
datastream.io

An open-source framework for real-time anomaly detection using Python, Elasticsearch and Kibana.

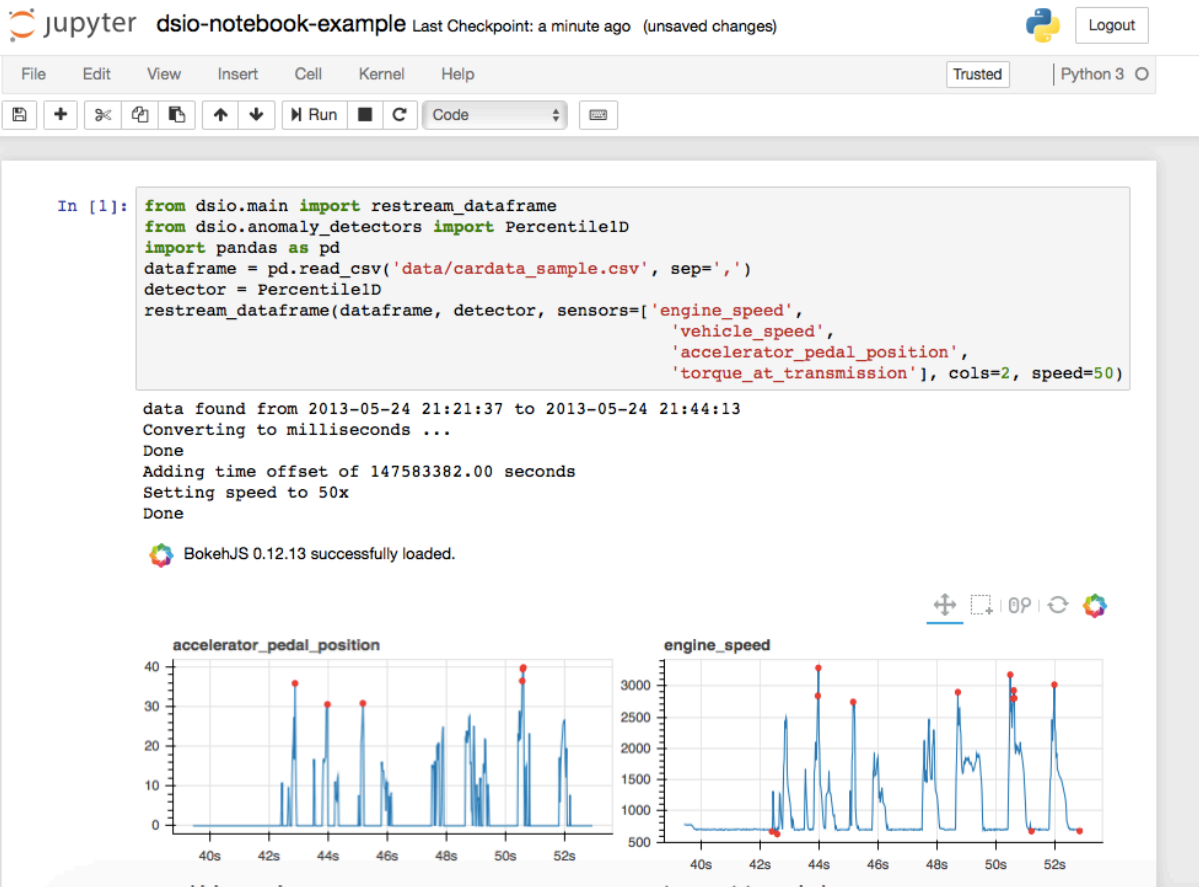
Installation

The recommended installation method is to use pip within a Python 3.x virtualenv.

```
1 virtualenv --python=python3 dsio-env
2 source dsio-env/bin/activate
3 pip install -e git+https://github.com/MentatInnovations/datastream.io#egg=dsio
```

Usage

You can use dsio through the command line or import it in your Python code. You can visualize your data streams using the built-in Bokeh server or you can restream them to Elasticsearch and visualize them with Kibana. In either case, dsio will generate an appropriate dashboard for your stream. Also, if you invoke dsio through a Jupyter notebook, it will embed the streaming Bokeh dashboard within the same notebook.



