# 01d - Introduction - OpenML

January 15, 2017

## 1 OpenML in Python

OpenML is an online collaboration platform for machine learning. It allows you to share and reuse machine learning datasets, algorithms, models, and experiments. The openml Python package allows you to do all of this directly from Python scripts, and can be used together with machine learning libraries such as scikit-learn to easily run large-scale machine learning experiments, or to collaborate with others in real time online.

```
In [6]: from preamble import * # Ignore, this is just to make code cleaner
        HTML('''<style>.CodeMirror{min-width:100% !important;}</style>''') # For slides
```

```
Out[6]: <IPython.core.display.HTML object>
```

### 1.1 Authentication

- Create an OpenML account (free) on http://www.openml.org.
- After logging in, open your account page (click your avatar on the top right)
- Open 'Account Settings', then 'API authentication' to find your API key.

There are two ways to authenticate:

- Create a plain text file `~/.openml/config` with the line 'apikey=MYKEY', replacing MYKEY with your API key.
- Run the code below, replacing 'MYKEY' with your API key.

```
In [7]: # Run this to authenticate.  Not needed if you have stored your API key locally
        # or when you launched this notebook from the OpenML website.
        # !!! Never share scripts that contain your API key.
        import openml
        openml.config.apikey = os.environ.get('OPENMLKEY','MYKEY')
```

## 2 Data sets

We can list, select, and download all OpenML datasets

### 2.0.1 List datasets

In [8]:
```
from openml import datasets, tasks, runs
```

In [9]:
```
datalist = datasets.list_datasets()
datalist = pd.DataFrame.from_dict(datalist, orient='index')
print("First 10 of %s datasets..." % len(datalist))
datalist[:10][['did','name','NumberOfInstances','NumberOfFeatures','NumberOfClasses']]
```

```
First 10 of 19492 datasets...
```

Out[9]:

| | did | name | NumberOfInstances | NumberOfFeatures | NumberOfClasses |
|---|---|---|---|---|---|
| 1 | 1 | anneal | 898 | 39 | 6 |
| 2 | 2 | anneal | 898 | 39 | 6 |
| 3 | 3 | kr-vs-kp | 3196 | 37 | 2 |
| 4 | 4 | labor | 57 | 17 | 2 |
| 5 | 5 | arrhythmia | 452 | 280 | 16 |
| 6 | 6 | letter | 20000 | 17 | 26 |
| 7 | 7 | audiology | 226 | 70 | 24 |
| 8 | 8 | liver-disorders | 345 | 7 | -1 |
| 9 | 9 | autos | 205 | 26 | 7 |
| 10 | 10 | lymph | 148 | 19 | 4 |

There are many properties that we can query

In [10]:
```
list(datalist)
```

Out[10]:
```
['MajorityClassSize',
 'MinorityClassSize',
 'NumberOfClasses',
 'NumberOfMissingValues',
 'NumberOfFeatures',
 'format',
 'NumberOfInstances',
 'NumberOfSymbolicFeatures',
 'status',
 'NumberOfNumericFeatures',
 'name',
 'NumberOfInstancesWithMissingValues',
 'did',
 'MaxNominalAttDistinctValues']
```

and we can filter or sort on all of them

In [11]:
```
datalist[datalist.NumberOfInstances>10000].sort(['NumberOfInstances'])[:20][['did','nam
```

Out[11]:

| | did | name | NumberOfInstances | \ |
|---|---|---|---|---|
| 23515 | 23515 | sulfur | 10081 | |
| 372 | 372 | internet_usage | 10108 | |

