Analyzing data with OpenML

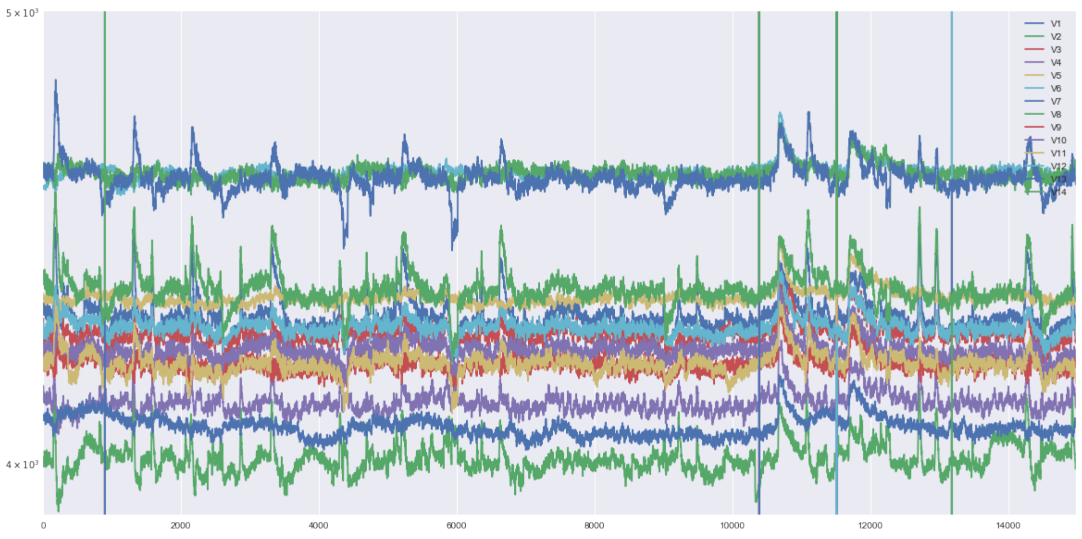
This is a simple example where we: - Download an EEG dataset from OpenML - Visualize it - Build and analyze machine learning models locally - Train, evaluate and upload a classifier to OpenML - Compare it to all other models built on that same dataset by other people

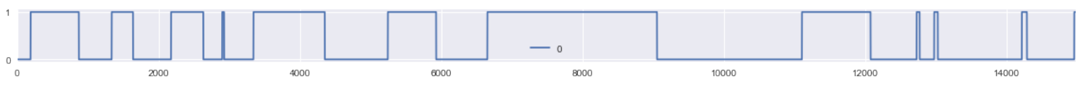
```
[2]: %matplotlib inline
     import openml as oml
     import pandas as pd
     import numpy as np
     import seaborn as sns
     from matplotlib import pyplot
     from sklearn import neighbors, model_selection
```

Download dataset, extract data, plot

The dataset (#1471 on OpenML) contains EEG data (top) labeled with whether your eyes are open or closed at the time of measurement (bottom).

```
[193]: dataset = oml.datasets.get_dataset(1471)
       X, y, attribute_names = dataset.get_data(target=dataset.default_target_at
       eeg = pd.DataFrame(X, columns=attribute_names)
       eeg.plot(logy=True,ylim=(3900,5000),figsize=(20,10))
       pd.DataFrame(y).plot(figsize=(20,1));
```



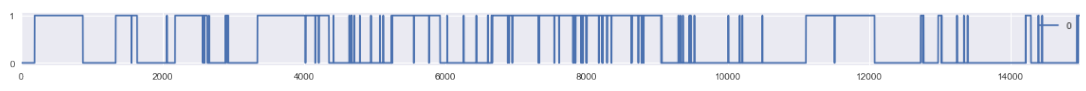


Train simple machine learning model and predict

Using scikit-learn

```
[152]: X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y)
       clf = neighbors.KNeighborsClassifier(n_neighbors=1)
       clf.fit(X_train, y_train)
       pd.DataFrame(clf.predict(X)).plot(figsize=(20,1))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x14cb40f98>
```



or evaluate

```
[112]: kfold = model_selection.StratifiedKFold(n_splits=5, shuffle=True, random_
       results = model_selection.cross_val_score(clf, X, y, cv=kfold)
       print("Accuracy: %.3f%% (+- %.3f)" % (results.mean(), results.std()))
```

```
Accuracy: 0.976% (+- 0.002)
```

