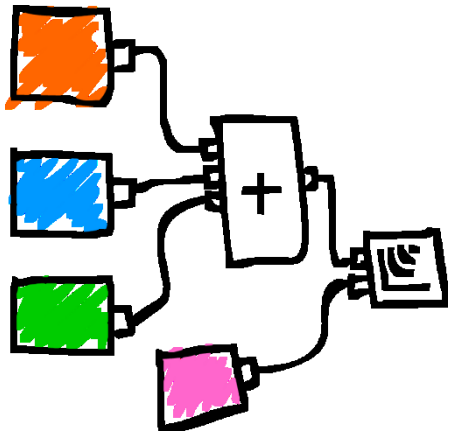
ClowdFlows

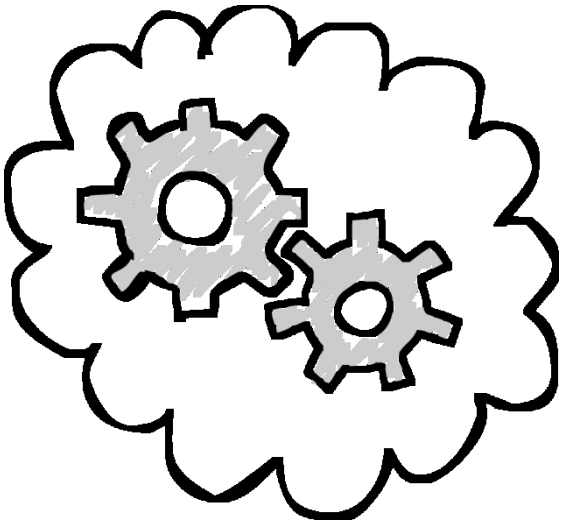
Janez Kranjc

What is ClowdFlows



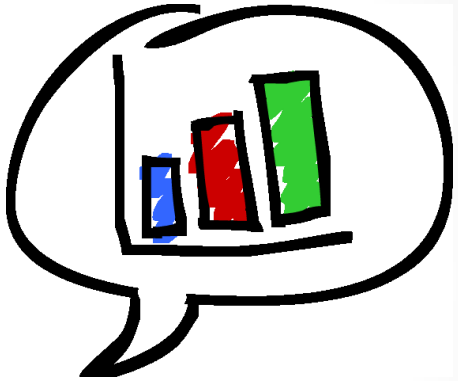
CONSTRUCT

a workflow in the browser



EXECUTE

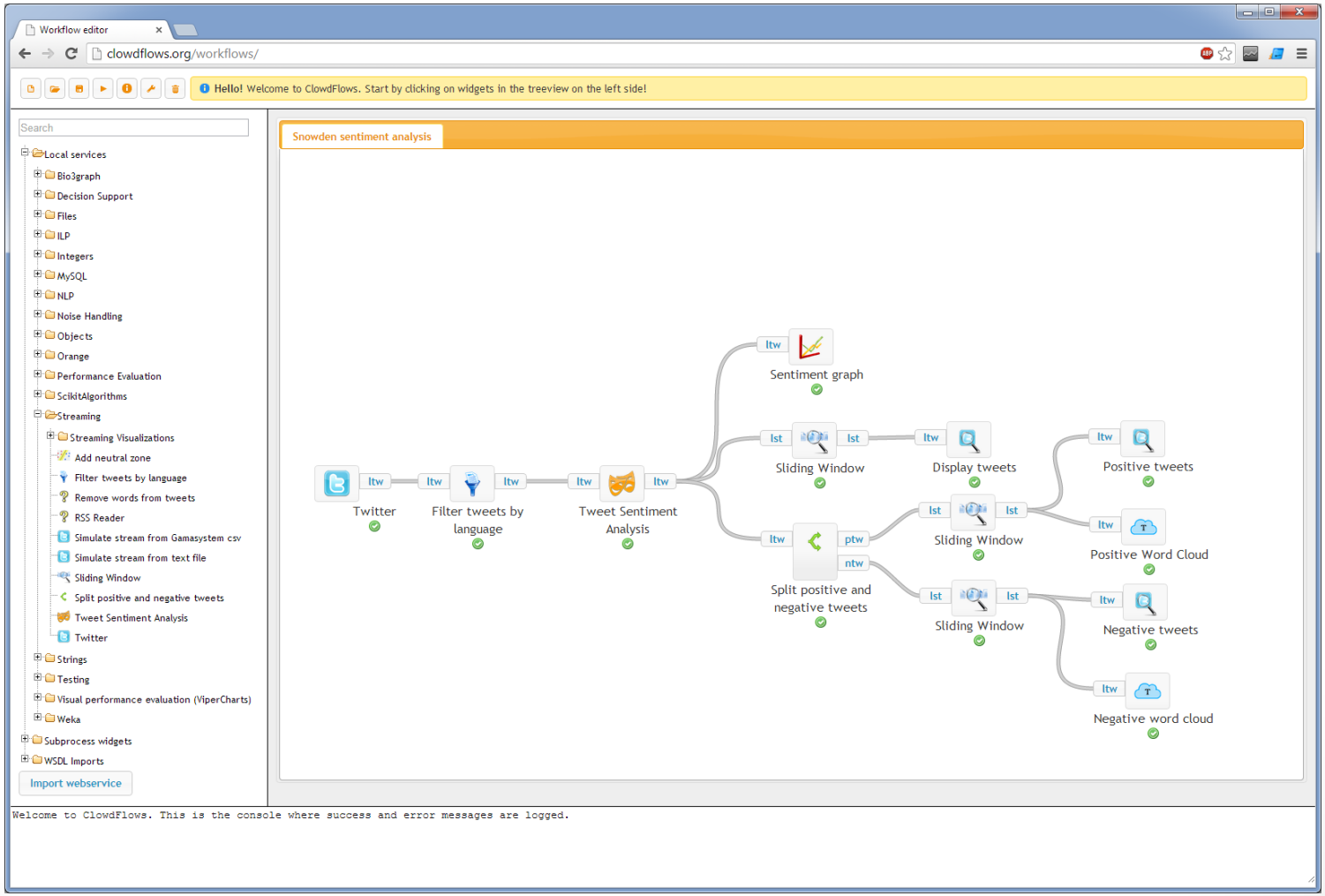
on the cloud



SHARE

your experiments and results ☺

What is ClowdFlows

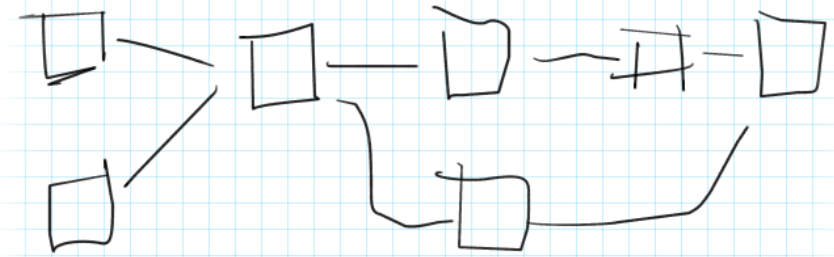


What is ClowdFlows

- 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

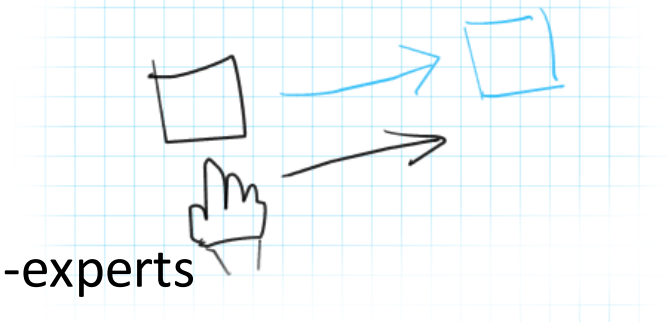
Building scientific workflows

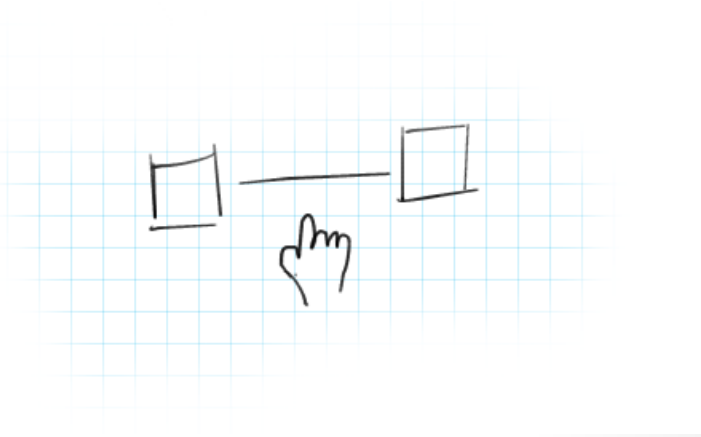


- visual programming paradigm
- implemented in
  - Weka,
    Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. 3. edn. Morgan Kaufmann, Amsterdam (2011)
  - Orange,
    Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From experimental machine learning 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.: GfKl. 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



