OpenML Cheat Sheet (Python)

config

Find your API key (required for uploads):

• www.openml.org > Your profile > API Authentication

Main OpenML servers:

- Public: https://www.openml.org/api/v1 (default)
- Test: https://test.openml.org/api/v1

datasets

list_datasets(offset=None, size=None, tag=None)

- · offset and size for paging results
- · tag to filter datasets (e.g. 'uci')
- status: active, in_preparation, deactivated
- data name, data version, number instances,...

get_dataset(dataset_id)

- · returns OpenMLDataset object
- · automatically downloads and caches the data itself

OpenMLDataset

- .features: list of features and their properties
- .qualities: list of all dataset properties
- .get_data (target,return_attribute_names=False,return_categorical_indicator=False returns data as numpy arrays, attribute names, and which are categorical
- .retrieve_class_labels(target_name='class'): return all class labels for the given target attribute

Upload new datasets

- Create a new OpenML dataset with all relevant information
- · datasets.functions.create dataset for uploading pandas dataframes or numpy arrays
- Call .publish() to upload

tasks

list_tasks(task_type_id=None, offset=None, size=None, tag=None)

- offset and size for paging results, tag to filter tags
- task type id: 1=Classification, 2=Regression,...
- · Task IDs do not match dataset IDs

get_task(task_id)

- · returns OpenMLTask object
 - includes estimation procedure, target name, cost matrix....
- · automatically caches the task description

OpenMLTask

- .get_dataset(): downloads associated dataset
- .download_split(): downloads train/test splits

```
# General imports
from openml import datasets, tasks, runs, flows, config, evalua
import os, pandas, sklearn, arff, pprint, numpy, seaborn
```

Set server, API key and cache directory (default: ~/.openml/cache)

```
config.apikey = 'qxlfpbe...ebairtd'
config.server = 'https://...
config.set_cache_directory(os.path.expanduser('~/mycache'))
```

Or, create a config file called $\sim/.openml/config$ and add these lines:

```
server=https://www.openml.org/api/v1
apikey=qxlfpbeaudtprb23985hcqlfoebairtd
cachedir=/homedir/.openml/cache
```

```
dlist = datasets.list datasets(size=100)
pandas.DataFrame.from_dict(dlist, orient='index')[
['name','NumberOfInstances', 'NumberOfFeatures']][:3]
```

	name	NumberOfInstances	NumberOfFeatures	
2	anneal	898	39	
3	kr-vs-kp	3196	37	
4	labor	57	17	

```
odata = datasets.get dataset(1471)
print(odata.name, "Target: "+ odata.default_target_attribute,
     odata.description[260:308], sep='\n')
```

```
eeq-eye-state
Target: Class
All data is from one continuous EEG measurement
```

```
X, y, attribute names = odata.get data(
    target=odata.default_target_attribute,
   return_attribute_names=True)
pandas.DataFrame(X, columns=attribute_names)[:2]
```

	V1	V2	V3	V 4	V 5
0	4329.229980	4009.229980	4289.229980	4148.209961	4350.259766
1	4324.620117	4004.620117	4293.850098	4148.720215	4342.049805

```
md = datasets.OpenMLDataset(data_file='dataset.arff', name='t',
   description='t', version='1', format='ARFF', licence='CC0',
   visibility='public', default_target_attribute='class')
data_id = md.publish()
print("New dataset ID: " + str(data id))
```

New dataset ID: 6677

```
tlist = tasks.list_tasks(task_type_id=1, size=100)
pandas.DataFrame.from_dict(tlist, orient='index')[
['name','estimation_procedure']][:3]
```

name estimation_procedure

- anneal 10-fold Crossvalidation
- 3 kr-vs-kp 10-fold Crossvalidation
- labor 10-fold Crossvalidation

```
task = tasks.get_task(14951)
pprint.pprint(task.estimation_procedure)
```

```
{'data_splits_url': 'https://www.openml.org/api_spl
Task_14951_splits.arff',
 'parameters': {'number_folds': '10'
                 'number_repeats': '1',
                 'percentage': ''
                'stratified_sampling': 'true'},
 'type': 'crossvalidation'}
```

Create new tasks

Under development

flows

list_flows(offset=None, size=None, tag=None, uploader=None)

