01d - Introduction - OpenML

January 18, 2017

1 OpenML in Python

OpenML is an online collaboration platform for machine learning:

- Share/reuse machine learning datasets, algorithms, models, experiments
- Well documented/annotated datasets, uniform access
- APIs in Java, R, Python*,... to download/upload everything
- Better reproducibility of experiments, reuse of machine learning models
- Works well with machine learning libraries such as scikit-learn
- Large scale benchmarking, compare to state of the art

```
In [24]: %matplotlib inline
         from preamble import * # Ignore, this is just to make code cleaner
         InteractiveShell.ast_node_interactivity = "all"
         HTML('''<style>html, body{overflow: visible !important} .CodeMirror{min-width:105% !imp

Out[24]: <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 (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 [25]: # Uncomment and run this to authenticate. Don't share your API key!
         # oml.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 [26]: datalist = oml.datasets.list_datasets() # Returns a dict
         datalist = pd.DataFrame.from_dict(datalist, orient='index') # Create a DataFrame
         print("First 10 of %s datasets..." % len(datalist))
         datalist[:10][['did','name','NumberOfInstances',
                        'NumberOfFeatures','NumberOfClasses']]
```

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

```
Out[26]:     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 [27]: list(datalist)
         datalist = datalist[['did','name','NumberOfInstances',
                        'NumberOfFeatures','NumberOfClasses']]
```

```
Out[27]: ['status',
          'NumberOfSymbolicFeatures',
          'did',
          'NumberOfInstances',
          'NumberOfFeatures',
          'MinorityClassSize',
          'NumberOfNumericFeatures',
          'MajorityClassSize',
          'name',
          'NumberOfMissingValues',
          'format',
          'NumberOfInstancesWithMissingValues',
          'NumberOfClasses',
          'MaxNominalAttDistinctValues']
```

and we can filter or sort on all of them

```
In [28]: datalist[datalist.NumberOfInstances>10000
                 ].sort(['NumberOfInstances'])[:20]
```

```
Out[28]:          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  NumberOfClasses
         23515                 7               -1
         372                  72               46
         981                  69                2
         1536                  4                5
         4562                 72               -1
         1531                  4                5
         1534                  4                5
         1459                  8               10
         1478                562                6
         1533                  4                5
         1532                  4                5
         1053                 22                2
         1414                 12               -1
         1044                 28                3
         1019                 17                2
         32                   17               10
         4534                 31                2
         399               11466               10
         310                   7                2
         1568                  9                4
```

or find specific ones

```
In [29]: datalist.query('name == "eeg-eye-state"')
```

```
Out[29]:        did          name  NumberOfInstances  NumberOfFeatures  \
1471  1471  eeg-eye-state             14980                15

      NumberOfClasses
1471                2
```

```
In [30]: datalist.query('NumberOfClasses > 50')
```

```
Out[30]:        did                         name  NumberOfInstances  NumberOfFeatures  \
         1491  1491   one-hundred-plants-margin               1600                65
         1492  1492    one-hundred-plants-shape               1600                65
         1493  1493  one-hundred-plants-texture               1599                65
         4546  4546                      Plants              44940                16
         4552  4552           BachChoralHarmony               5665                17

               NumberOfClasses
         1491              100
         1492              100
         1493              100
         4546               57
         4552              102
```

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

```
In [31]: dataset = oml.datasets.get_dataset(1471)

         print("This is dataset '%s', the target feature is '%s'" %
               (dataset.name, dataset.default_target_attribute))
         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
```