| | | | |
|---|---|---|---|
| 981 | 981 | kdd_internet_usage | 10108 |
| 1536 | 1536 | volcanoes-b6 | 10130 |
| 4562 | 4562 | InternetUsage | 10168 |
| 1531 | 1531 | volcanoes-b1 | 10176 |
| 1534 | 1534 | volcanoes-b4 | 10190 |
| 1459 | 1459 | artificial-characters | 10218 |
| 1478 | 1478 | har | 10299 |
| 1533 | 1533 | volcanoes-b3 | 10386 |
| 1532 | 1532 | volcanoes-b2 | 10668 |
| 1053 | 1053 | jm1 | 10885 |
| 1414 | 1414 | Kaggle_bike_sharing_demand_challange | 10886 |
| 1044 | 1044 | eye_movements | 10936 |
| 1019 | 1019 | pendigits | 10992 |
| 32 | 32 | pendigits | 10992 |
| 4534 | 4534 | PhishingWebsites | 11055 |
| 399 | 399 | ohscal.wc | 11162 |
| 310 | 310 | mammography | 11183 |
| 1568 | 1568 | nursery | 12958 |

| | NumberOfFeatures |
|---|---|
| 23515 | 7 |
| 372 | 72 |
| 981 | 69 |
| 1536 | 4 |
| 4562 | 72 |
| 1531 | 4 |
| 1534 | 4 |
| 1459 | 8 |
| 1478 | 562 |
| 1533 | 4 |
| 1532 | 4 |
| 1053 | 22 |
| 1414 | 12 |
| 1044 | 28 |
| 1019 | 17 |
| 32 | 17 |
| 4534 | 31 |
| 399 | 11466 |
| 310 | 7 |
| 1568 | 9 |

or find specific ones

In [12]:
```
datalist.query('name == "iris"')[['did','name','NumberOfClasses']]
datalist[datalist['name'].str.contains("eeg")][['did','name']]
```

Out[12]:

| | did | name | NumberOfClasses |
|---|---|---|---|
| 61 | 61 | iris | 3 |
| 969 | 969 | iris | 2 |

```
Out[12]:        did          name
        1471  1471  eeg-eye-state
```

Download a specific dataset. This is done based on the dataset ID (called 'did').

```
In [13]: dataset = datasets.get_dataset(1471)

         print("This is dataset '%s', the target feature is '%s'" % (dataset.name, dataset.defau
         print("URL: %s" % dataset.url)
         print(dataset.description[:500])
```

```
This is dataset 'eeg-eye-state', the target feature is 'Class'
URL: http://www.openml.org/data/download/1587924/phplE7q6h
**Author**: Oliver Roesler, it12148'@'lehre.dhbw-stuttgart.de
**Source**: [UCI](https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State), Baden-Wuerttemberg, Co
**Please cite**:

All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration o
```

Get the actual data

```
In [14]: X, y, attribute_names = dataset.get_data(target=dataset.default_target_attribute, retur
         eeg = pd.DataFrame(X, columns=attribute_names)
         eeg['class'] = y
         print(eeg[:10])
```

```
        V1       V2       V3       V4  ...       V12      V13      V14  class
0  4329.23  4009.23  4289.23  4148.21  ...   4280.51  4635.90  4393.85      0
1  4324.62  4004.62  4293.85  4148.72  ...   4279.49  4632.82  4384.10      0
2  4327.69  4006.67  4295.38  4156.41  ...   4282.05  4628.72  4389.23      0
3  4328.72  4011.79  4296.41  4155.90  ...   4287.69  4632.31  4396.41      0
4  4326.15  4011.79  4292.31  4151.28  ...   4288.21  4632.82  4398.46      0
5  4321.03  4004.62  4284.10  4153.33  ...   4281.03  4628.21  4389.74      0
6  4319.49  4001.03  4280.51  4151.79  ...   4269.74  4625.13  4378.46      0
7  4325.64  4006.67  4278.46  4143.08  ...   4266.67  4622.05  4380.51      0
8  4326.15  4010.77  4276.41  4139.49  ...   4273.85  4627.18  4389.74      0
9  4326.15  4011.28  4276.92  4142.05  ...   4277.95  4637.44  4393.33      0

[10 rows x 15 columns]
```