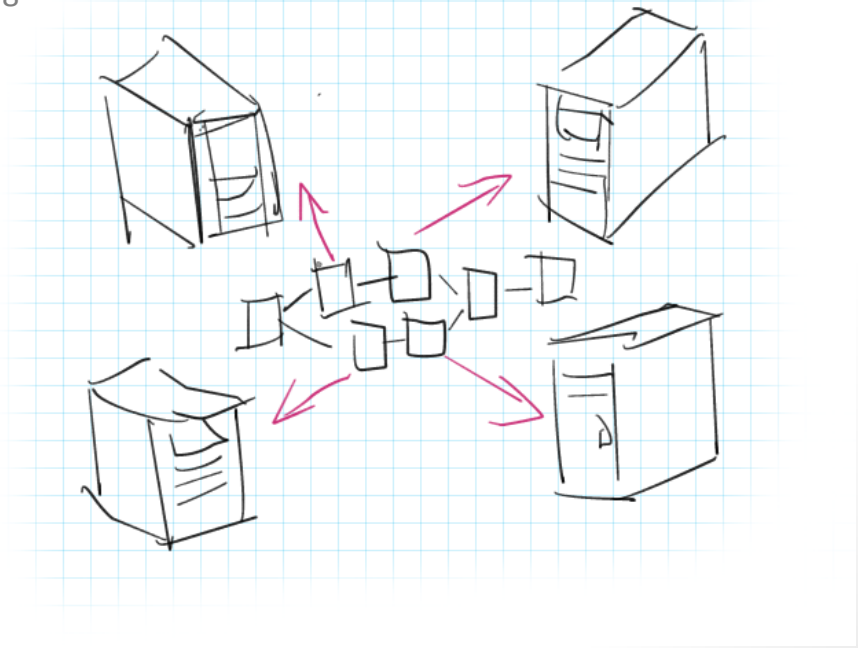


Distributed processing

- 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



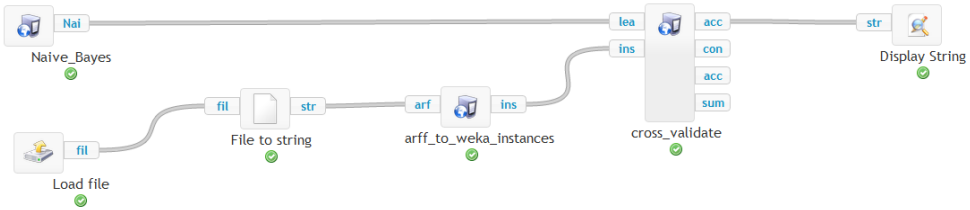
Distributed processing

- 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

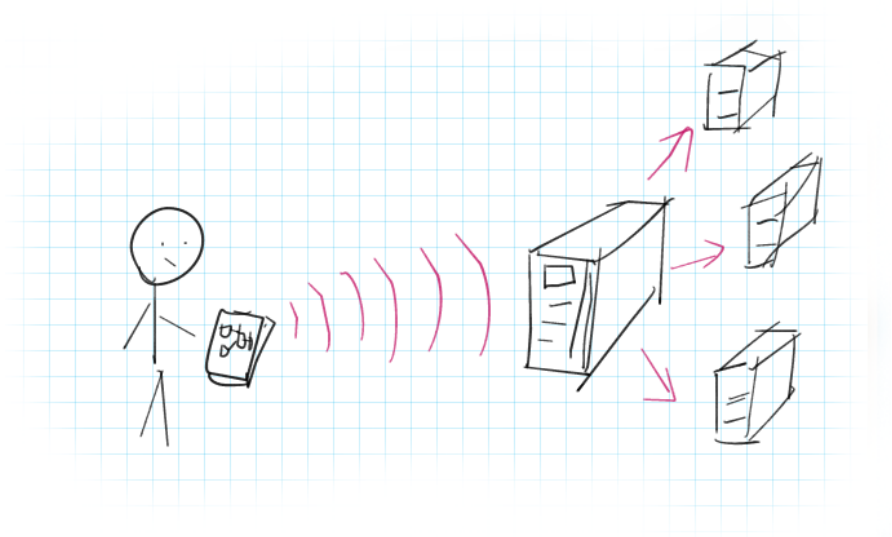
- Allow users to publicly upload their workflows so that they are available to a wider audience
- A link may be published in a research paper
- Like the *myExperiment* website

  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

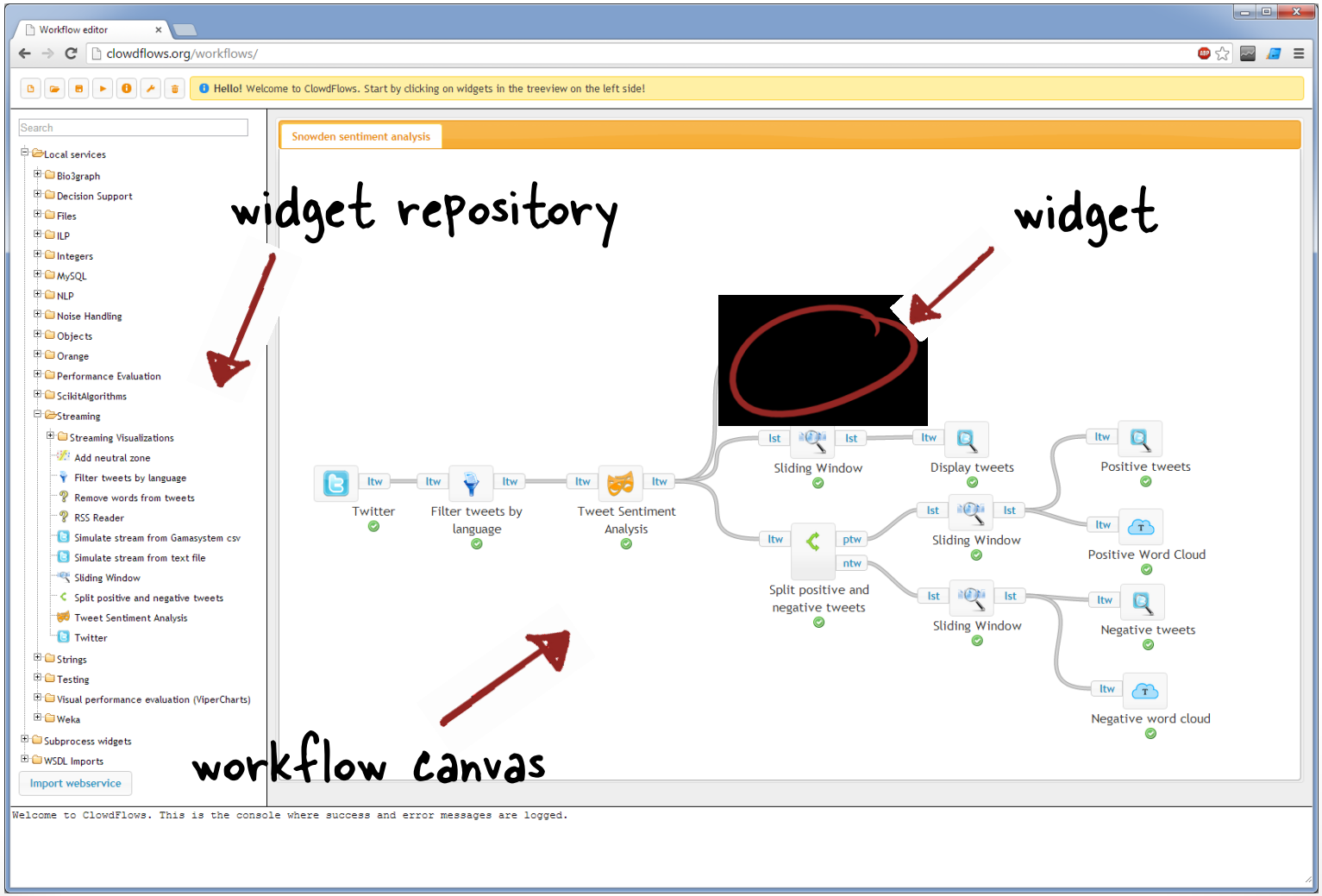


Remote execution (cloud based)

