TP2 - Arbres

Alexandre CAPEL

2023-09-28

Le sujet de ce TP porte sur l'étude des arbres de décisions (ou decision tree). Nous allons apprendre à générer nos premiers arbres à partir de données simulées et enregistrées (du package `scikit-learn`) et à faire de la sélection de modèle.

Génération aritficielle de données

Durant ce TP, nous utiliserons des fonctions pour simuler ou afficher des graphiques. Ces derniers ont été empruntés au fichier `tp_arbres_source.py`.

0.1 Classification avec les arbres

0.1.1 Aparté dans le monde de la régression

Toutes les informations concernant la formalisation du problème et sa méthode de résolution (avec l'algorithme CART notamment), sont présenté dans l'énoncé du TP. Cependant, ce dernier nous donne une solution dans le cas de la classification.

En effet, on pourrait se demander quelle mesure d'homogénéité peut être utilisée dans un cadre de régression. Dans le cas de variables continues, un bon indicateur de l'hétérogénéité d'une groupe d'individu serait de calculer leur variance : en effet, cette dernière mesure le fameux "écart à la moyenne" et sera d'autant plus petite que les individus ont des valeurs proches pour cette variables.

A partir de maintenant, on se place dans le paradigme de la classification.

0.1.2 Premières simulations

Avec `scikit-learn`, on peut construire des arbres de décisions grâce au package `tree`. On obtient le classifieur souhaité avec la classe `tree.DecisionTreeClassifier`.

```
from sklearn import tree
```

Faisons nos premières simulations : nous allons utiliser la fonction `rand_checkers` pour construire un échantillon (équilibré) de taille $n = 456$. On peut construire nos arbres selon les deux critères présentés (entropie et indice de Gini) à l'aide du code suivant :

```
from tp_arbres_source import rand_checkers
import numpy as np
from sklearn import tree

# Construction des classifieur
dt_entropy = tree.DecisionTreeClassifier(criterion='entropy')
dt_gini = tree.DecisionTreeClassifier(criterion='gini')

data = rand_checkers(n1=114, n2=114, n3=114, n4=114)
n_samples = len(data)
X = data[:,:2]
Y = np.asarray(data[:,-1], dtype=int)
dt_gini.fit(X, Y)
dt_entropy.fit(X, Y)

print("Gini criterion")
print(dt_gini.get_params())
print(dt_gini.score(X, Y))

print("Entropy criterion")
print(dt_entropy.get_params())
print(dt_entropy.score(X, Y))
```

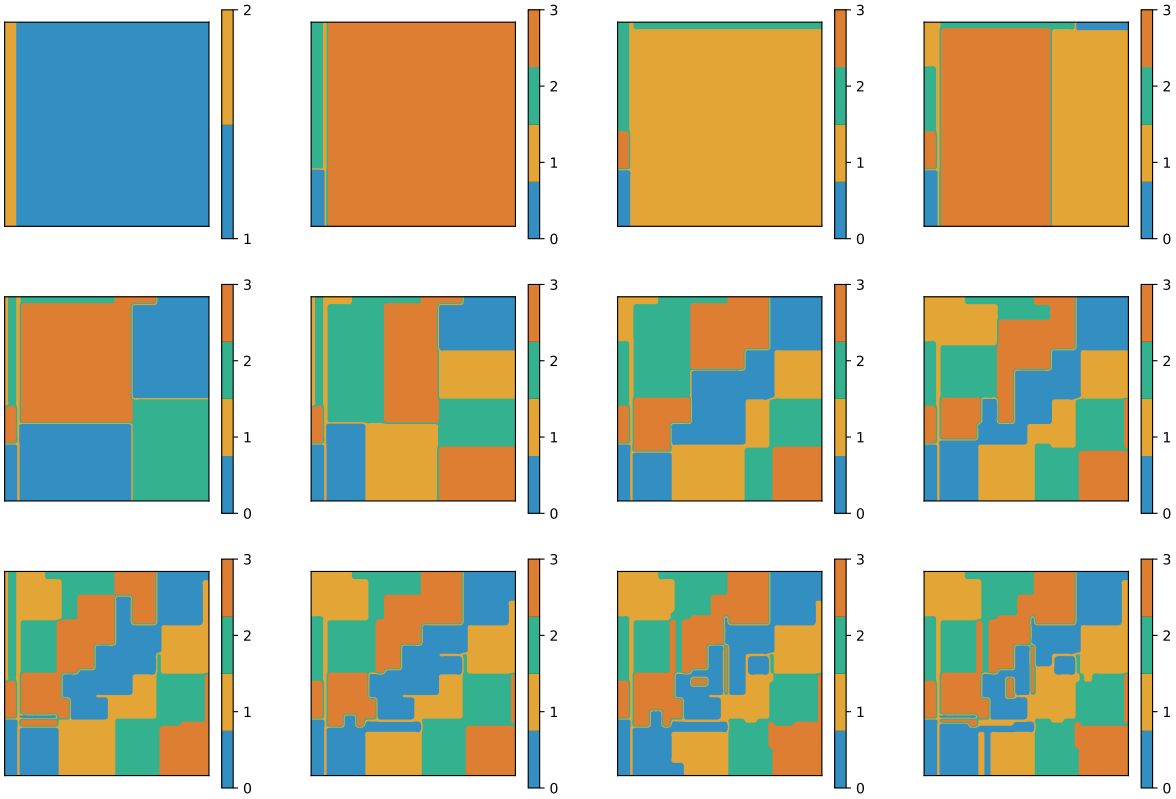
```
Gini criterion
{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_feature
1.0
Entropy criterion
{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'max_fea
1.0
```

