TP MLA : AutoML

Stéphane Canu

12 décembre 2025, ITI, INSA Rouen

Le but du TP est d'étudier et de comparer différentes méthodes automatique d'apprentissage.

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. Enchainez 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()),
('classifier', 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 AutoGluon, avec un budget de 60 secondes, après les avoir installé (si besoin)

```
% https://github.com/awslabs/autogluon
pip install autogluon.tabular
% https://automl.github.io/auto-sklearn/master/
pip install auto-sklearn -user
```

```
from autogluon.tabular import TabularPredictor, TabularDataset
from autogluon.tabular.configs.hyperparameter_configs import
    get_hyperparameter_config
import pandas as pd

train_data = pd.DataFrame()
test_data = pd.DataFrame()
train_data.loc[:, 'target'] = y_train
[n,p] = X_train.shape
for j in range(p):
 train_data.loc[:,str(j)] = X_train[:,j]
 test_data.loc[:,str(j)] = X_test[:,j]

label = 'target'
MODEL_PRESETS = "best_quality"
model_AutoGluon = TabularPredictor(label=label).fit(train_data, time_limit=60,
    presets=MODEL_PRESETS)
y_pred = model_AutoGluon.predict(test_data)
```

   b) analysez les résultats

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

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

```
% https://github.com/sb-ai-lab/LightAutoML
pip install -U lightautoml
% http://epistasislab.github.io/tpot/
pip install tpot
% https://github.com/mljar/mljar-supervised
pip install mljar-supervised
% https://github.com/automl/TabPFN
pip install tabpfn
```

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

Ex. 2 — A vous de jouer

1. Réappliquez cette méthode (gridsearchCV, pipline et automl) sur un autre jeu de données (comme par exemple celles d'un autre TP). Commentez et concluez.