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

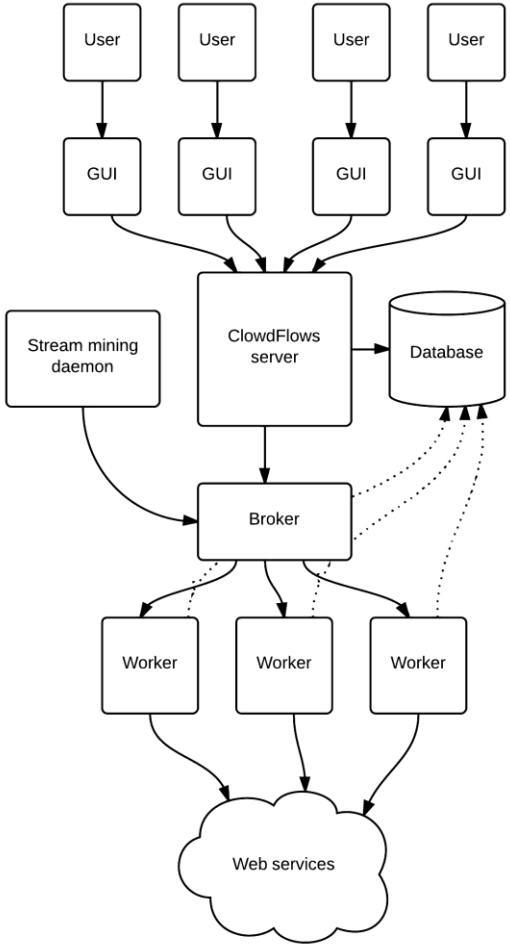


The user interface

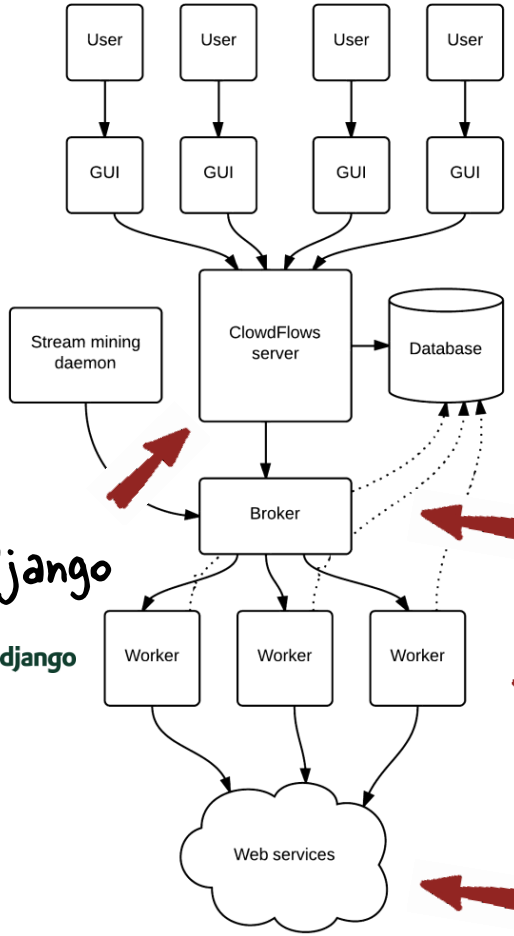


The architecture

- GUI
  - User constructs workflows by connecting widgets on the canvas
- ClowdFlows 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 ClowdFlows server (they do not serve the user interface)
- Web services
  - Widgets may also be created by importing SOAP web services



Technologies used



HTML and JavaScript

Multiple relational databases supported (MySQL, PostgreSQL, Oracle, SQLite)

