Rapport TP2 Arbres

Mathieu Le Séac'h

Classification avec les arbres

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rc

from sklearn import tree, datasets, model_selection

from itertools import product
from collections import defaultdict

from tp_arbres_source import (rand_gauss, rand_bi_gauss, rand_tri_gauss,
                              rand_checkers, rand_clown,
                              plot_2d, frontiere)
```

Q1)

Dans le cadre d'une régression, on peut utiliser TODO comme mesure d'homogénéité.

Q2)

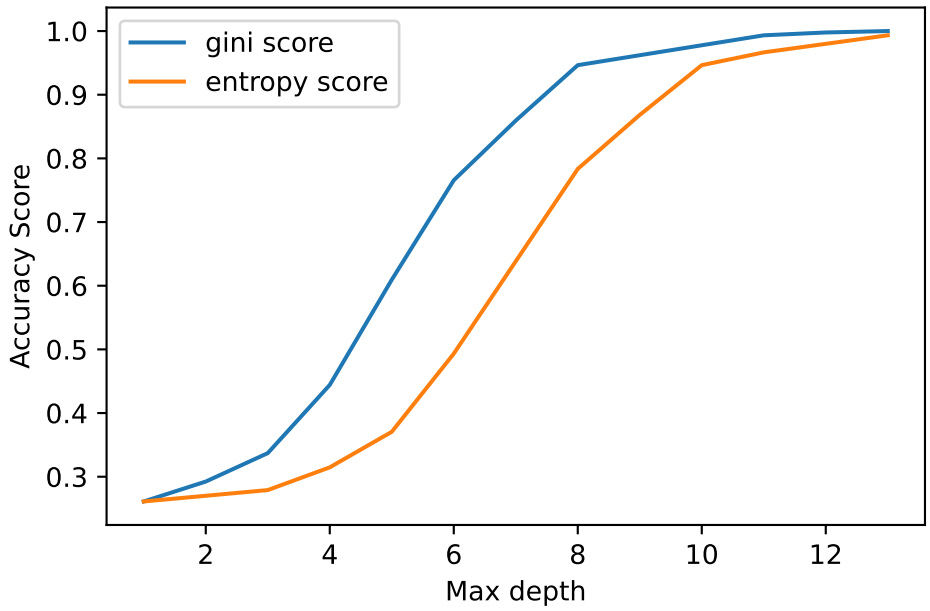
On cherche à tracer les courbes donnant le pourcentage d'erreurs commises en fonction de la profondeur maximale de l'arbre pour

```
np.random.seed(10)

criterions = ["gini", "entropy"]
```

```
depths = range(1, 14)

n = 456
data = rand_checkers(n//4, n//4, n//4, n//4)
X_fit = data[:, :2]
y_fit = data[:, 2]


def fit_all_classifiers(criterions, depths, X_fit, y_fit):
    classifiers = defaultdict(list)

    for (criterion, max_depth) in product(criterions, depths):
        classifier = tree.DecisionTreeClassifier(
            criterion=criterion,
            max_depth=max_depth,
        ).fit(
            X_fit, y_fit
        )

        classifiers[criterion].append(classifier)

    return classifiers

classifiers = fit_all_classifiers(criterions, depths, X_fit, y_fit)

def compute_all_scores(classifiers, X_test, y_test):
    scores = dict()

    for criterion in classifiers.keys():
        scores[criterion] = list(map(
            lambda classifier: classifier.score(X_test, y_test),
            classifiers[criterion]
        ))

    return scores

scores = compute_all_scores(classifiers, X_fit, y_fit)


plt.figure()
```

```
for criterion in scores.keys():
    plt.plot(depths, scores[criterion], label=f"{criterion} score")
    print(f"Scores with {criterion} criterion: {scores[criterion]}")