Examples

For this section, it is best to run commands from inside the `examples` directory. If you have installed dsio via pip as demonstrated above, you'd need to run the following command:

```
1 cd dsio-env/src/dsio/examples
```

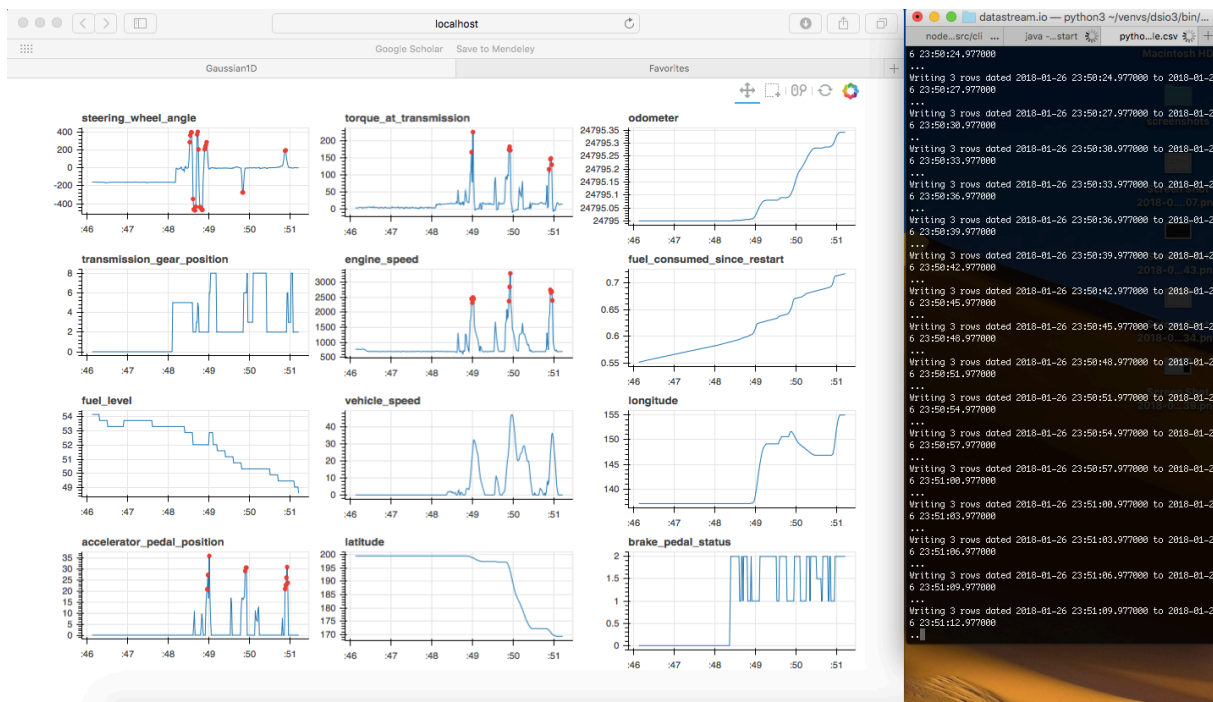
If instead you cloned the github repo then just `cd dsio/examples` will do.

You can use the example csv datasets or provide your own. If the dataset includes a time dimension, dsio will attempt to detect it automatically. Alternatively, you can use the `--timefield` argument to manually configure the field that designates the time dimension. If no such field exists, dsio will assume the data is a time series starting from now with 1sec intervals between samples.

```
1 dsio data/cardata_sample.csv
```

The above command will load the cardata sample csv and will use the default Gaussian1D anomaly detector to apply scores on every numeric column. Then it will generate an appropriate Bokeh dash-

board and restream the data. A browser window should open that will point to the generated dashboard.



You can experiment with different datasets and anomaly detectors. E.g.

```
1 dsio --detector percentile1d path_to_my_dataset/my_dataset.csv
```

You can select specific columns using the `--sensors` argument and you can increase or decrease the streaming speed using the `--speed` argument.

```
1 dsio --sensors accelerator_pedal_position engine_speed --detector gaussian1d --speed 5 data/cardata_sample.csv
```