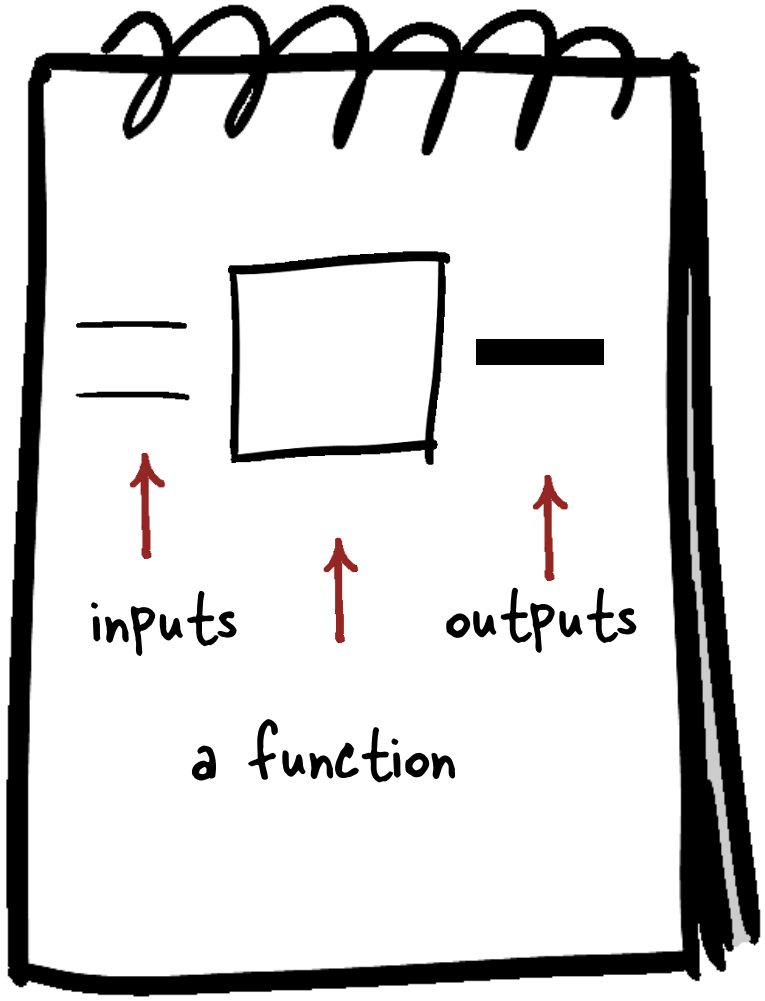
Django

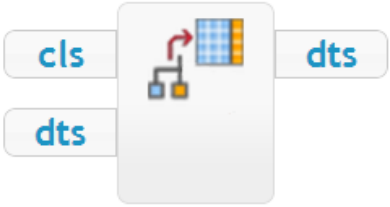
RabbitMQ

Django Celery

SOAP Web services – PySimpleSoap

The widget





Apply Classifier

```
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

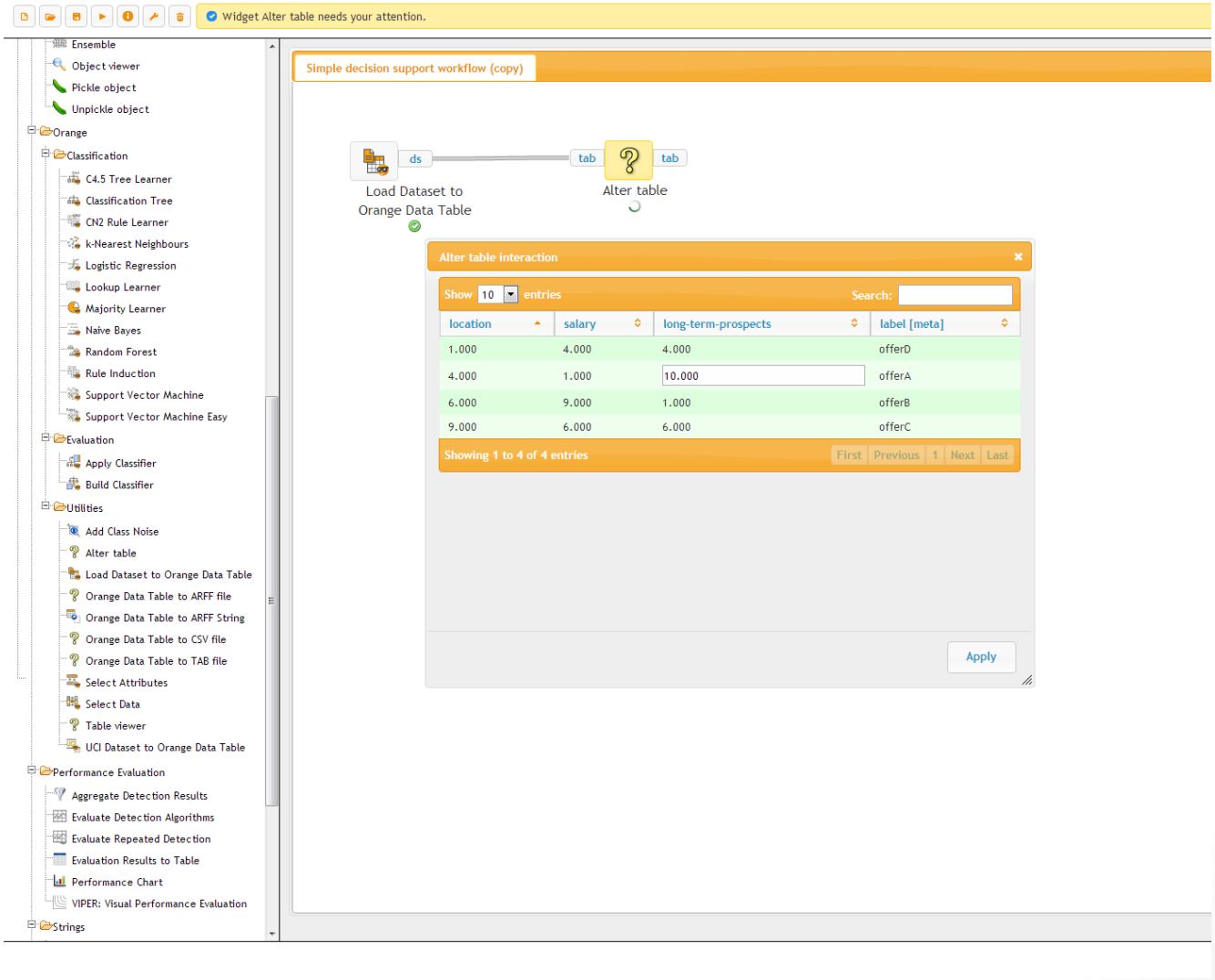
- Extended versions of regular widgets
- Visualization widgets also return HTML and JavaScript that is rendered in the user‘s browser
- Visualization widgets are regular widgets with the addition of a Python function which control the rendering of a template.

Example visualization widget

Interactive widgets

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

Example interactive widget



