# Arbres de décisions

## 1 La maraîchere

Dans cet exercice, l'objectif est de prendre en main l'API de `Scikit-learn` pour créer des arbres de décision.

L'idée est de créer un classificateur capable de distinguer une pomme d'une orange en fonction de certains caractéristiques: le poids du fruit et la texture de la peau.

Nous disposons des valeurs suivantes:

| **Poids** | **Texture** | **Étiquette** |
|---|---|---|
| 140g | Lisse | Pomme |
| 130g | Lisse | Pomme |
| 150g | Rugueuse | Orange |
| 170g | Rugueuse | Orange |

Table 1: Exemples d'entraînement

1. Importer le module `tree` de la librairie `sklearn`
2. Déclarer le tableau des caractéristiques
3. Déclarer le tableau des étiquettes
4. Appeler la méthode `tree.DecisionTreeClassifier()` et stocker le résultat dans `clf`
5. Construire l'arbre de décision à partir des données d'entraînement, via la méthode `clf.fit(X,y)` où $X$ représente les données d'entrées et $y$ les étiquettes.
6. Utiliser l'arbre de décision obtenu pour classer le fruit de 160g à la peau rugueuse. Qu'obtenez-vous ?
7. En utilisant le code suivant, visualiser l'arbre construit.

```
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_pdf("fruits.pdf")
```

Que constatez-vous ?

8. L'algorithme de construction de l'arbre de décision n'est pas déterministe. Saisissez différentes valeurs pour le paramètre `random_state` du constructeur du classificateur. Que constatez-vous ?

## 2 Le fleuriste

La librairie `sklearn` propose différents jeux de données. L'un d'entre eux, nommé *Iris*, comprend 50 échantillons de chacune des trois espèces d'iris (Iris setosa, Iris virginica et Iris versicolor). Quatre caractéristiques ont été mesurées à partir de chaque échantillon : la longueur et la largeur des sépales et des pétales, en centimètres.

Sur la base de la combinaison de ces quatre variables, vous devez construire un arbre de décision permet de déterminer l'espèce d'un iris.

1. Commencer par charger le jeu de données:

```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
```

2. Séparer les données en exemples d'entraînements et de tests. Pour cela, il est nécessaire de faire appel à un module supplémentaire:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

```

   Notez qu'il est possible de modifier `random_state` ainsi que la part de données de test à extraire.

3. Construire et entraîner un arbre de décision sur les données d'entraînement (il est pertinent de fixer `random_state` à une valeur arbitraire pour la reproductibilité).

4. Calculer la qualité de prédiction de l'arbre de décision sur les données de tests, via l'API `clf.score(X,y)`

5. Renouveler les opérations en limitant le nombre maximum de feuilles à 3, via l'API `max_leaf_nodes` du constructeur. Que constatez-vous ?

6. Comparer visuellement les deux arbres.

7. Le choix du `random_state` lors de la séparation des données inflence la qualité de prédiction de l'arbre. Faire varier ce paramètre pour observer les différences. Ensuite, utiliser le module `cross_val_score` de la librairie `sklearn.model_selection` pour automatiser le processus.

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(clf, iris.data, iris.target, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
    std() * 2))

```

# 3 La secouriste

Vous devez maintenant traiter du cas des survivants du Titanic. Les données d'entraînements et de tests sont disponibles sur le partage IMT: `https://frama.link/TTNC`. Le mot de passe est "maitredumonde", sans guillemets.

Pour charger les données, vous aurez besoin de la librairie `pandas`:

```
import pandas as pd

df = pd.read_csv('titanic.csv', sep=';')
print(df)
```

Les caractéristiques d'un passager sont les suivante:

- la classe de son ticket
- s'il a survécu (1)
- son nom
- son sexe
- son âge
- le nombre de frères, sœurs, époux, épouse à bord
- le nombre de parents, enfants à bord
- le prix de son billet
- le numéro de sa cabine
- le port d'embarquement
- s'il avait un gilet de sauvetage (1)
- le numéro d'identification du corps
- sa ville origine et de destination

1. Extraire les colonnes pertinentes, en utilisant la commande:

```
df = df[['poids','taille']]
```

2. Convertir, si nécessaire, les chaînes de caractères en entiers:

```
df['taille'] = df['taille'].map({'petit': 0, 'grand': 1})
```

3. Supprimer les données incomplètes:

```
df = df.dropna()
```

4. Construire un arbre de décision, en tâchant de le rendre le plus compact possible, tout en lui assurant une bonne qualité de prédiction sur l'ensemble de test.

5. Afficher la matrice de confusion:

```
from sklearn.metrics import confusion_matrix

cfm = confusion_matrix(y_test, y_predict)
pd.DataFrame(
    cfm,
    columns=['Mort prevue', 'Survie prevue'],
    index=['Mort', 'Survivant']
)
```