## 2.1 Train models

Train a scikit-learn model on the data manually

```
In [15]: from sklearn import preprocessing, ensemble

         dataset = datasets.get_dataset(1471)
```

```
X, y = dataset.get_data(target=dataset.default_target_attribute)
clf = ensemble.RandomForestClassifier()
clf.fit(X, y)
```

```
Out[15]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_split=1e-07, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
```

You can also ask which features are categorical to do your own encoding

```
In [16]: X, y, categorical = dataset.get_data(target=dataset.default_target_attribute,return_cat
         enc = preprocessing.OneHotEncoder(categorical_features=categorical)
         X = enc.fit_transform(X)
         clf.fit(X, y)
```

```
Out[16]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_split=1e-07, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
```

# 3 Tasks

To run benchmarks consistently (also across studies and tools), OpenML offers Tasks, which include specific train-test splits and other information to define a scientific task. Tasks are typically created via the website by the dataset provider.

## 3.1 List ALL the tasks

```
In [17]: task_list = tasks.list_tasks(size=10)

         mytasks = pd.DataFrame(task_list).transpose()
         print("First 5 of %s tasks:" % len(mytasks))
         #print(mytasks.columns)
         print(mytasks[:5][['tid','did','name','task_type','estimation_procedure']])
```

```
First 5 of 10 tasks:
  tid did        name                  task_type     estimation_procedure
1   1   1      anneal  Supervised Classification  10-fold Crossvalidation
2   2   2      anneal  Supervised Classification  10-fold Crossvalidation
3   3   3    kr-vs-kp  Supervised Classification  10-fold Crossvalidation
4   4   4       labor  Supervised Classification  10-fold Crossvalidation
5   5   5  arrhythmia  Supervised Classification  10-fold Crossvalidation
```

### 3.2 Download tasks

```
In [18]: task = tasks.get_task(10)
         pprint(vars(task))
```

```
{'class_labels': ['normal', 'metastases', 'malign_lymph', 'fibrosis'],
 'cost_matrix': None,
 'dataset_id': 10,
 'estimation_parameters': {'number_folds': '10',
                           'number_repeats': '1',
                           'percentage': '',
                           'stratified_sampling': 'true'},
 'estimation_procedure': {'data_splits_url': 'http://www.openml.org/api_splits/get/10/Task_10_sp
                          'parameters': {'number_folds': '10',
                                         'number_repeats': '1',
                                         'percentage': '',
                                         'stratified_sampling': 'true'},
                          'type': 'crossvalidation'},
 'evaluation_measure': 'predictive_accuracy',
 'target_name': 'class',
 'task_id': 10,
 'task_type': 'Supervised Classification'}
```

# 4 Runs

We can run (many) scikit-learn algorithms on (many) OpenML tasks.

```
In [19]: task = tasks.get_task(10)
         clf = ensemble.RandomForestClassifier()
         run = runs.run_task(task, clf)
         run.model
```

```
Out[19]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_split=1e-07, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
                     verbose=0, warm_start=False)
```

Share the run on the OpenML server

```
In [20]: myrun = run.publish()
         print("Check your run on http://www.openml.org/r/" + str(myrun.run_id))
```

```
Check your run on http://www.openml.org/r/1846427
```

### 4.1 All together

Train any model on any OpenML dataset and upload to OpenML in a few lines of code

```
In [21]: task = tasks.get_task(14951)
         clf = ensemble.RandomForestClassifier()
         run = runs.run_task(task, clf)
         myrun = run.publish()
         print("Check your run on http://www.openml.org/r/" + str(myrun.run_id))
```

```
Check your run on http://www.openml.org/r/1846428
```

### 4.2 Other possibilities

OpenML's Python API is currently still under development. To be added soon:

- Browse and reuse previous algorithms and pipelines
- Organizing data sets, algorithms, and experiments into studies
- Downloading previous experiments, evaluations and models
- Uploading new datasets to OpenML
- Filters for listings (e.g. filter by author, tags, other properties)

All of this is already possible with the R and Java API.