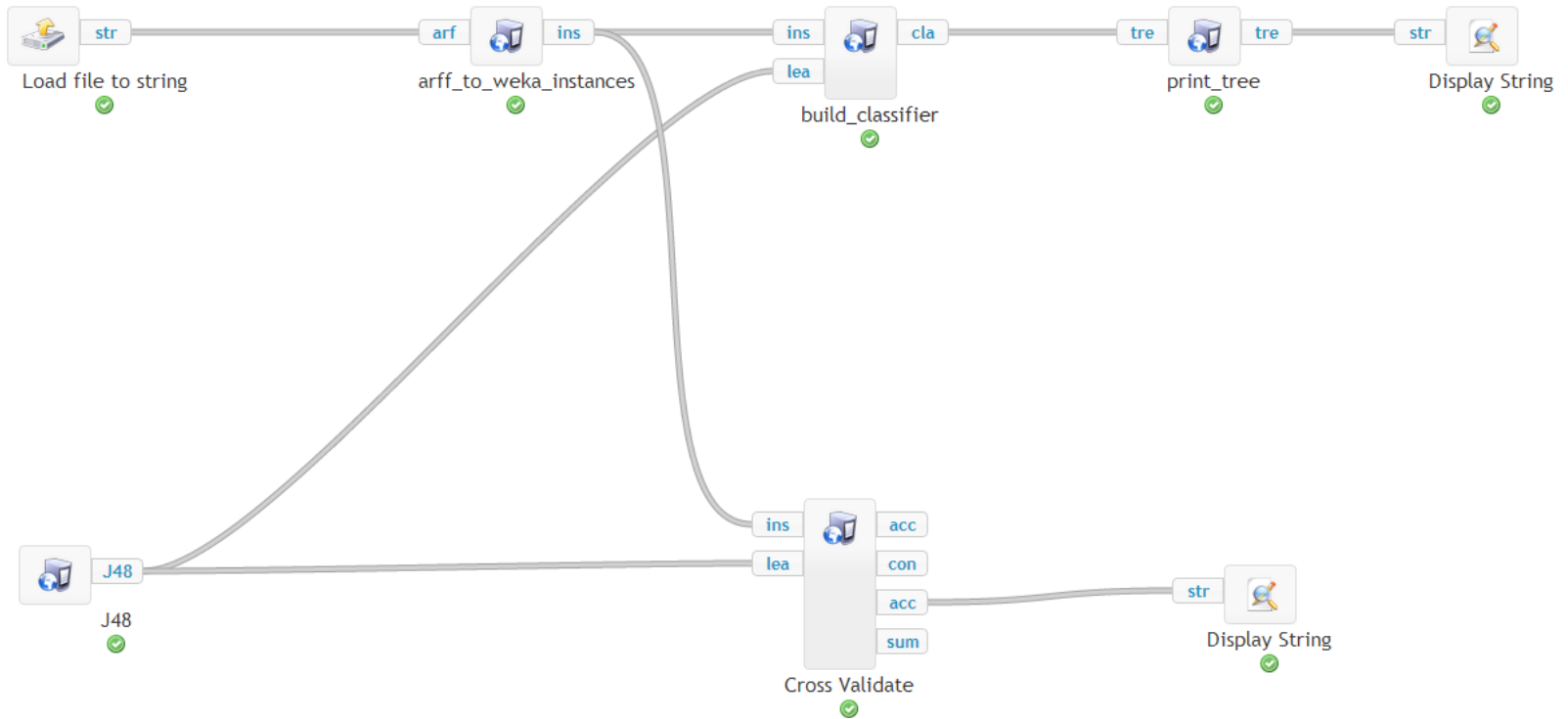
- Widgets are joined in packages which allows
  - Distributed development
  - Enabling/disabling widgets that are not useful to a particular user
- Packages currently include
  - Base package (basic data manipulation and preprocessing)
  - Scikit-learn package
  - Orange package (implementations of the Orange data mining tool algorithms)
  - Weka package (Weka algorithms exposed as webservices and imported in ClowdFlows)
  - ILP package
  - Text mining package
  - Natural language processing package
  - Performance evaluation and visualization
  - Stream mining package
  - ...