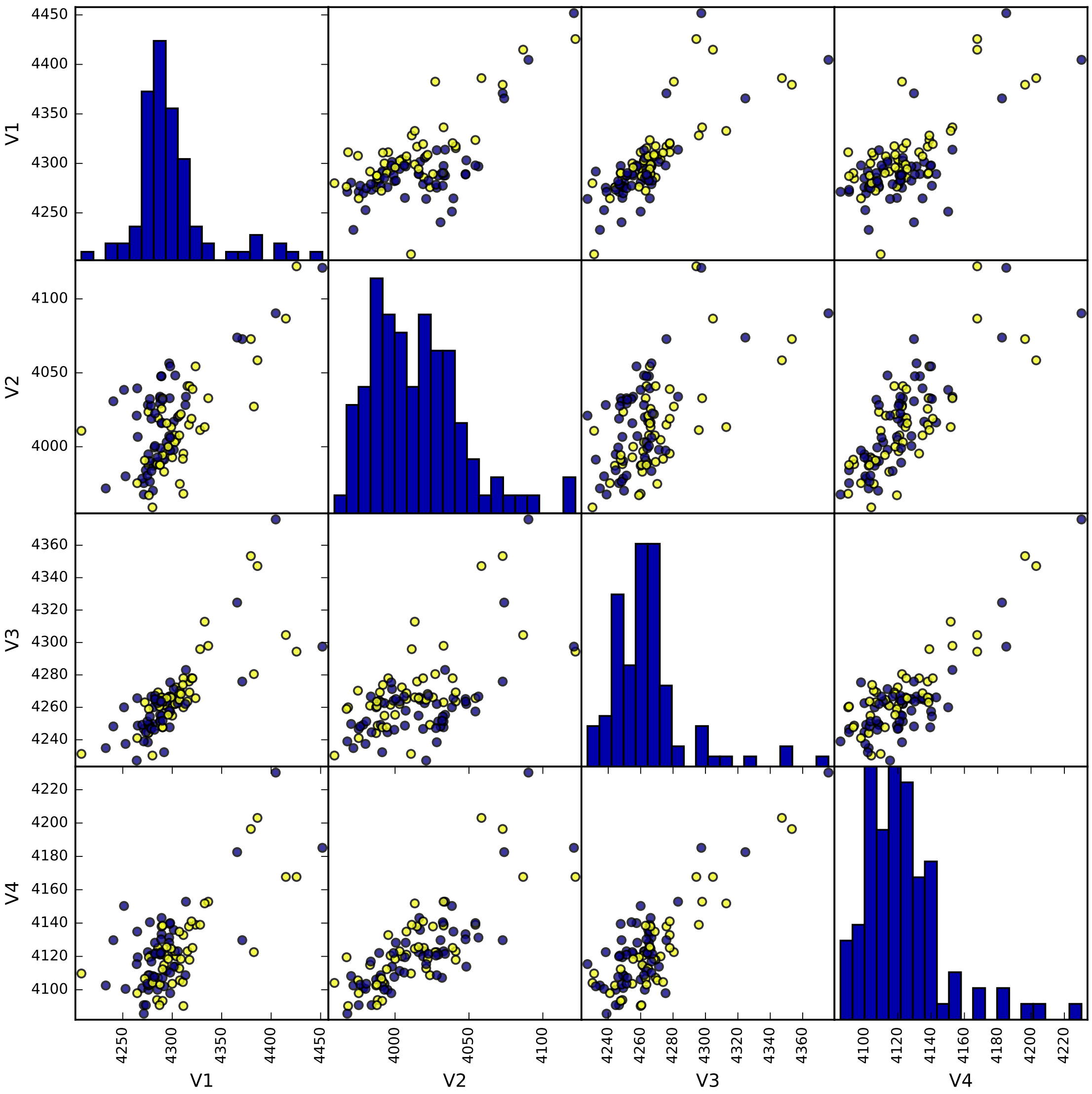
Convert the data to a DataFrame for easier processing/plotting

```
In [32]: X, y, attribute_names = dataset.get_data(
             target=dataset.default_target_attribute,
             return_attribute_names=True)
         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]
```

```
In [43]: eegs = eeg.sample(n=1000)
         _ = pd.scatter_matrix(eegs.iloc[:100,:4], c=eegs[:100]['class'], figsize=(10, 10),
                               marker='o', hist_kwds={'bins': 20},
                               alpha=.8, cmap='plasma')
```



2.1 Train models

Train a scikit-learn model on the data manually

```
In [34]: from sklearn import neighbors

         dataset = oml.datasets.get_dataset(1471)
         X, y = dataset.get_data(target=dataset.default_target_attribute)
         clf = neighbors.KNeighborsClassifier(n_neighbors=1)
         clf.fit(X, y)
```

```
Out[34]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                    weights='uniform')
```

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

```
In [35]: from sklearn import preprocessing
         dataset = oml.datasets.get_dataset(10)
         X, y, categorical = dataset.get_data(
             target=dataset.default_target_attribute,
             return_categorical_indicator=True)
         print("Categorical features: %s" % categorical)
         enc = preprocessing.OneHotEncoder(categorical_features=categorical)
         X = enc.fit_transform(X)
         clf.fit(X, y)
```

```
Categorical features: [True, True, True, True, True, True, True, True, False, False, True, True,
```

```
Out[35]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                    weights='uniform')
```