Workflow control widgets

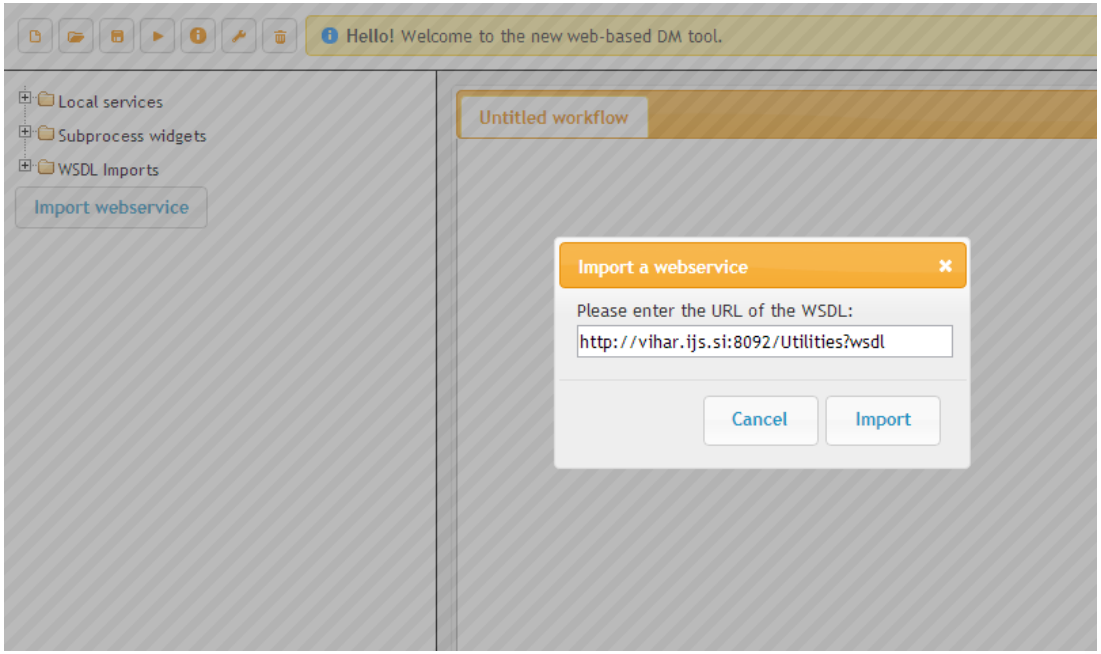
- Sub-workflow widget
- Input widget
- Output widget
- For loops (and cross validation)

The workflow execution engine

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

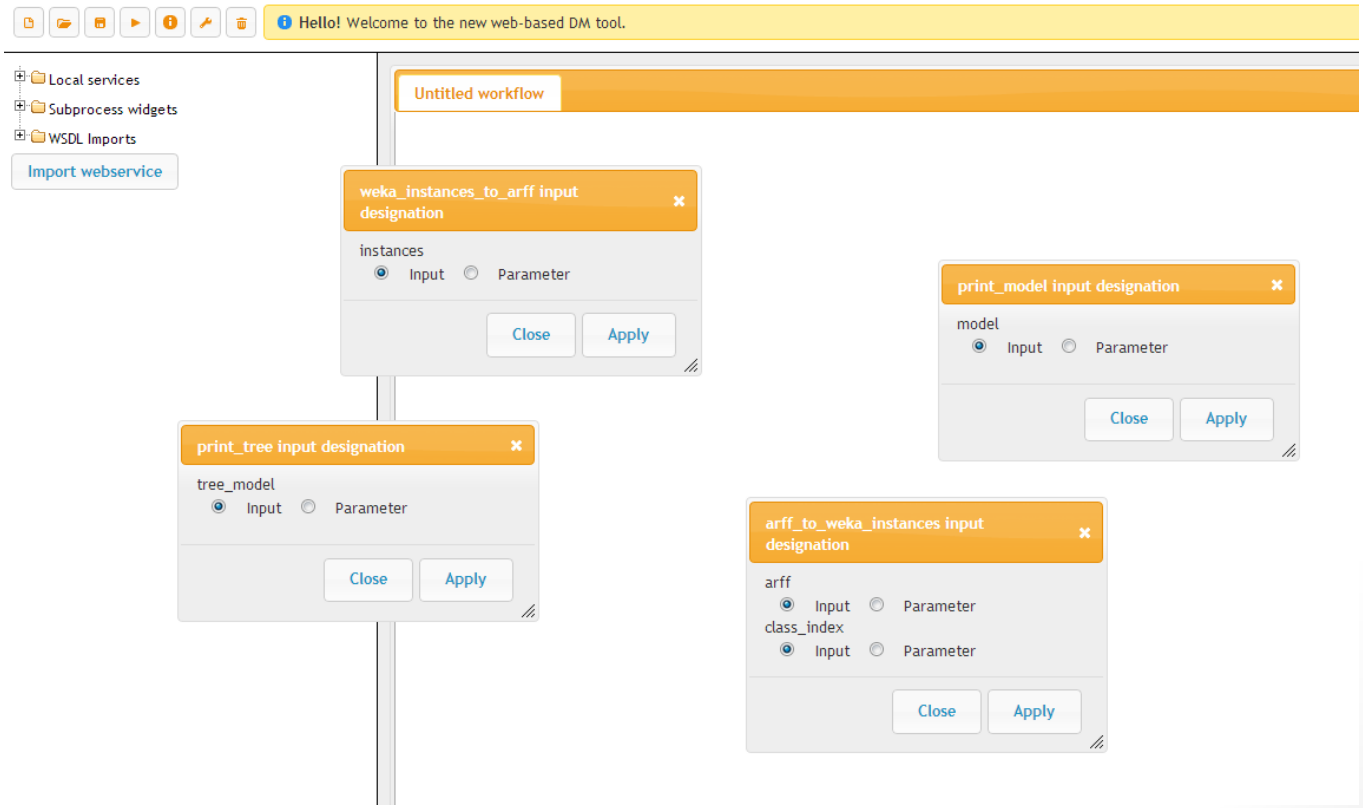
Expanding the widget repository

- With Web services



Expanding the widget repository

- With Web services



Expanding the widget repository

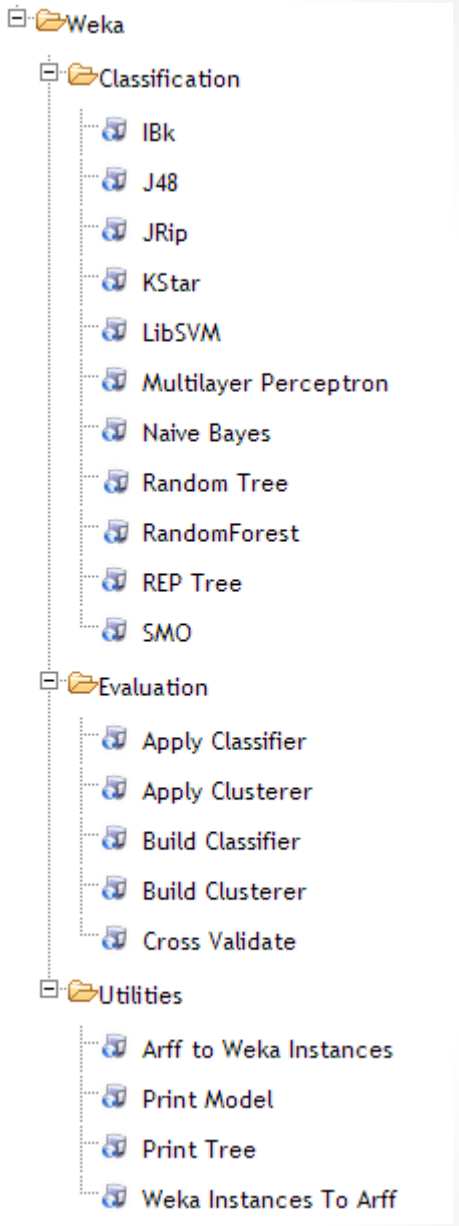
- By adding new Python functions directly to the source code
  - More powerful

Packages

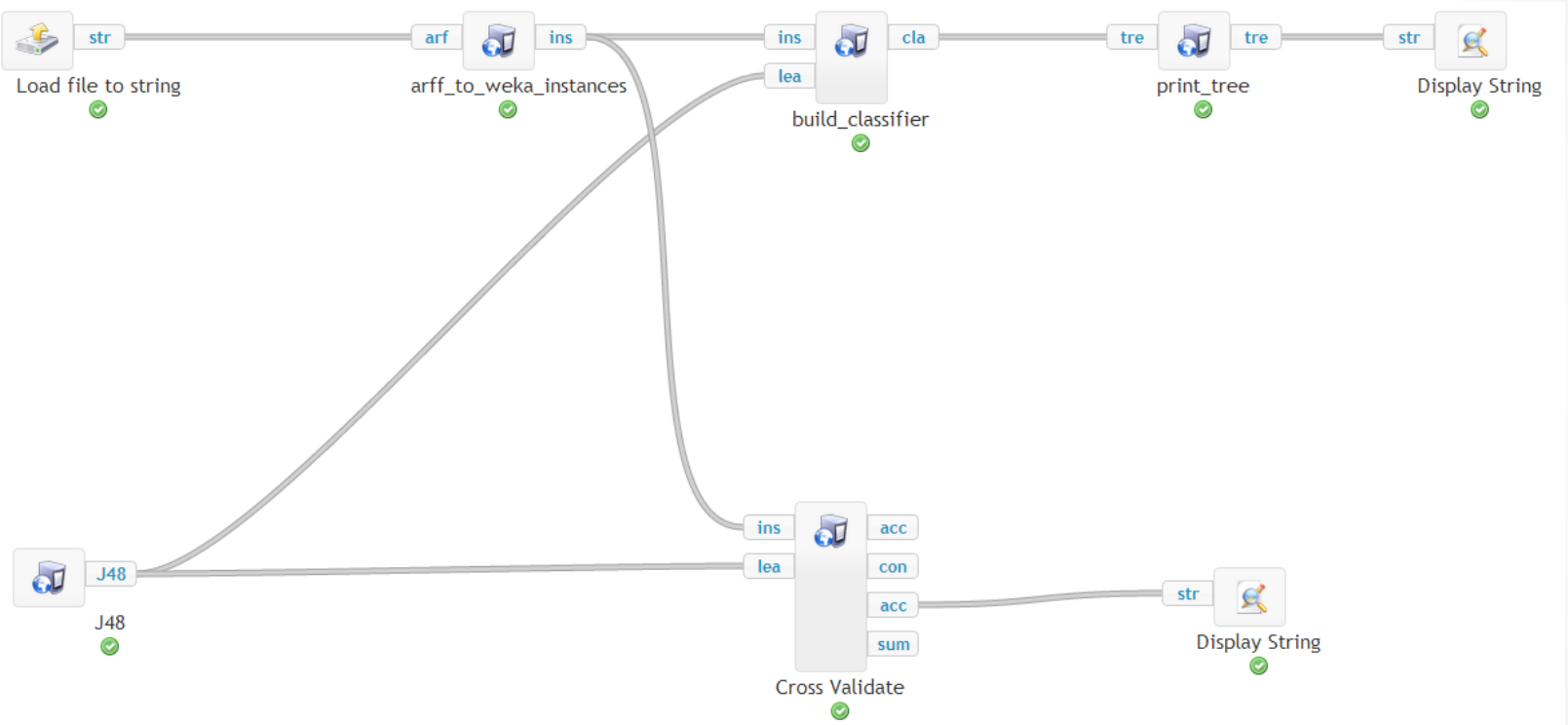
- Widgets are joined in packages which allows
  - Distributed development