# Weka widgets

- Implemented as Web services

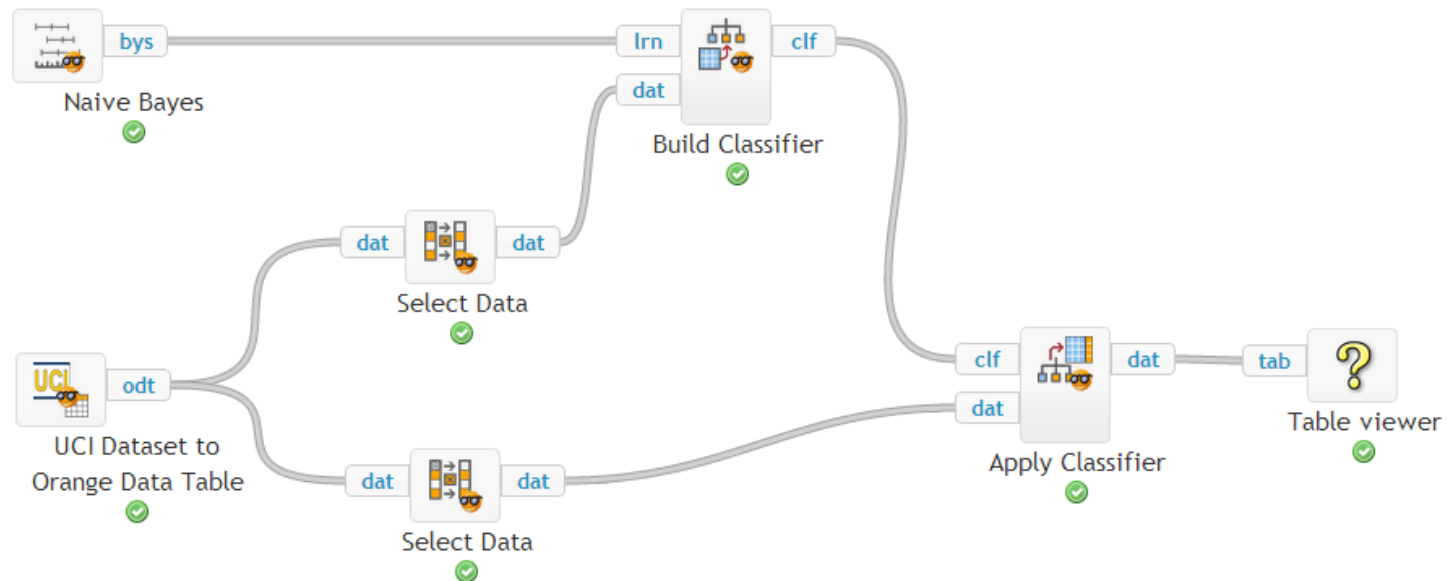


# Weka widgets



# Orange widgets

- Python functions wrapped in ClowdFlows widgets



# Decision support

- Python functions built from scratch

The screenshot displays the Orange3 interface with a workflow titled "Decision support (copy)". The workflow consists of the following widgets: "Load Dataset to Orange Data Table", "Alter table", "Kepner-Tregoe", "Table viewer", and "Sensitivity analysis".

The "Table viewer results" widget shows a table with 8 entries. The columns are: cena, oddaljenost, cajt, quality, kok\_se\_najes, score [meta], and label [meta].