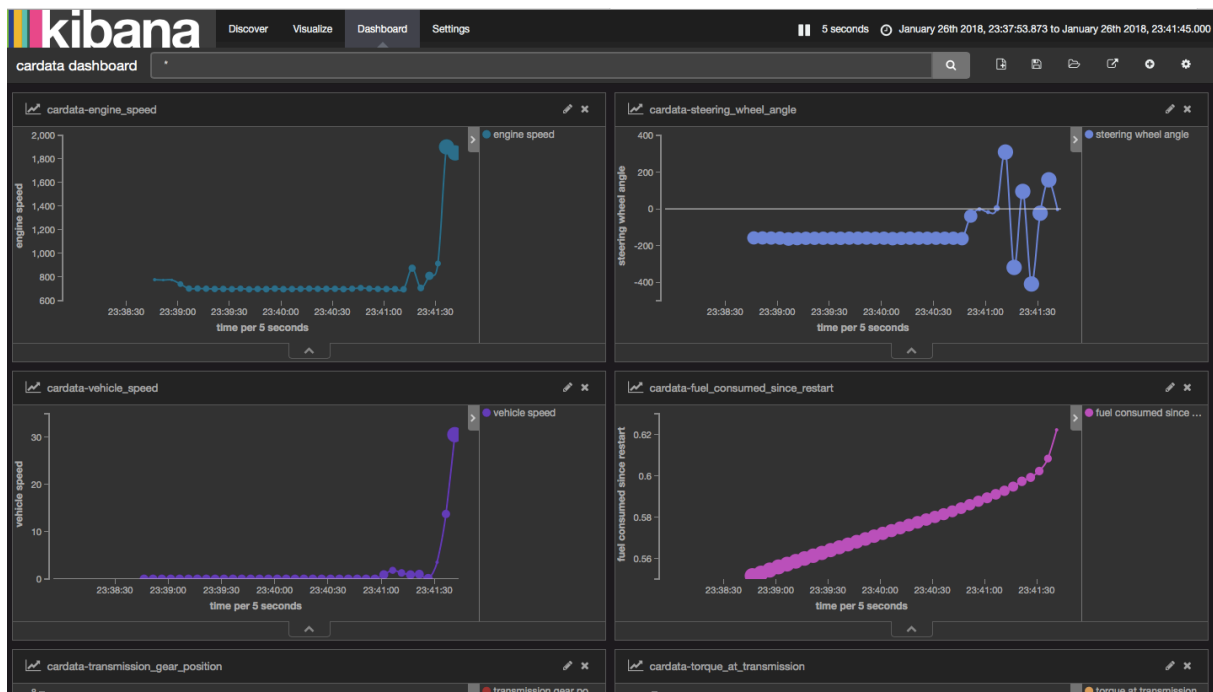
Elasticsearch & Kibana (optional)

In order to restream to an Elasticsearch instance that you're running locally and generate a Kibana dashboard you can use the `--es-uri` and `--kibana-uri` arguments.

```
1 dsio --es-uri http://localhost:9200/ --kibana-uri http://localhost:5601/app/kibana data/cardata_sample.csv
```

If you are using localhost and the default Kibana and ES ports, you can use the shorthand:

```
1 dsio --es data/cardata_sample.csv
```



If you don't have access to Elasticsearch and Kibana 5.x instances, you can easily start them up in your machine using the `docker-compose.yml` file within the `examples` directory. Docker and docker-compose need to be installed for this to work.

```
1 docker-compose up -d
```

Check that Elasticsearch and Kibana are up.

```
1 docker-compose ps
```

Once you're done you can bring them down.

```
1 docker-compose down
```

Keep in mind that docker-compose commands need to be run in the directory where the `docker-compose.yml` file resides (e.g. `dsio-env/src/dsio/examples`)

Defining your own anomaly detectors

You can use dsio with your own hand coded anomaly detectors. These should inherit from the `AnomalyDetector` abstract base class and implement at least the `train`, `update` & `score` methods. You can find an example 99th percentile anomaly detector in the `examples` dir. Load the python modules that contain your detectors using the `--modules` argument and select the target detector by providing its class name to the `--detector` argument (case insensitive).

```
1 dsio --modules detector.py --detector GreaterThanMaxRolling data/
  cardata_sample.csv
```

Integration with scikit-learn

Naturally we encourage people to use `dsio` in combination with `sklearn`: we have no wish to reinvent the wheel! However, `sklearn` currently supports regression, classification and clustering interfaces, but not anomaly detection as a standalone category. We are trying to correct that by the introduction of the `AnomalyMixin`: an interface for anomaly detection which follows `sklearn` design patterns. When you import an `sklearn` object you can therefore simply define or override certain methods to make it compatible with `dsio`. We have provided an example for you here:

```
1 ./datamstream.io/examples/lof_anomaly_detector.py
```