  - Enabling/disabling widgets that are not useful to a particular user
- Packages currently include
  - Base package (basic data manipulation and preprocessing)
  - Scikit-learn package
  - Orange package (implementations of the 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

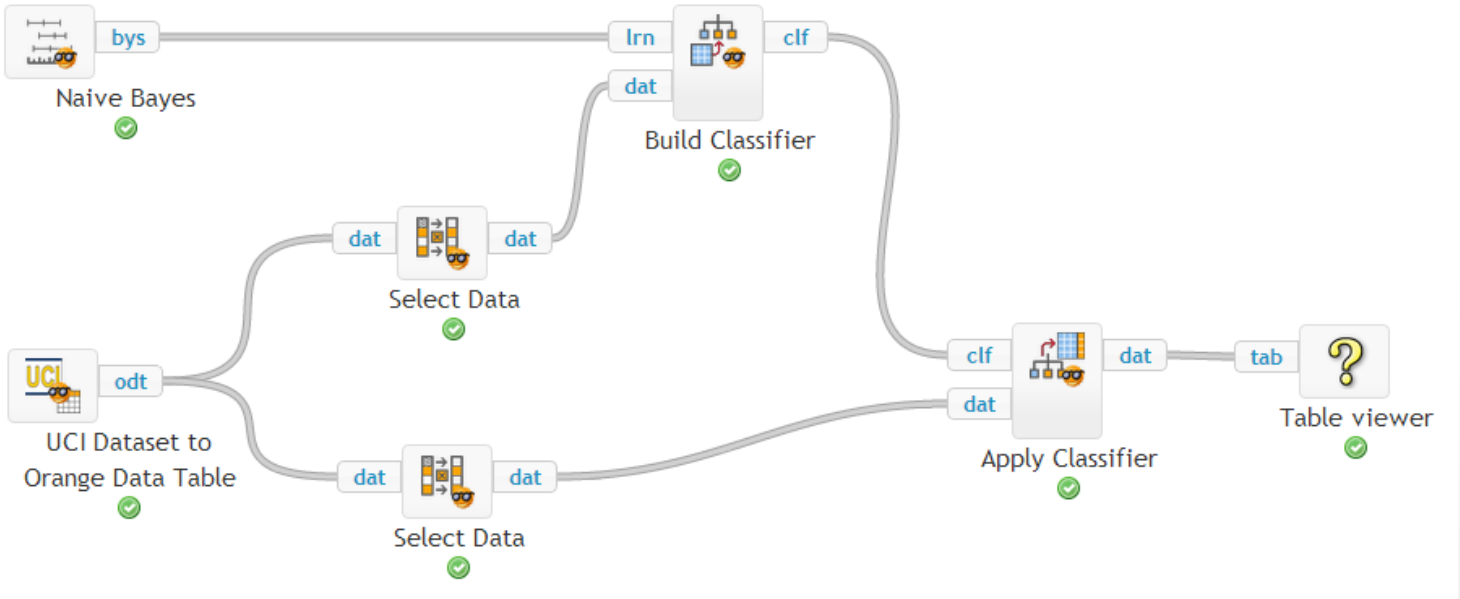


Weka widgets



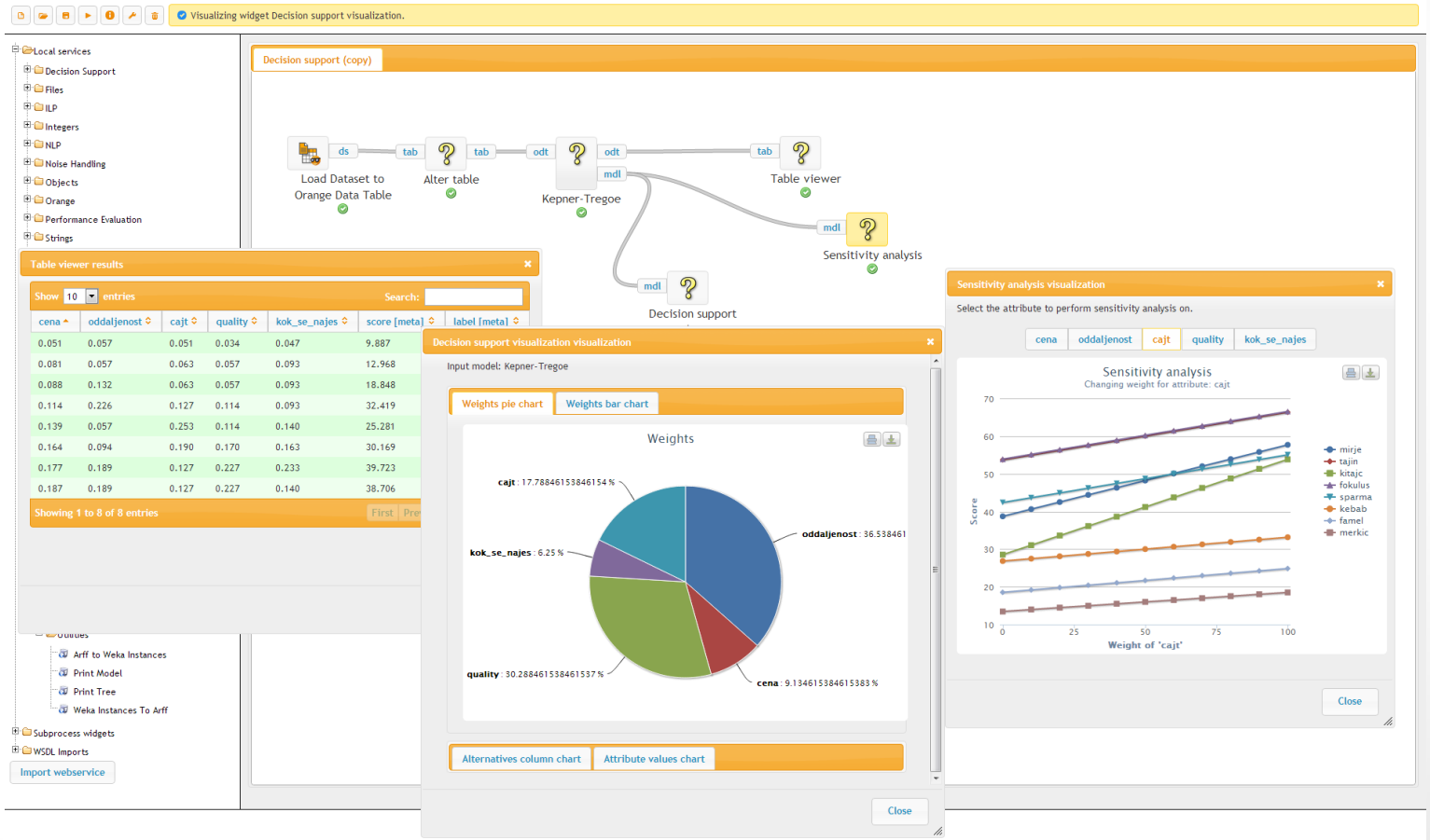
Orange widgets

- Python functions wrapped in ClowdFlows widgets

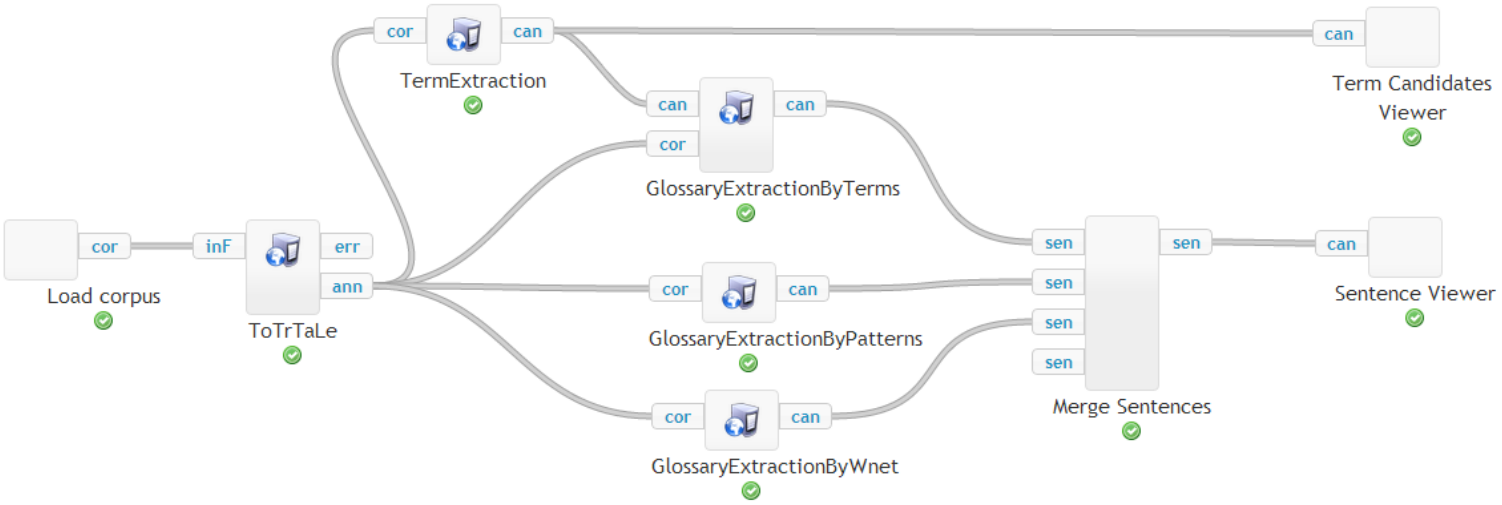


Decision support

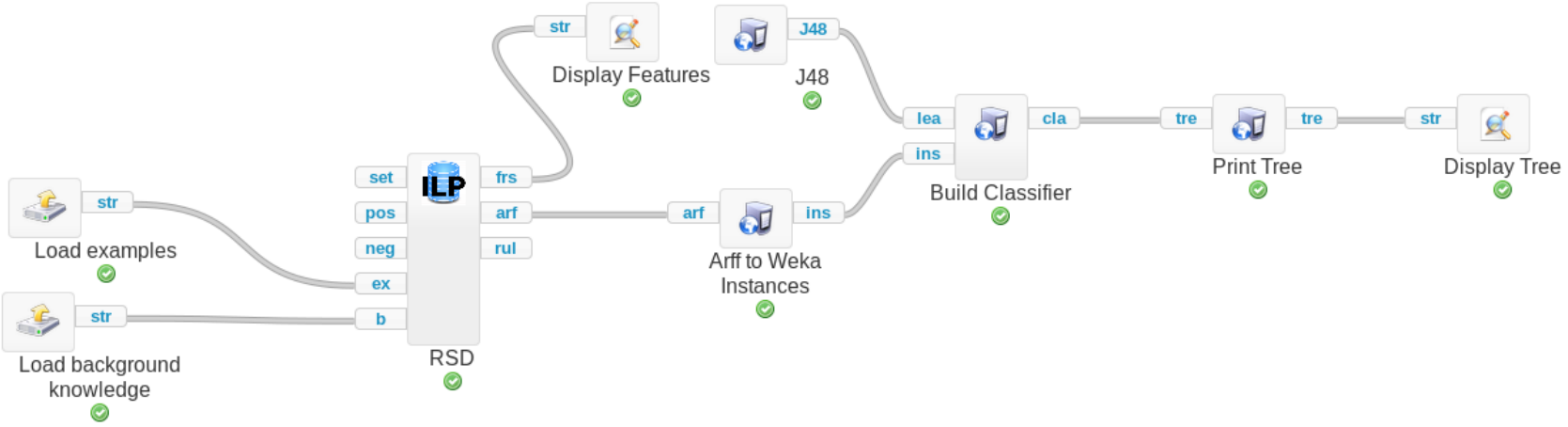
- Python functions built from scratch



Natural Language Processing



ILP



Display Features visualization