plt.xlabel('Max depth')
plt.ylabel('Accuracy Score')
plt.legend()
plt.draw()
```

```
Scores with gini criterion: [0.2611607142857143, 0.2924107142857143, 0.33705357142857145, 0.4
Scores with entropy criterion: [0.2611607142857143, 0.2700892857142857, 0.27901785714285715,
```



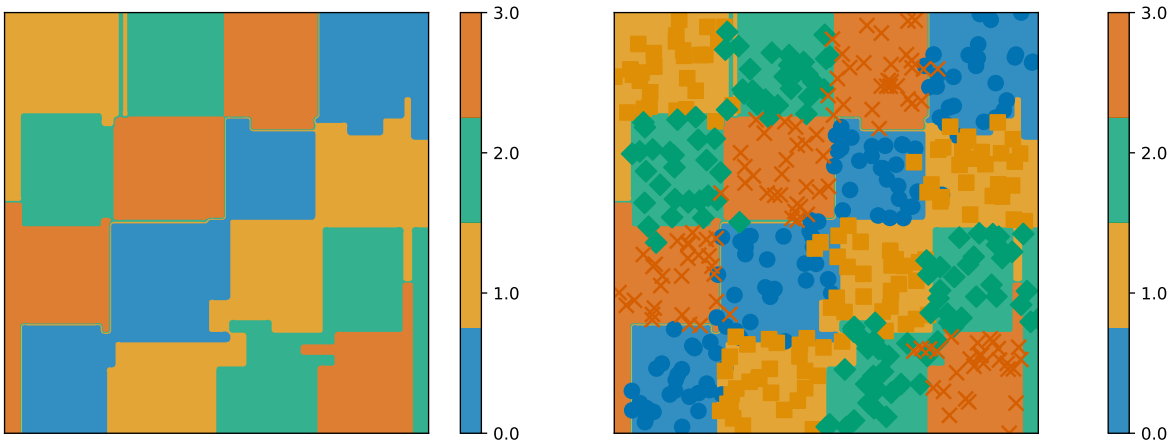
Comme on s'y attendait, la précision des arbres étant testés sur les données d'apprentissage, plus le modèle a de paramètres (ici de profondeur), mieux il va connaître les données d'apprentissage.

Q3)

Comme expliqué à la question précédente, plus la profondeur sera grande, plus l'erreur sera petite. Ainsi avec les profondeurs testées (de 1 à 13), l'arbre de décision qui minimise l'erreur est celui de profondeur 13, cependant dans notre cas le classificateur atteint déjà une précision

parfaite à partir d'une profondeur de 16. Ainsi avec le critère d'entropie et une profondeur de 13 on obtient la classification suivante:

```
best_depth = np.argmax(scores["entropy"]) + 1
assert(best_depth == 13)

best_classifier = classifiers["entropy"][best_depth - 1]

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
frontiere(
    lambda x: best_classifier.predict(x.reshape((1, -1))),
    X_fit, y_fit,
    step=100, samples=False
)

plt.subplot(2, 1, 1)
plot_2d(X_fit, y_fit)
frontiere(
    lambda x: best_classifier.predict(x.reshape((1, -1))),
    X_fit, y_fit,
    step=100, samples=False
)
```



Dans l'ensemble le damier ressemble bien à celui des données, cependant on remarque certains rectangles

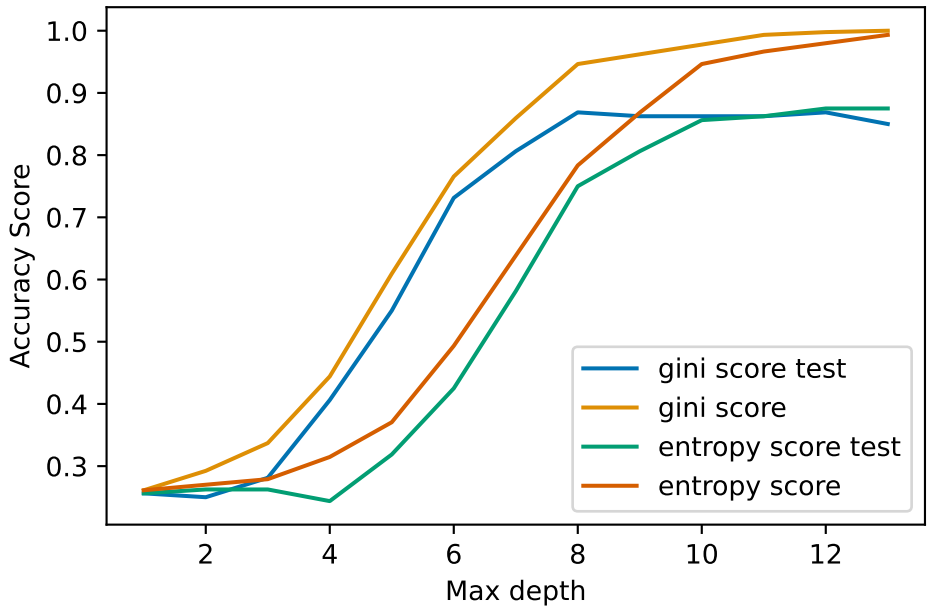
Q4)

TODO: exporter l'arbre

Q5)