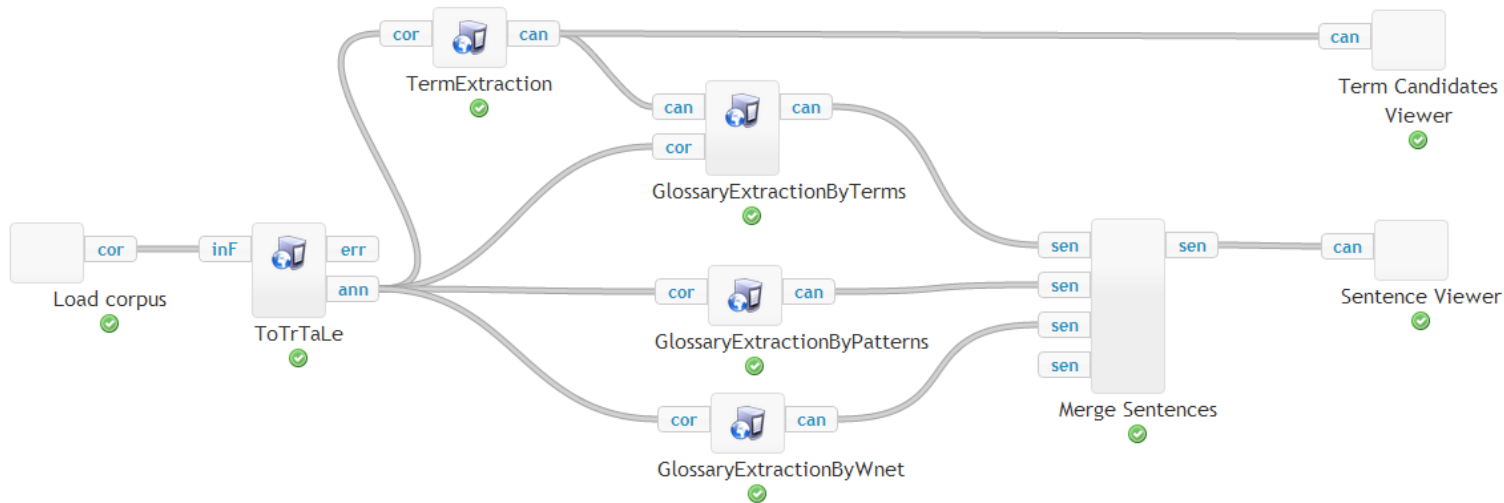
| cena  | oddaljenost | cajt  | quality | kok_se_najes | score [meta] | label [meta] |
|-------|-------------|-------|---------|--------------|--------------|--------------|
| 0.051 | 0.057       | 0.051 | 0.034   | 0.047        | 9.887        |              |
| 0.081 | 0.057       | 0.063 | 0.057   | 0.093        | 12.968       |              |
| 0.088 | 0.132       | 0.063 | 0.057   | 0.093        | 18.848       |              |
| 0.114 | 0.226       | 0.127 | 0.114   | 0.093        | 32.419       |              |
| 0.139 | 0.057       | 0.253 | 0.114   | 0.140        | 25.281       |              |
| 0.164 | 0.094       | 0.190 | 0.170   | 0.163        | 30.169       |              |
| 0.177 | 0.189       | 0.127 | 0.227   | 0.233        | 39.723       |              |
| 0.187 | 0.189       | 0.127 | 0.227   | 0.140        | 38.706       |              |

The "Decision support visualization visualization" widget shows a pie chart titled "Weights" for the input model "Kepner-Tregoe". The weights are:

