

Le but du TP est d'étudier et de comparer différentes méthodes automatique d'apprentissage, à partir de <https://automl.github.io/auto-sklearn/master/>

Ex. 1 — AutoML

1. Quelle est la liste des données disponibles grâce à sklearn ?
2. Générez les données du problème

```
import sklearn.model_selection
import sklearn.datasets
import sklearn.metrics
X, y = sklearn.datasets.load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = \
sklearn.model_selection.train_test_split(X, y, random_state=1)
```

3. Utilisez un algorithme de base de sklearn pour résoudre ce problème

```
from sklearn import ensemble
import time
Boosting_clf = ensemble.GradientBoostingClassifier()
t0 = time.time()
Boosting_clf.fit(X_train, y_train)
print("Computing time %.2f seconds" % (time.time() - t0))
y_hat = Boosting_clf.predict(X_test)
print("Accuracy score %.3f " % (sklearn.metrics.accuracy_score(y_test, y_hat)))
print(sklearn.metrics.classification_report(y_test, y_hat))
```

4. Optimisez les hyperparamètres grace à GridSearchCV

```
from sklearn.model_selection import GridSearchCV
import numpy as np
parameters = {"learning_rate": [0.1, 0.2],
              "max_depth": [3,8],
              "subsample": [0.5, 1.0],}
BoostingCV_clf = GridSearchCV(ensemble.GradientBoostingClassifier(), parameters, cv=10,
                               n_jobs=-1)
t0 = time.time()
BoostingCV_clf.fit(X_train, y_train)
print("Computing time %.2f seconds" % (time.time() - t0))

y_hat = BoostingCV_clf.predict(X_test)
print("Accuracy score %.3f " % (sklearn.metrics.accuracy_score(y_test, y_hat)))
```

5. Enchaenez une méthode de normalisation en utilisant le Pipeline de sklearn <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

steps = [('scaling', StandardScaler()),
         ('classif', ensemble.GradientBoostingClassifier())]
pipeline = Pipeline(steps=steps)

t0 = time.time()
pipeline.fit(X_train, y_train)
print("Computing time %.2f seconds" % (time.time() - t0))

y_hat = pipeline.predict(X_test)
print("Accuracy score %.3f " % (sklearn.metrics.accuracy_score(y_test, y_hat)))
```

6. Essayez maintenant de coupler GridSearchCV et le pipeline

```
parameters = {
    "scaling__with_std": ['True', 'False'],
    "classifier__learning_rate": [0.1, 0.2],
    "classifier__max_depth": [3,8],
    "classifier__subsample": [0.5, 1.0],
}
steps = [('scaling', StandardScaler()),
         ('classifier', ensemble.GradientBoostingClassifier())
        ]
pipeline = Pipeline(steps=steps)
Boost_pipeline_CV_clf = GridSearchCV(pipeline, parameters, cv=5, n_jobs=-1)
t0 = time.time()
Boost_pipeline_CV_clf.fit(X_train, y_train)
print("Computing time %.2f seconds" % (time.time() - t0))
y_hat = Boost_pipeline_CV_clf.predict(X_test)
print("Accuracy score %.3f " % (sklearn.metrics.accuracy_score(y_test, y_hat)))
print(Boost_pipeline_CV_clf.best_params_)
```

7. AutoML

a) Essayez avec autoML, avec un budget de 60 secondes, après l'avoir installé (si besoin)

<https://automl.github.io/auto-sklearn/master/>

pip install auto-sklearn -user

```
import autosklearn.classification

automl=autosklearn.classification.AutoSklearnClassifier(time_left_for_this_task=30)
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
print("Accuracy score", sklearn.metrics.accuracy_score(y_test, y_hat))
```

b) analysez les résultats

```
automl.sprint_statistics()
print(automl.get_models_with_weights())
```

c) Essayez avec TPOT, MLJar et AutoGluon, après les avoir installé (attention aux problèmes de versions)

% <http://epistasislab.github.io/tpot/>

pip install tpot

% <https://github.com/mljar/mljar-supervised> pip install mljar-supervised

% <https://github.com/awsmlabs/autogluon>

pip install autogluon.tabular

```
from tpot import TPOTClassifier

pipeline_optimizer = TPOTClassifier(generations=5, population_size=20, cv=10,
random_state=42, verbosity=2)
t0 = time.time()
pipeline_optimizer.fit(X_train, y_train)
print("Computing time %.2f seconds" % (time.time() - t0))

y_hat = pipeline_optimizer.predict(X_test)
print("Accuracy score %.3f " % (sklearn.metrics.accuracy_score(y_test, y_hat)))
```

8. Quelle est la meilleure méthode pour ce jeu de données ?

Ex. 2 — A vous de jouer

1. Réappliquez cette méthode (gridsearchCV, pipeline et automl) sur un autre jeu de données (comme par exemple celles du TP de la semaine dernière). Commentez et concluez.