```
n_test = 160
data_test = rand_checkers(n_test//4, n_test//4, n_test//4, n_test//4)
X_test = data_test[:, :2]
y_test = data_test[:, 2]

scores_test = compute_all_scores(classifiers, X_test, y_test)

plt.figure()
for criterion in criterions:
    plt.plot(depths, scores_test[criterion], label=f"{criterion} score test")
    print(f"Scores on test data with {criterion} criterion: {scores_test[criterion]}")
    plt.plot(depths, scores[criterion], label=f"{criterion} score ")
    print(f"Scores with {criterion} criterion: {scores[criterion]}")

plt.xlabel('Max depth')
plt.ylabel('Accuracy Score')
plt.legend()
plt.draw()
```

```
Scores on test data with gini criterion: [0.25625, 0.25, 0.28125, 0.40625, 0.55, 0.73125, 0.8
Scores with gini criterion: [0.2611607142857143, 0.2924107142857143, 0.33705357142857145, 0.4
Scores on test data with entropy criterion: [0.25625, 0.2625, 0.2625, 0.24375, 0.31875, 0.425
Scores with entropy criterion: [0.2611607142857143, 0.2700892857142857, 0.27901785714285715,
```

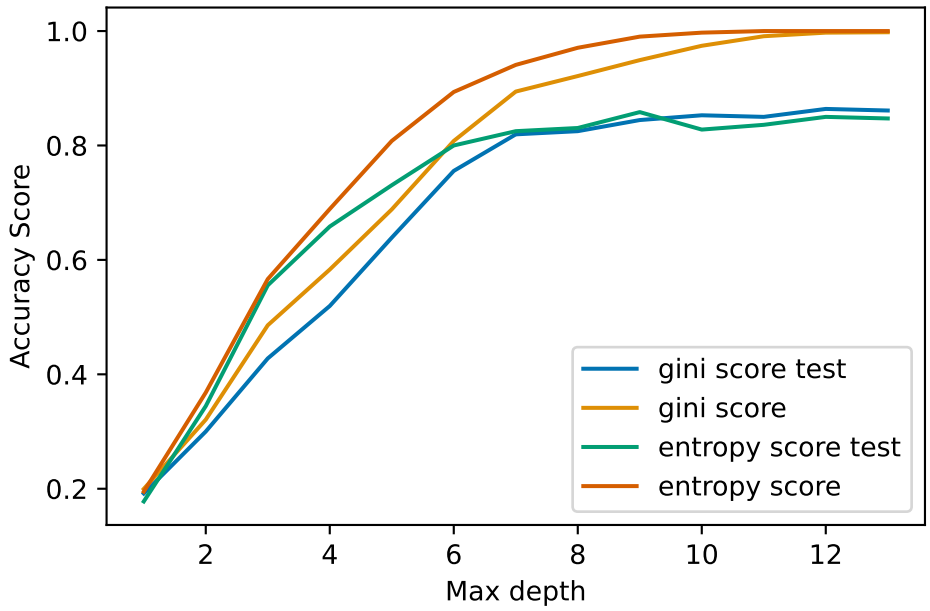


Q6)

```
digits = datasets.load_digits()
X = digits.data
y = digits.target

X_fit, X_test, y_fit, y_test = model_selection.train_test_split(X, y, test_size=0.2, rando

criterions = ["gini", "entropy"]
depths = range(1, 14)

classifiers = fit_all_classifiers(criterions, depths, X_fit, y_fit)

scores = compute_all_scores(classifiers, X_fit, y_fit)
scores_test = compute_all_scores(classifiers, X_test, y_test)

plt.figure()

for criterion in scores.keys():
    plt.plot(depths, scores_test[criterion], label=f"{criterion} score test")
    print(f"Scores on test data with {criterion} criterion: {scores_test[criterion]}")
    plt.plot(depths, scores[criterion], label=f"{criterion} score ")
```

```
    print(f"Scores with {criterion} criterion: {scores[criterion]}")

plt.xlabel('Max depth')
plt.ylabel('Accuracy Score')
plt.legend()
plt.draw()
```

```
Scores on test data with gini criterion: [0.19166666666666668, 0.3, 0.42777777777777776, 0.5
Scores with gini criterion: [0.19972164231036882, 0.32080723729993044, 0.4857341684064022, 0
Scores on test data with entropy criterion: [0.17777777777777778, 0.34444444444444444, 0.555
Scores with entropy criterion: [0.19485038274182323, 0.36812804453723036, 0.5664578983994433
```



on retrouve les mêmes problèmes

Méthodes de choix de paramètres - Sélection de modèle

Q7)