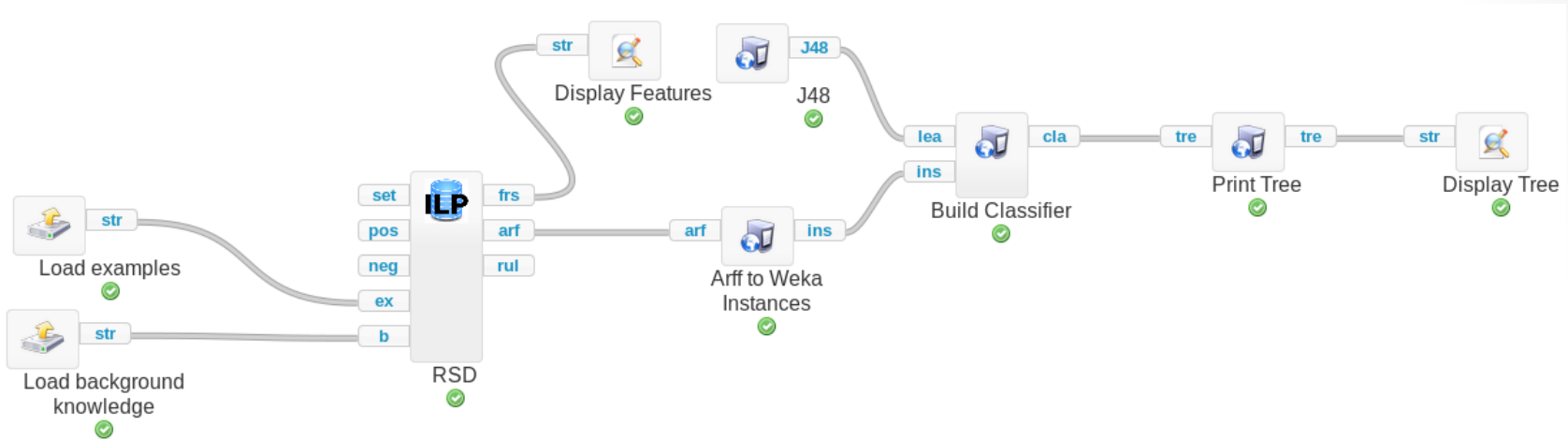
- cajt: 17.78846153846154 %
- kok\_se\_najes: 6.25 %
- quality: 30.288461538461537 %
- cena: 9.134615384615383 %
- oddaljenost: 36.538461 %

The "Sensitivity analysis visualization" widget shows a line chart titled "Sensitivity analysis" for the attribute "cajt". The chart plots "Score" (y-axis, 0 to 70) against "Weight of 'cajt'" (x-axis, 0 to 100). The legend includes: mizje, tazin, kitajc, fokulus, sparma, kebab, famel, and merick.

# Natural Language Processing



# ILP



**Display Features visualization** [Close]

```

f(1,A):-hasCar(A,B),carshape(B,u_shaped).
f(2,A):-hasCar(A,B),carshape(B,bucket).
f(3,A):-hasCar(A,B),carshape(B,hexagon).
f(4,A):-hasCar(A,B),carshape(B,ellipse).
f(5,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long).
f(6,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short).
f(7,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short),has_sides(B,double).
f(8,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short),has_sides(B,not_double).
f(9,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long),has_roof(B,flat).
f(10,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long),has_roof(B,none).
f(11,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short),has_roof(B,none).
f(12,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long),has_roof(B,jagged).
f(13,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short),has_roof(B,peaked).
f(14,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,short),has_roof(B,flat).
f(15,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long),has_wheels(B,3).
f(16,A):-hasCar(A,B),carshape(B,rectangle),carlength(B,long),has_wheels(B,2).
  
```

**Display Tree visualization** [Close]

```

J48 pruned tree
-----

f8 = +
|  f99 = +: east (10.0/1.0)
|  f99 = -: west (3.0/1.0)
f8 = -: west (7.0)

Number of Leaves :    3
Size of the tree :    5
  
```



# Real-time processing module

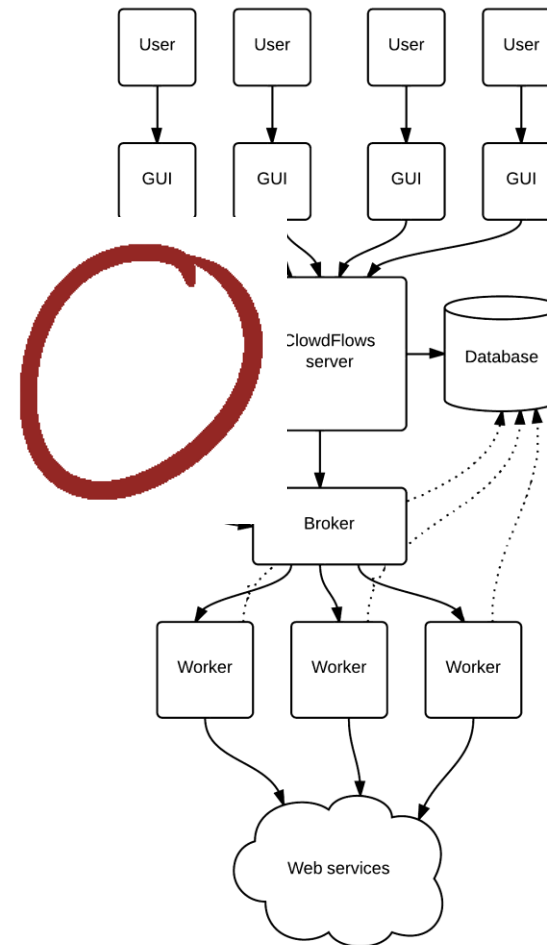
## Regular workflows and stream mining workflows

