

OpenML in Python

OpenML is an online collaboration platform for machine learning:

- Find or share interesting, well-documented datasets
- Define research / modelling goals (tasks)
- Explore large amounts of machine learning algorithms, with APIs in Java, R, Python
- Log and share reproducible experiments, models, results
- Works seamlessly with scikit-learn and other libraries
- Large scale benchmarking, compare to state of the art

Installation

- `pip install openml`

```
In [0]: !pip install openml
```

Authentication

It is important to configure the Python connector with the proper API endpoint (usually good by default) and the proper API key. Find your API key after logging in on OpenML

```
In [0]: import openml

openml.config.server = 'https://test.openml.org/api/v1/'
openml.config.apikey = 'FILL_IN'
```

Run model/flow on task

Running a scikit-learn model on a task is done using the function `run_model_on_task(...)` (see docs (https://openml.github.io/openml-python/master/generated/openml.runs.run_model_on_task.html#openml.runs.run_model_on_task)) or `run_flow_on_task(...)`. In particular, review the `avoid_duplicate_run` option (especially important for tutorials). The function `get_metric_fn` (doc (<https://openml.github.io/openml-python/master/generated/openml.OpenMLRun.html#openml.OpenMLRun>)) can be used to obtain metric scores before uploading.

- Use the function `run_model_on_task` to run your favorite scikit-learn classifier (e.g., a `Random Forest Classifier`) on the diabetes dataset. (Hint: there are several ways of obtaining a task from the diabetes dataset). Report the score.
- Use the function `run_flow_on_task` to run another scikit-learn classifier on the diabetes dataset. Report the score.

```
In [0]: # Use the function `run_model_on_task` to run your favorite scikit-learn classifier (e.g., a `Random Forest Classifier`) on the `diabetes` dataset.
```

```
In [0]: # Use the function run_flow_on_task to run another scikit-learn classifier on the diabetes dataset
```

Random Search and Grid Search

Scikit-learn natively supports Random Search and Grid Search procedures, to optimize the hyperparameters. These classifiers can natively be used using the openml connector. Read [this article \(https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html\)](https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html) to understand how these work.

- Run Random Search and Grid Search on a SVM from scikit-learn. Make sure to optimize at least 2 hyperparameters. What are the most important hyperparameters? What is the main difference between these two classifiers?

```
In [0]: # Run Random Search
```

```
In [0]: # Run Grid Search
```

ColumnTransformer and Pipelines

Note that we did the previous examples on the diabetes dataset. This is a particularly nice dataset, as it only contains numeric features and no missing values. In many cases, we have to deal with complicated workflows. For example, the `credit-a` dataset mixes categorical and numeric features, and contains missing values.

- verify that our previously used classifier does not work on the `credit-a` dataset. What is the reason for this?
- review the following scikit-learn components:
 - [ColumnTransformer](https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html>)
 - [Pipeline](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>)
 - [SimpleImputer](https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>)
 - [OneHotEncoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>)
 - [StandardScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>)
 - Remember that in order to make a flow compatible with OpenML, there can be no duplicate polymorphic classifiers
- create a general classifier that runs on the `credit-a` (or in general each) dataset. Note that the function `get_features_by_type` from the [OpenMLDataset](https://openml.github.io/openml-python/master/generated/openml.OpenMLDataset.html#openml.OpenMLDataset) (<https://openml.github.io/openml-python/master/generated/openml.OpenMLDataset.html#openml.OpenMLDataset>) object can prove useful.

```
In [0]: # Verify the behaviour
```

```
In [0]: import sklearn.preprocessing
import sklearn.pipeline
import sklearn.feature_selection
import sklearn.compose
import sklearn.impute

nominal_indices = task.get_dataset().get_features_by_type('nominal', [task.target_name])
# numeric_indices = ...

# build a classifier that works on credit-a
```

Pipelines, ColumnTransformers and Random Search

Combine Pipelines, ColumnTransformers and RandomSearchCV to work on any dataset (in particular the `credit-a` dataset). Note that the parameter distribution parameter needs to be adjusted.

```
In [0]: # Much can be copy/pasted ..
```