td2a_progressbar

January 1, 2023

1 2A.i - ProgressBar et fusion de random forest - énoncé

L'idée de ce notebook n'est pas de se servir de faire du machine learning mais de modifier la fonction *fit* pour afficher une barre d'avancement dans le notebook. Lorsque les affichages (print) sont trop nombreux et prennent tout l'écran, une barre de défilement est une solution pratique et efficace. On applique cela à un assemblage de random forest.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Module tqdm

Le module tqdm permet d'afficher le progrès d'un processus assez long. Quelques exemples issus de la documentation.

[2]:
```
from tqdm import tqdm
pbar = tqdm(["a", "b", "c", "d"])
for char in pbar:
    pbar.set_description("Processing %s" % char)
```

```
Processing d:
100%|¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿␣
 ↪¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ | 4/4
[00:00<00:00, 7989.15it/s]
```

[3]:
```
from tqdm import tnrange
from random import random, randint
from time import sleep

t = tnrange(100)
for i in t:
    # Description will be displayed on the left
    t.set_description('GEN %i' % i)
    # Postfix will be displayed on the right, and will format automatically
    # based on argument's datatype
    t.set_postfix(loss=random(), gen=randint(1,999), str='h', lst=[1, 2])
    sleep(0.1)
```

1.2 RandomForest

L'article Combining random forest models in scikit learn explique comment fusionner des random forest. L'objectif est d'en apprendre 10 sur n'importe quel jeu de données en affichant une barre d'avancement. A quoi sert le paramètre warm_start ? Peut-on imaginer tracer la décroissance du taux d'erreur en fonction du nombre d'arbres ?

[4]:
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
X, y = make_classification(n_samples=500, n_features=25,
                           n_clusters_per_class=1, n_informative=15)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

[5]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

[5]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

[6]:
```

```