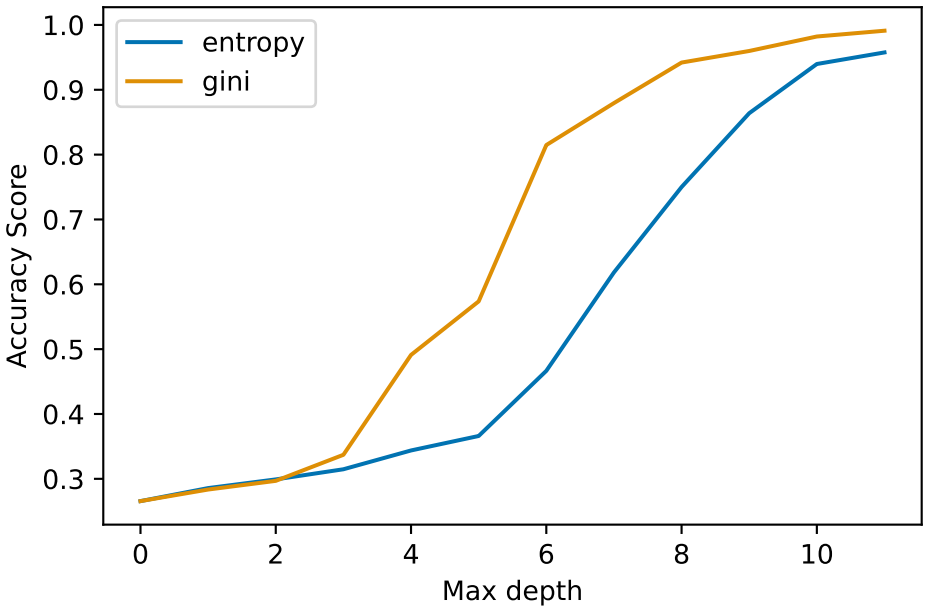
Cette classe a donc pleins d'attributs qui nous permettent d'avoir des information sur notre classifieur : ses paramètres avec `get_params()` ou sa fiabilité avec la fonction `score`.

Amusons nous à changer la profondeur maximale de l'arbre (paramètre `max_depth`), et traçon la courbe d'erreur en fonction de cette dernière. On obtient alors le graphique suivant :

```
Scores with entropy criterion:  [0.265625   0.28571429 0.29910714 0.31473214 0.34375    0.360
 0.46651786 0.61830357 0.75       0.86383929 0.93973214 0.95758929]
```

```
Scores with Gini criterion:  [0.265625   0.28348214 0.296875   0.33705357 0.49107143 0.57366
 0.81473214 0.87946429 0.94196429 0.95982143 0.98214286 0.99107143]
```





0.2 Méthode de choix de paramètres - Sélection de modèle