Use OpenML tasks to easily build, evaluate, and upload models

A completely self-contained experiments in 5 lines of code: - Download the task (a wrapper around the data also including evaluation details, e.g. train/test splits) - Create any scikit-learn classifier (or pipeline) - Convert the pipeline to an OpenML 'flow' and run it on the task - Publish (upload) if you want

```
[3]: task = oml.tasks.get_task(14951)
     clf = neighbors.KNeighborsClassifier(n_neighbors=1)
     flow = oml.flows.sklearn_to_flow(clf)
     run = oml.runs.run_flow_on_task(task, flow)
     myrun = run.publish()
     print("Uploaded to http://www.openml.org/r/" + str(myrun.run_id))
```

```
        -------------------------------------------------------------------------

        PyOpenMLError                             Traceback (most recent call la

        <ipython-input-3-2e2546d22246> in <module>()
          2 clf = neighbors.KNeighborsClassifier(n_neighbors=1)
          3 flow = oml.flows.sklearn_to_flow(clf)
    ----> 4 run = oml.runs.run_flow_on_task(task, flow)
          5 myrun = run.publish()
          6 print("Uploaded to http://www.openml.org/r/" + str(myrun.run_id))
```

```
/Users/joa/anaconda/lib/python3.5/site-packages/openml/runs/functions.py
 85         ids = _run_exists(task.task_id, setup_id)
 86         if ids:
---> 87             raise PyOpenMLError("Run already exists in server. Run i
 88         _copy_server_fields(flow_from_server, flow)
 89
```

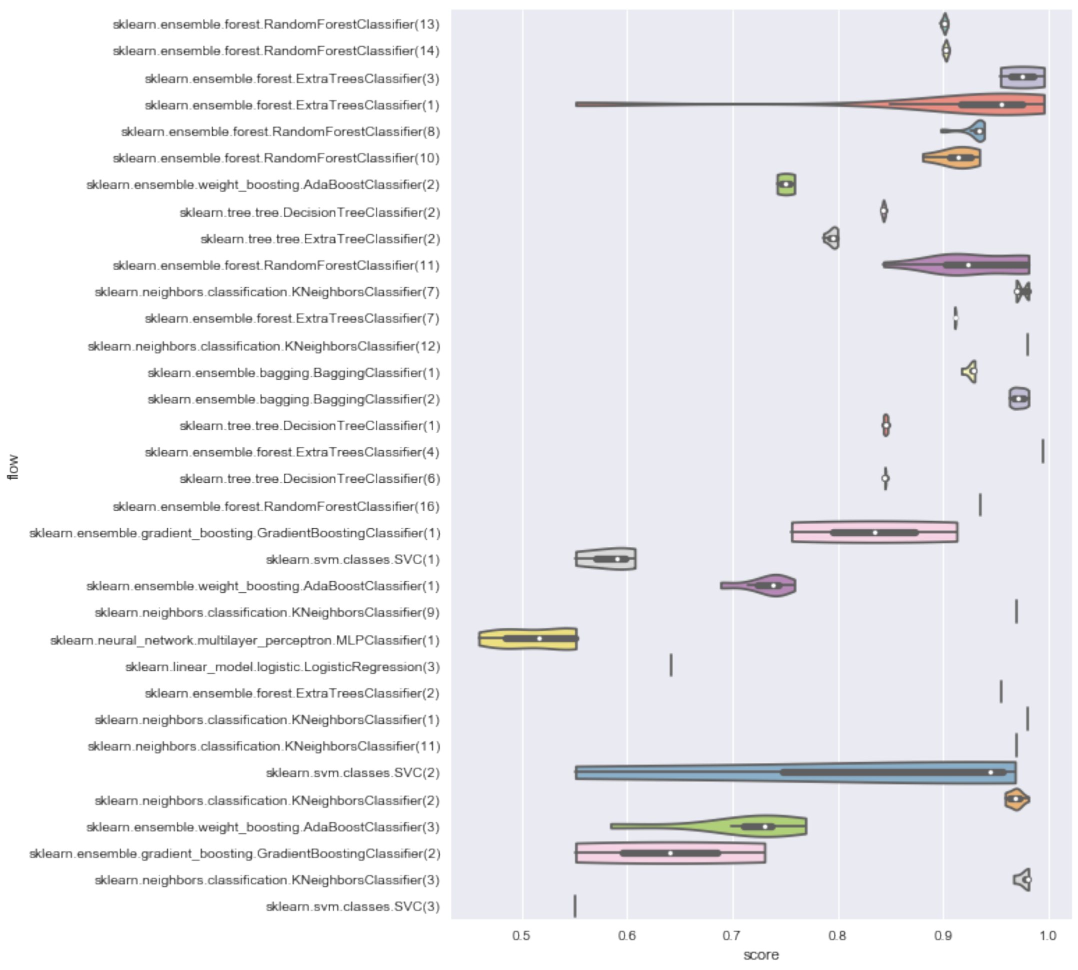
```
PyOpenMLError: Run already exists in server. Run id(s): {7932096}
```