```
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).
```

Close

Display Tree visualization

```
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
```

Close

Real-time processing module

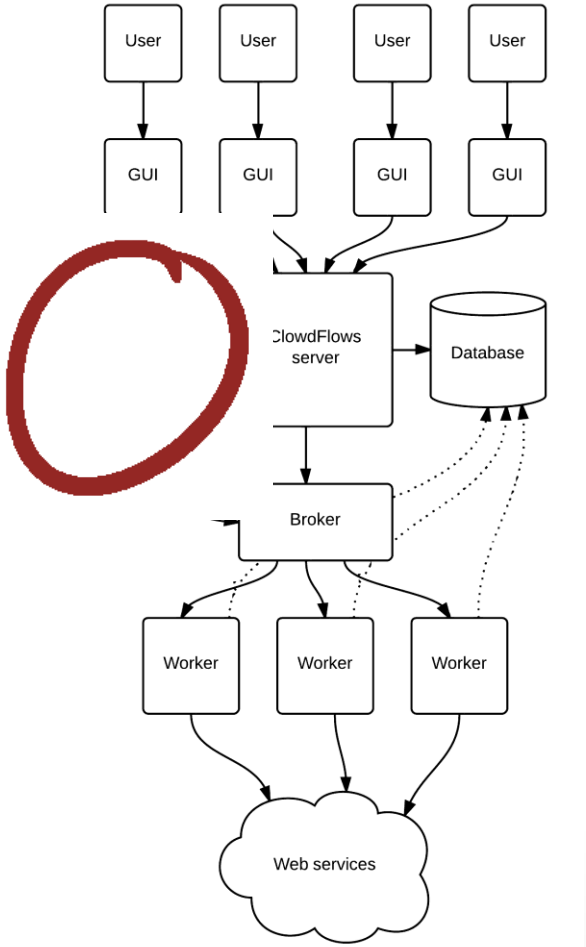
Regular workflows and stream mining workflows

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

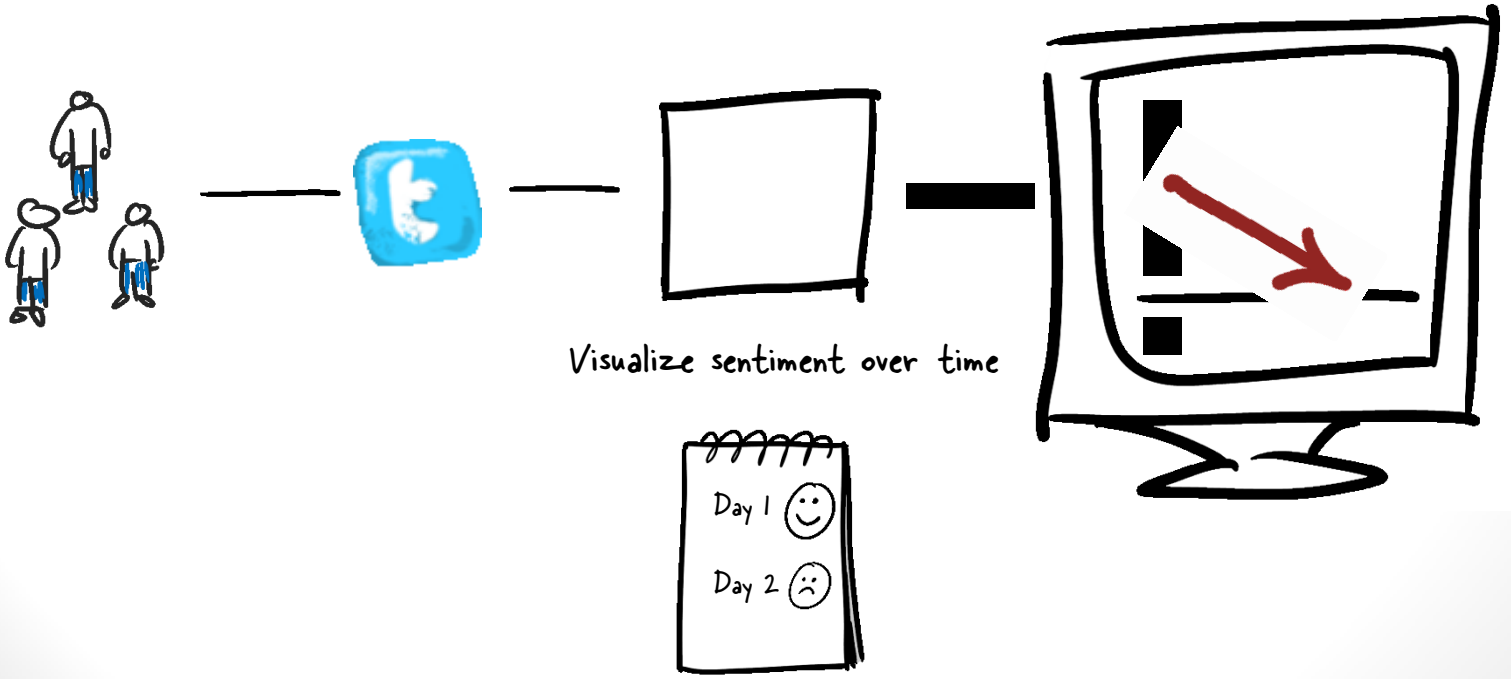
Stream mining workflows

- 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