### Static Workflows

- The workflow is composed of several components
- Each component is executed a finite amount of times
- The results are available immediately after execution

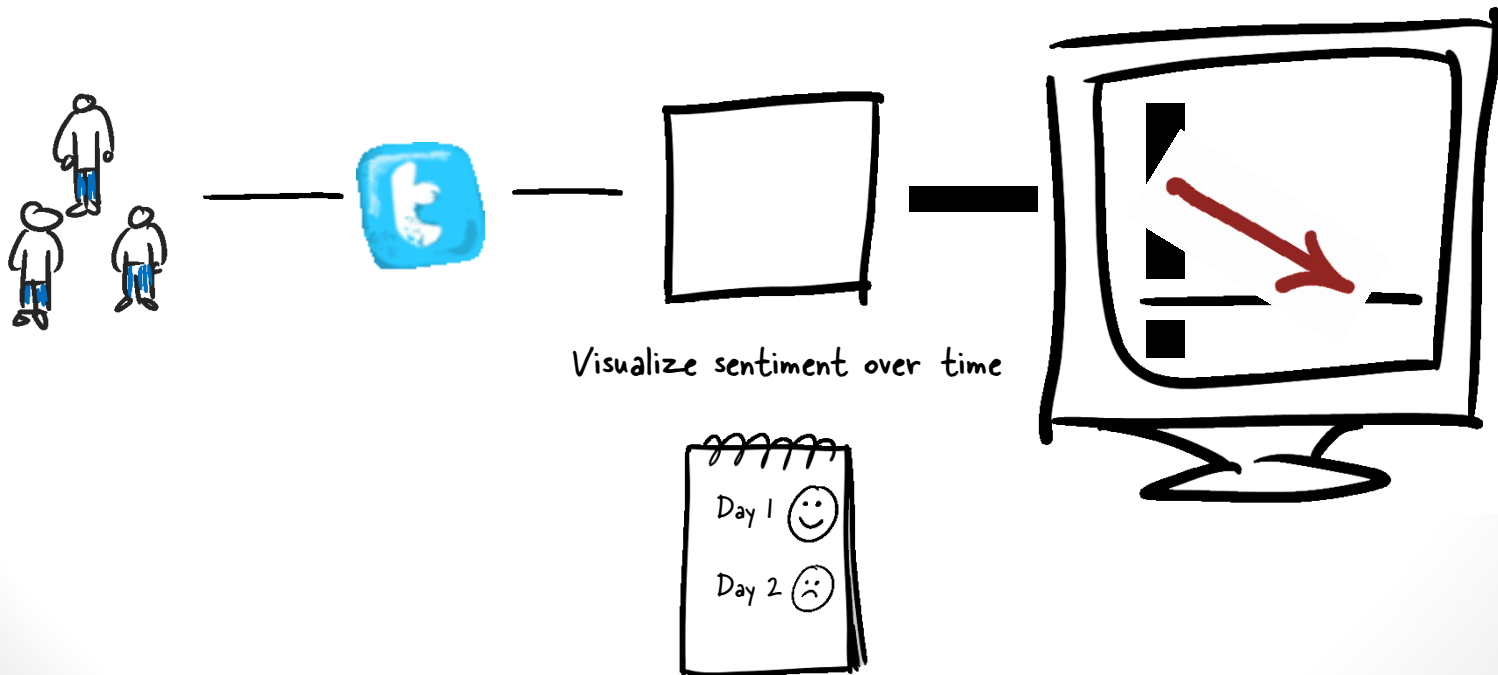
### Stream mining workflows

- The workflow is composed of several components
- It is not defined how many times each component will be executed
- The results are usually available after an initial delay