Download everyone else's results on the same dataset

Check whether other people built better models on the same task by downloading their evaluations (computed on the OpenML server) and comparing directly against them.

```
[ ]: myruns = oml.runs.list_runs(task=[14951],size=10000)
     scores = []
     for id, _ in myruns.items():
         run = oml.runs.get_run(id)
         if str.startswith(run.flow_name, 'sklearn'):
             scores.append({"flow":run.flow_name, "score":run.evaluations['predi
```

```
[190]: fig, ax = pyplot.subplots(figsize=(8, 12))
       sns.violinplot(x="score", y="flow", data=pd.DataFrame(scores), scale="wid
```


flow
sklearn.ensemble.forest.RandomForestClassifier(13)
sklearn.ensemble.forest.RandomForestClassifier(14)
sklearn.ensemble.forest.ExtraTreesClassifier(3)
sklearn.ensemble.forest.ExtraTreesClassifier(1)
sklearn.ensemble.forest.RandomForestClassifier(8)
sklearn.ensemble.forest.RandomForestClassifier(10)
sklearn.ensemble.weight_boosting.AdaBoostClassifier(2)
sklearn.tree.tree.DecisionTreeClassifier(2)
sklearn.tree.tree.ExtraTreeClassifier(2)
sklearn.ensemble.forest.RandomForestClassifier(11)
sklearn.neighbors.classification.KNeighborsClassifier(7)
sklearn.ensemble.forest.ExtraTreesClassifier(7)
sklearn.neighbors.classification.KNeighborsClassifier(12)
sklearn.ensemble.bagging.BaggingClassifier(1)
sklearn.ensemble.bagging.BaggingClassifier(2)
sklearn.tree.tree.DecisionTreeClassifier(1)
sklearn.ensemble.forest.ExtraTreesClassifier(4)
sklearn.tree.tree.DecisionTreeClassifier(6)
sklearn.ensemble.forest.RandomForestClassifier(16)
sklearn.ensemble.gradient_boosting.GradientBoostingClassifier(1)
sklearn.svm.classes.SVC(1)
sklearn.ensemble.weight_boosting.AdaBoostClassifier(1)
sklearn.neighbors.classification.KNeighborsClassifier(9)
sklearn.neural_network.multilayer_perceptron.MLPClassifier(1)
sklearn.linear_model.logistic.LogisticRegression(3)
sklearn.ensemble.forest.ExtraTreesClassifier(2)
sklearn.neighbors.classification.KNeighborsClassifier(1)
sklearn.neighbors.classification.KNeighborsClassifier(11)
sklearn.svm.classes.SVC(2)
sklearn.neighbors.classification.KNeighborsClassifier(2)
sklearn.ensemble.weight_boosting.AdaBoostClassifier(3)
sklearn.ensemble.gradient_boosting.GradientBoostingClassifier(2)
sklearn.neighbors.classification.KNeighborsClassifier(3)
sklearn.svm.classes.SVC(3)
0.5
0.6
0.7
0.8
0.9
1.0
score

```
[ ]:
```