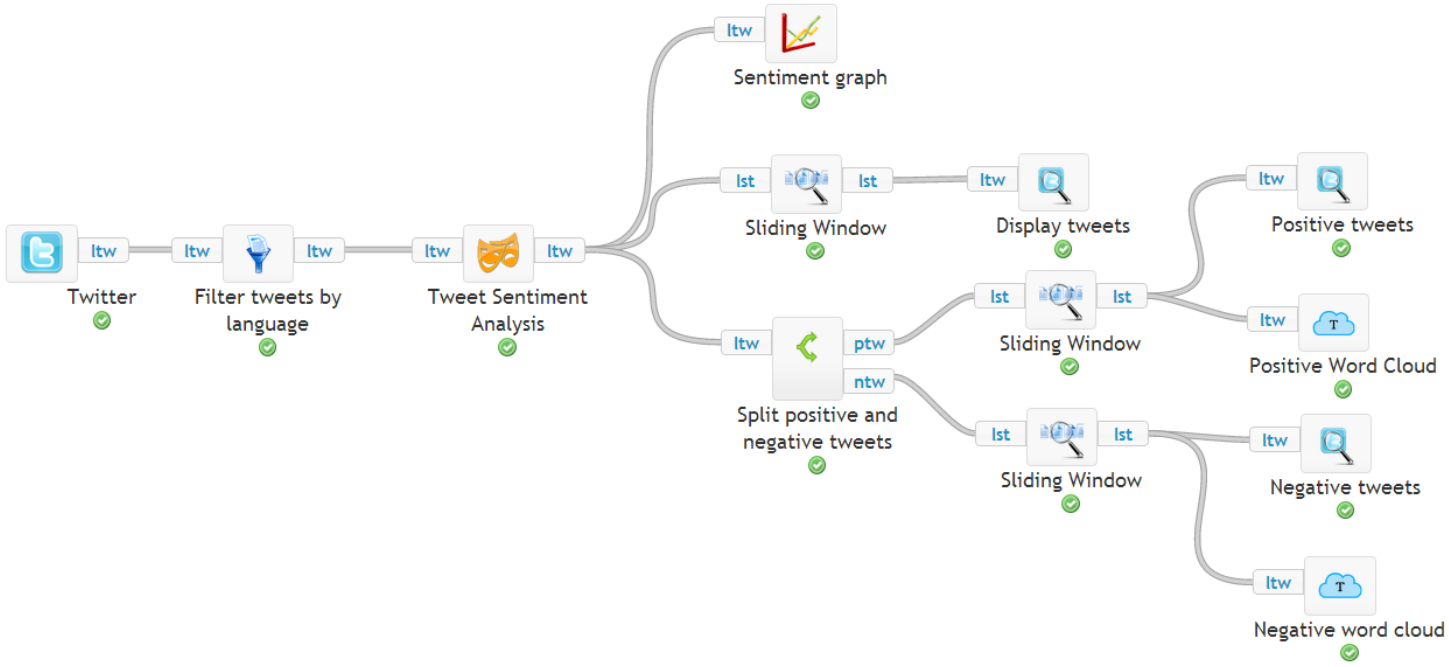


Real-time processing module

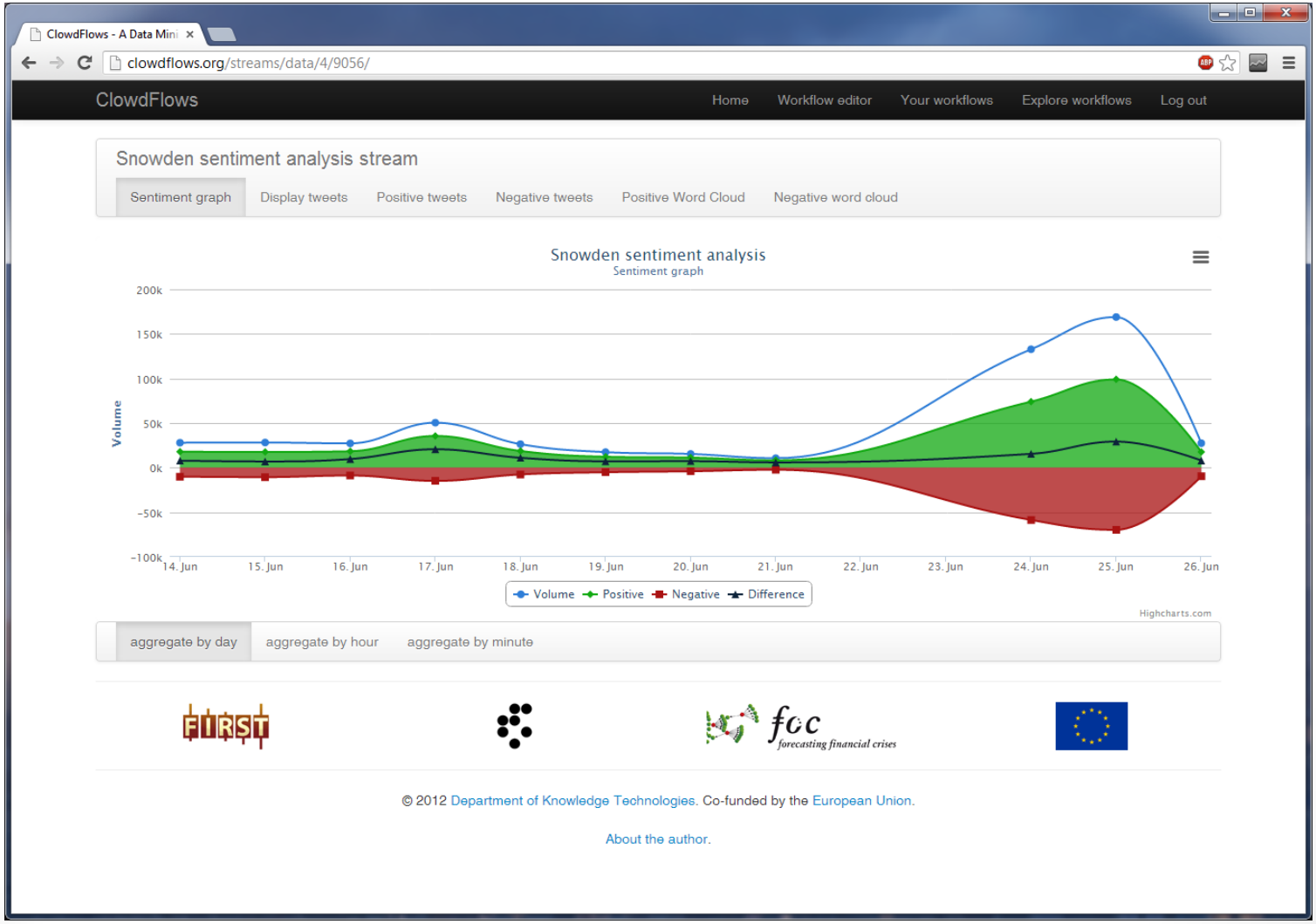
- In order to create streaming workflows we need widgets that are capable of handling streams
- Every stream mining workflow needs at least one **streaming widget**
  - Streaming widgets have additional persistent memory



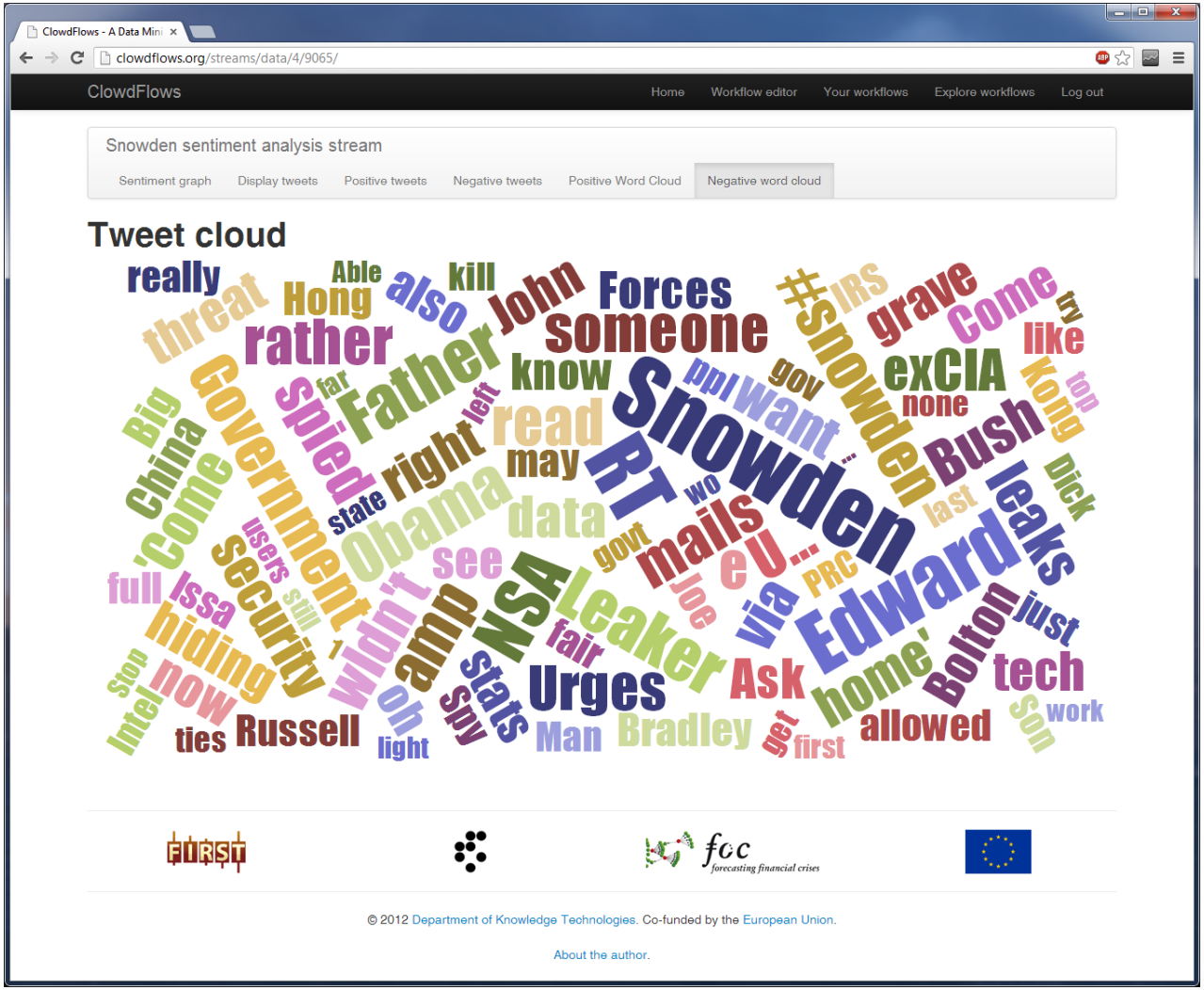
Sentiment Analysis Example



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