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

Exercise

- Find datasets with more than 10000 examples
- Find a dataset called 'eeg_eye_state'
- Find all datasets with more than 50 classes

In [ ]:
```
# List datasets
import openml as oml
openml_list = oml.datasets.list_datasets()

# Show a nice table
import pandas as pd
datalist = pd.DataFrame.from_dict(openml_list, orient='index')
datalist = datalist[['did','name','NumberOfInstances','NumberOfFeatures','NumberOfClasses'
]]
print("First 10 of %s datasets..." % len(datalist))
datalist.head(n=10)
```

In [ ]:
```
datalist[datalist.NumberOfInstances>10000
        ].sort_values(['NumberOfInstances']).head(n=20)
```

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

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

Download datasets

Download the `eeg_eye_state` dataset. This is done based on the dataset ID ('did').

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

# Print a summary
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])
```

Get the actual data.
Returned as numpy array, with meta-info (e.g. target feature, feature names,...)

In [ ]:
```
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])
```

Exercise

- Explore the data visually

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

Task

The function `openml.evaluation.list_evaluations(...)` returns a dictionary of evaluation records. It has several filtering functions, to keep the resulting set small (keep in mind that OpenML has almost 10 million runs, and more than a billion evaluation records). The function is documented in the API docs (https://openml.github.io/openml-python/master/generated/openml.evaluations.list_evaluations.html#openml.evaluations.list_evaluations). It returns a dict mapping from `run_id` to OpenMLEvaluation (https://openml.github.io/openml-python/master/generated/openml.OpenMLEvaluation.html#openml.OpenMLEvaluation). Examples of filters are `task`, `flow` and `function`. Note that one of these is mandatory.

- Obtain a subset of 100 predictive accuracy (`predictive_accuracy`) results on the letter dataset (task id = 6).
- Obtain a subset of 100 predictive accuracy (`predictive_accuracy`) results per task in the OpenML 100 and plot these

In [ ]:
```
import seaborn as sns
import pandas as pd
import openml as oml

suite = oml.study.get_study('OpenML100')
scores = []
for task_id in suite.tasks[:10]: # [SPEED] only first 10 tasks
  results = oml.evaluations.list_evaluations(function='predictive_accuracy', task=[task_id], size=100)
  # Download the tasks and plot the scores
  for evaluation in results.values():
    scores.append({"flow": evaluation.flow_name, "score": evaluation.value, "task": evaluation.data_name})

sns.violinplot(x="task", y="score", data=pd.DataFrame(scores), scale="width", palette="Set3");
```

Dataset Upload

There are various ways to upload a dataset. The most convenient ways are documented in this example (https://github.com/openml/openml-python/blob/master/examples/create_upload_tutorial.py). Most conveniently, this can be done using a pandas dataframe (https://github.com/openml/openml-python/blob/a0ef724fec6ab31f6381d3ac2a84827ab535170d/examples/create_upload_tutorial.py#L206). Additionally, we need to create a OpenMLDataset (https://openml.github.io/openml-python/master/generated/openml.OpenMLDataset.html#openml.OpenMLDataset) object, containing information about the dataset. Most notably, the arguments `name`, `default_target_attribute`, `attributes` and `data` need to be set.

- Find your favorite dataset (on your laptop), load it as pandas dataframe and upload it to OpenML.
- Common problem: Server returns error 131. This means that the description file was not complete. The XSD (https://github.com/openml/OpenML/blob/master/openml_OS/views/pages/api_new/v1/xsd/openml.data.upload.xsd) for uploading the dataset hints what fields are mandatory.

In [ ]:
```
data = [
    ['sunny', 85, 85, 'FALSE', 'no'],
    ['sunny', 80, 90, 'TRUE', 'no'],
    ['overcast', 83, 86, 'FALSE', 'yes'],
    ['rainy', 70, 96, 'FALSE', 'yes'],
    ['rainy', 68, 80, 'FALSE', 'yes'],
    ['rainy', 65, 70, 'TRUE', 'no'],
    ['overcast', 64, 65, 'TRUE', 'yes'],
    ['sunny', 72, 95, 'FALSE', 'no'],
    ['sunny', 69, 70, 'FALSE', 'yes'],
    ['rainy', 75, 80, 'FALSE', 'yes'],
    ['sunny', 75, 70, 'TRUE', 'yes'],
    ['overcast', 72, 90, 'TRUE', 'yes'],
    ['overcast', 81, 75, 'FALSE', 'yes'],
    ['rainy', 71, 91, 'TRUE', 'no'],
]

attribute_names = [
    ('outlook', ['sunny', 'overcast', 'rainy']),
    ('temperature', 'REAL'),
    ('humidity', 'REAL'),
    ('windy', ['TRUE', 'FALSE']),
    ('play', ['yes', 'no']),
]

description = (
    'The weather problem is a tiny dataset that we will use repeatedly'
    ' to illustrate machine learning methods. Entirely fictitious, it '
    'supposedly concerns the conditions that are suitable for playing '
    'some unspecified game. In general, instances in a dataset are '
    'characterized by the values of features, or attributes, that measure '
    'different aspects of the instance. In this case there are four '
    'attributes: outlook, temperature, humidity, and windy. '
    'The outcome is whether to play or not.'
)

oml.config.server = 'https://test.openml.org/api/v1/xml'
oml.config.apikey = 'FILL_IN_APIKEY'

df = pd.DataFrame(data, columns=[col_name for col_name, _ in attribute_names])
# enforce the categorical column to have a categorical dtype
df['outlook'] = df['outlook'].astype('category')
df['windy'] = df['windy'].astype('bool')
df['play'] = df['play'].astype('category')
print(df.info())

#############################################################################
# We enforce the column 'outlook', 'windy', and 'play' to be a categorical
# dtype while the column 'rnd_str' is kept as a string column. Then, we can
# call :func:`create_dataset` by passing the dataframe and fixing the parameter
# ``attributes`` to ``'auto'``.

weather_dataset = oml.datasets.create_dataset(
    name="Weather",
    description=description,
    default_target_attribute='play',
    attributes='auto',
    data=df,
    creator=None,
    contributor=None,
    collection_date=None,
    language=None,
    licence=None,
    ignore_attribute=None,
    citation=None,
)


upload_did = weather_dataset.publish()
print('URL for dataset: %s/data/%d' % (oml.config.server, upload_did))
```