TP : Arbres de décision

Dans ce TP, nous allons étudier les arbres de décision en classification puis en régression.

Classification

Premier exemple

Dans cette partie nous allons voir un premier exemple sur la base de données barbecue vue en TD.

```
import pandas as pd
from sklearn import tree

def main():
  data = pd.read_csv("barbecue.csv")
  print(data)
  print(data['barbecue'])
  x_train = data
  y_train = data['barbecue']
  del x_train['barbecue']
  model = tree.DecisionTreeClassifier(criterion='entropy')
  model.fit(x_train, y_train)
  tree.export_graphviz(model, out_file='tree.dot',
  feature_names=['Meteo', 'Amis', 'Vent', 'Jour'])

if __name__ == '__main__':
  main()
```

Base de données `glass`

Nous allons travailler sur la base de données glass. Il s'agit d'apprendre le type d'un verre (batiment, voiture, ...) en fonction de différentes caractéristiques chimiques. La base de données est disponible sur Nexcloud et sa description est fournie sur le site Kaggle : https ://www.kaggle.com/datasets/uciml/glass

A partir de l'exemple fourni précédemment, écrivez un modèle d'arbre de décision pour cette base de données. Attention à ne pas négliger la partie d'analyse et prétraitement des données.

— Calculer les statistiques (moyenne et écart-type) des variables explicatives.
— Combien y a-t-il de type de verre et combien y a-t-il d'exemples de chaque classe ?
— Normalisez les données et séparez le jeu de données en deux : 70% pour l'apprentissage, 30% pour le test.
— Construisez un arbre de décision et entraînez le sur la base d'apprentissage. Une fois l'apprentissage terminé, visualiser l'arbre, soit avec matplotlib en passant par la méthode *plot_tree*, soit avec l'outil graphviz. Par exemple, avec matplotlib :

```
tree.plot_tree(model, filled=True)
```

— Évaluer la performance de votre arbre sur l'ensemble de test en utilisant les métriques suivante : précision, matrice de confusion, rapport de classification
— Changez les valeurs de paramètres *max_depth* et *min_samples_leaf*. Que constatez-vous ?
— Le problème ici étant particulièrement simple, refaites une division apprentissage/test avec 10% des données en apprentissage et 90% test. Calculez le taux d'éléments mal classifiés sur l'ensemble de test. Faites varier (ou mieux, réalisez une recherche par grille avec GridSearchCV) les valeurs des paramètres *max_depth* et *min_samples_leaf* pour mesurer leur impact sur ce score.

Régression

Les arbres de décision peuvent aussi être utilisés pour des problèmes de re- gression. Dans cette seconde partie, nous allons étudier la base de données winequality-red. Il s'agit de prédire la qualité d'un vin en fonction de différentes caractéristiques chimiques. La dernière colonne correspond à la qualité du vin, représentée par une note entière.

Comme pour la première partie, récupérez la base, analysez les données, et effectuez un apprentissage avec les arbres de décision. Vous devez utiliser les bibliothèques suivantes :

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
```