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 Listing tasks

```
In [36]: task_list = oml.tasks.list_tasks(size=5000) # Get first 5000 tasks

         mytasks = pd.DataFrame(task_list).transpose()
         print("First 5 of %s tasks:" % len(mytasks))
         print(mytasks.columns)
```

```
First 5 of 5000 tasks:
Index(['MajorityClassSize', 'MaxNominalAttDistinctValues', 'MinorityClassSize',
       'NumberOfClasses', 'NumberOfFeatures', 'NumberOfInstances',
       'NumberOfInstancesWithMissingValues', 'NumberOfMissingValues',
       'NumberOfNumericFeatures', 'NumberOfSymbolicFeatures', 'cost_matrix',
       'did', 'estimation_procedure', 'evaluation_measures', 'name',
       'number_samples', 'quality_measure', 'source_data',
       'source_data_labeled', 'status', 'target_feature',
       'target_feature_event', 'target_feature_left', 'target_feature_right',
       'target_value', 'task_type', 'tid', 'time_limit', 'ttid'],
      dtype='object')
```

```
In [37]: mytasks = mytasks[['tid','did','name','task_type','estimation_procedure','evaluation_me
         print(mytasks.head())
```

```
   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

   evaluation_measures
1  predictive_accuracy
2  predictive_accuracy
3  predictive_accuracy
4  predictive_accuracy
5  predictive_accuracy
```

Search for the tasks you need

```
In [38]: print(mytasks.query('name=="eeg-eye-state"'))
```

```
         tid   did           name                  task_type  \
9983    9983  1471  eeg-eye-state  Supervised Classification
14951  14951  1471  eeg-eye-state  Supervised Classification

          estimation_procedure  evaluation_measures
9983   10-fold Crossvalidation  predictive_accuracy
14951  10-fold Crossvalidation                  NaN
```

3.2 Download tasks

```
In [39]: task = oml.tasks.get_task(14951)
         pprint(vars(task))
```

```
{'class_labels': ['1', '2'],
 'cost_matrix': None,
 'dataset_id': 1471,
 'estimation_parameters': {'number_folds': '10',
                           'number_repeats': '1',
                           'percentage': '',
                           'stratified_sampling': 'true'},
 'estimation_procedure': {'data_splits_url': 'http://www.openml.org/api_splits/get/14951/Task_14
                          'parameters': {'number_folds': '10',
                                         'number_repeats': '1',
                                         'percentage': '',
                                         'stratified_sampling': 'true'},
                          'type': 'crossvalidation'},
 'evaluation_measure': None,
 'target_name': 'Class',
 'task_id': 14951,
 'task_type': 'Supervised Classification'}
```

4 Runs: Train models on tasks

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

```
In [40]: task = oml.tasks.get_task(14951)
         clf = neighbors.KNeighborsClassifier(n_neighbors=1)
         run = oml.runs.run_task(task, clf)
         run.model
```

```
Out[40]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                    weights='uniform')
```

Share the run on the OpenML server

```
In [41]: myrun = run.publish()
         print("Uploaded to http://www.openml.org/r/" + str(myrun.run_id))
```

```
Uploaded to http://www.openml.org/r/1846520
```

4.1 All together

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

```
In [44]: task = oml.tasks.get_task(14951)
         clf = neighbors.KNeighborsClassifier(n_neighbors=5)
         run = oml.runs.run_task(task, clf)
         myrun = run.publish()
         print("Uploaded to http://www.openml.org/r/" + str(myrun.run_id))
```

```
Uploaded to http://www.openml.org/r/1846522
```

4.2 A Challenge

We'll see many machine learning algorithms in this course. Try to build the best possible models on several OpenML tasks, and compare your results with the rest of the class, and learn from them. Some tasks you could try (or browse openml.org):

- EEG eye state: data_id:1471, task_id:14951
- Volcanoes on Venus: data_id:1527, task_id:10103
- Walking activity: data_id:1509, task_id: 9945, 150k instances
- Covertype (Satellite): data_id:150, task_id: 218. 500k instances
- Higgs (Physics): data_id:23512, task_id:52950. 100k instances, missing values

Easy benchmarking:

```
In [45]: import openml as oml
         from sklearn import neighbors

         for task_id in [14951,10103,9945]:
             task = oml.tasks.get_task(task_id)
             data = oml.datasets.get_dataset(task.dataset_id)
             clf = neighbors.KNeighborsClassifier(n_neighbors=5)
             run = oml.runs.run_task(task, clf)
             myrun = run.publish()
             print("kNN on %s: http://www.openml.org/r/%d" % (data.name, myrun.run_id))
```

```
kNN on eeg-eye-state: http://www.openml.org/r/1846523
kNN on volcanoes-a1: http://www.openml.org/r/1846524
kNN on walking-activity: http://www.openml.org/r/1846525
```

4.3 Other possibilities

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

- Support for uploading 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.