- · returns ID -> flow dict mapping
- · offset and size for paging results, tag to filter tags
- uploader: list of uploader IDs to filter on, e.g. [1,2,3]

sklearn_to_flow(sklearn_estimator)

• converts a scikit-learn estimator or pipeline to an OpenML Flow

publish()

· Uploads the flow to the server. Returns ID

runs

list_runs(offset=None, size=None, tag=None, id=None,
task=None, flow=None, uploader=None, display_errors=False)

- offset and size for paging results, tag to filter tags
- id, task, flow, uploader: list of IDs to filter, e.g. [1,2,3]
- · display_errors: whether to return failed runs

get_run(run_id)

- · returns OpenMLRun object
 - includes the exact task, exact flow, and all evaluations
- · automatically caches the run description

OpenMLRun

.uploader_name: full name of the run author
.flow_name: full name of the flow
.parameter_settings: hyperparameters of the flow
.evaluations: key-value pairs of metric and score
.fold evaluations: dict of per-fold evaluations

```
run_flow_on_task(flow, task)
run_model_on_task(model, task)
```

- Runs a flow or model (e.g. sklearn model) on the task
- Returns a OpenMLRun with all information
- Trains and tests the flow of all train/test splits defined by the task

publish()

• Publishes the run on OpenML

evaluations

list_evaluations(function=None, offset=None, size=None, tag=None, id=None, task=None, flow=None, uploader=None, display_errors=False)

- function: evaluation measure, e.g. `area_under_roc_curve'
- offset and size for paging results, tag to filter tags
- id, task, flow, uploader: list of IDs to filter, e.g. [1,2,3]
- per_fold: if True, returns per-fold evaluations
- setup: list of hyperparameter setup ID's

Benchmark suites

- Curated collections of tasks for benchmarking
- Run any model or pipeline on all tasks
- · Frictionless evaluation and sharing

```
flist = flows.list_flows(size=200)
pandas.DataFrame.from_dict(flist, orient='index')[
   ['name','version','external_version']][100:102]
```

name version external_version 101 moa.WEKAClassifier_REPTree 1 Moa_2014.03_1.0 102 weka_REPTree 2 Weka_3.7.5_9378

```
lr = sklearn.linear_model.LinearRegression().fit(X, y)
flow = flows.sklearn_to_flow(lr)

pipe = sklearn.pipeline.Pipeline(steps=[
    ('Imputer', sklearn.preprocessing.Normalizer()),
    ('Classifier', sklearn.linear_model.LinearRegression())])
flow2 = flows.sklearn_to_flow(pipe)
# flows.publish(flow)
```

```
rl = runs.list_runs(task=[14951],size=100)
pandas.DataFrame.from_dict(rl, orient='index')[1:5]
```

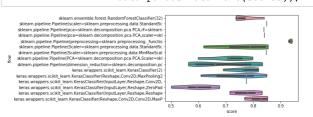
run id task id setup id flow id uploader **544514** 544514 14951 5540 3404 **595116** 595116 14951 6436 4074 **595117** 595117 14951 6436 4074 2 **595118** 595118 14951 6436 4074

	flow	score
17	sklearn.tree.tree.Decision Tree Classifier (2)	0.842479
18	sklearn. ensemble. for est. Random For est Classifier	0.964900
19	mlr.classif.rpart(11)	0.693537

```
task = tasks.get_task(14951)
clf = sklearn.linear_model.LogisticRegression()
run = runs.run_model_on_task(clf, task)
score = run.get_metric_fn(sklearn.metrics.accuracy_score)
myrun = run.publish()

print(myrun)
print("Accuracy: {:.2f}%".format(score.mean()))

[run id: 10154999, task id: 14951, flow id: 9652, f
```



```
benchmark_suite = study.get_study('OpenML-CC18','tasks')
clf = sklearn.linear_model.LogisticRegression()
for task_id in benchmark_suite.tasks[0:2]: # take small subset
    run = runs.run_model_on_task(clf, tasks.get_task(task_id))
    score = run.get_metric_fn(sklearn.metrics.accuracy_score)
    print('Data set: %s; Accuracy: %0.2f' % (task.get_dataset()
    # run.publish()
```

Data set: kr-vs-kp; Accuracy: 0.96
Data set: letter; Accuracy: 0.72

arn.linear_model.logis...]