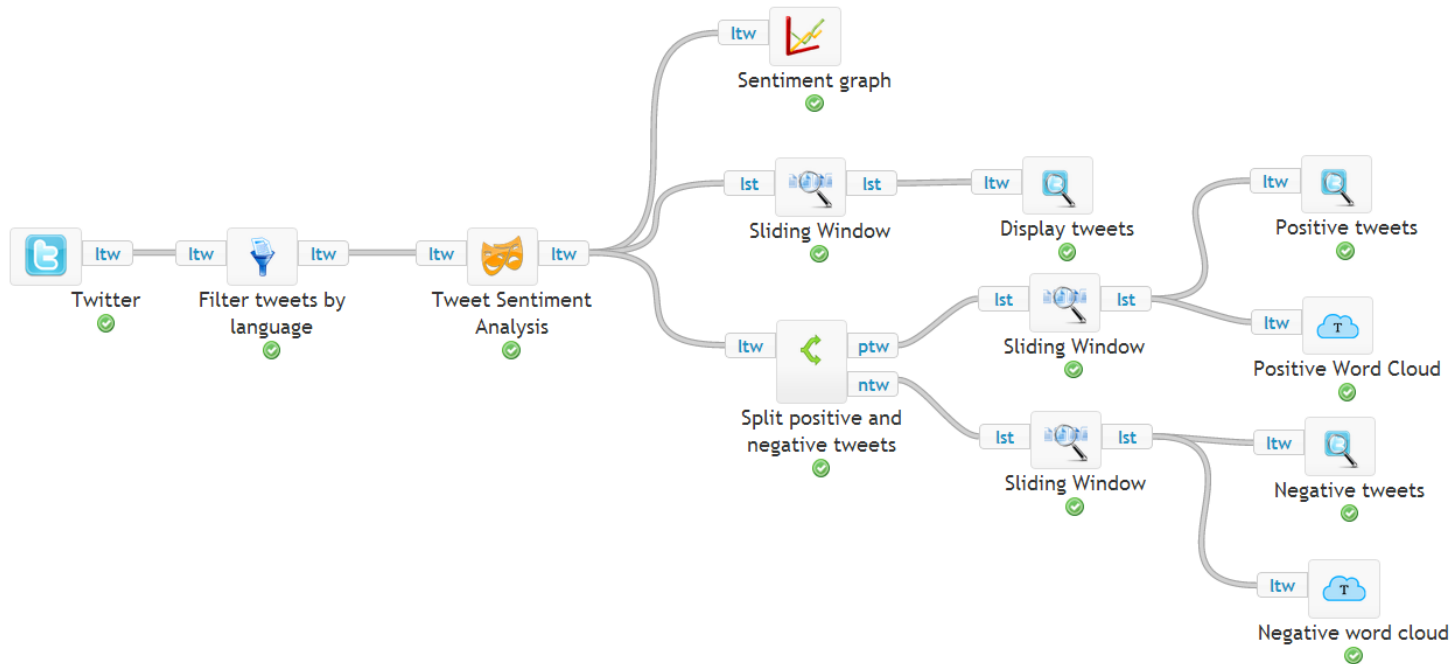


# Real-time processing module

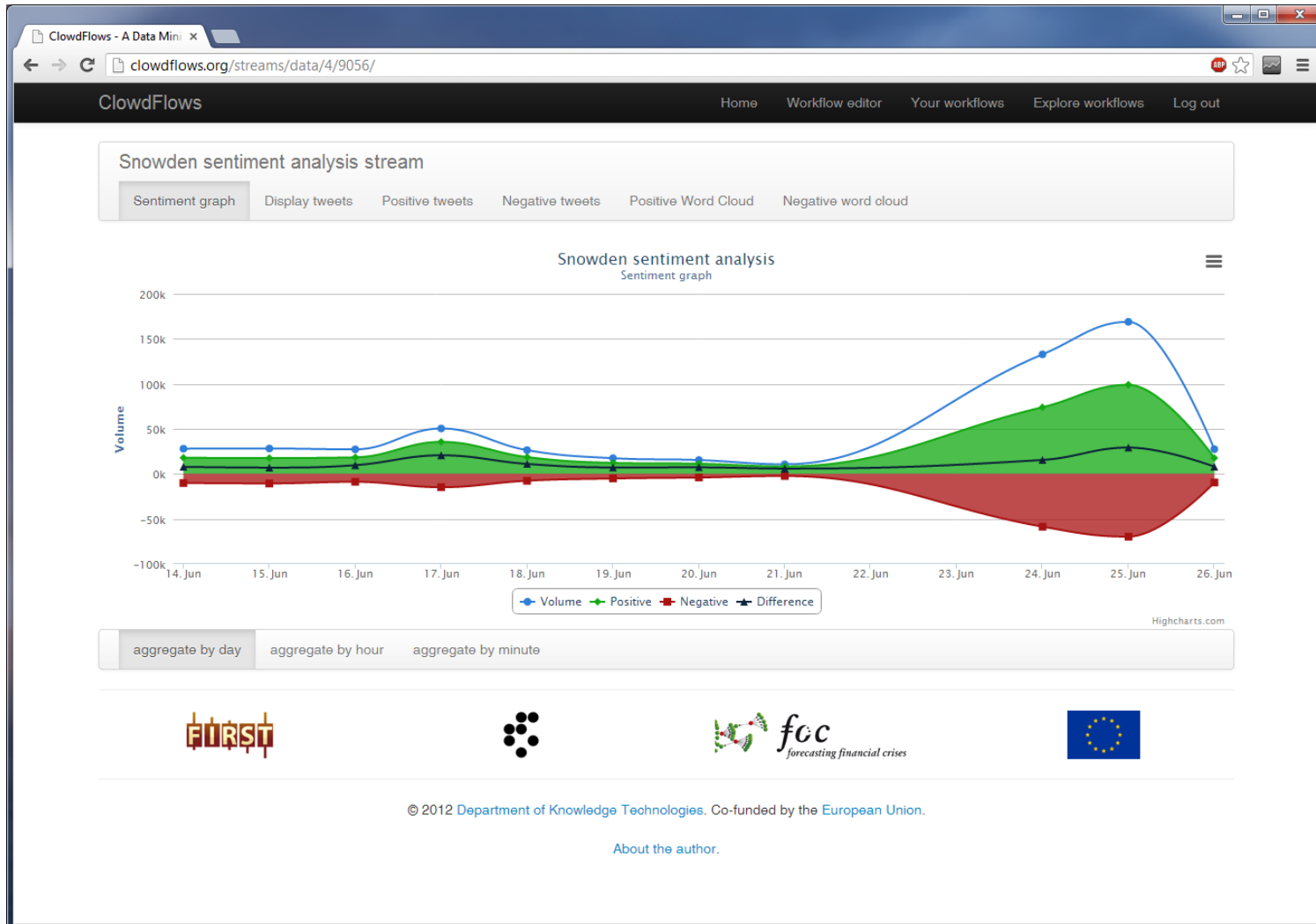
- In order to create streaming workflows we need widgets that are capable of handling streams
- Every stream mining workflow needs at least one **streaming widget**
  - Streaming widgets have additional persistent memory



# Sentiment Analysis Example



# Sentiment Analysis Example



# Sentiment Analysis Example



# Conclusion

- We have implemented an extensible cloud based platform for workflow construction and execution with real-time processing capabilities.
- ClowdFlows is available to use online at <http://clowdflows.org>
- Released open source under the MIT licence  
<https://github.com/xflows/clowdflows>