TP1 : Rappel sur sklearn : application sur une base de données déséquilibrée

Objectifs

Dans ce TP, nous allons revoir le fonctionnement de base de la bibliothèque `sklearn`. Nous allons l'appliquer sur la détection de fraudes dans des transactions bancaires.

Analyse des données

Commençons par récupérer la base de données sur la détection de fraude ici. Nous allons la charger dans un dataframe `pandas` à l'aide de la fonction `read_csv`.

Avec la bibliothèque `pandas`, il existe plusieurs fonctions permettant une première analyse simple des données:

- L'attribut `shape` permet de connaitre les dimensions du dataframe.
- La fonction `info` permet d'avoir un résumé rapide des données.
- La fonction `describe` permet d'avoir des statistiques sur différentes tendances des données.
- La fonction `head` permet d'afficher les premières lignes du dataframe.
- La fonction `value_counts` permet de compter le nombre d'occurrences de chaque valeur.

Utiliser ces fonctions pour répondre à ces questions :

- Combien de classes ?
- Combien de caractéristiques descriptives ? De quels types ?
- Combien d'exemples ?
- Combien d'exemples de chaque classe ?
- Comment sont organisés les exemples ?

Séparation des données en bases d'apprentissage et de test

La bibliothèque sklearn fournit la fonction `train_test_split` qui permet de séparer la base. Pour cela, nous allons utiliser `from sklearn.model_selection import train_test_split`. Cette fonction a l'avantage de randomiser l'ensemble avant de faire le split.

Apprentissage et test

Le gros avantage d'utiliser une bibliothèque telle que `sklearn` est que tous les différents algorithmes ont le même fonctionnement.

Pour l'apprentissage il faut utiliser la fonction `classifier.fit(x_train, y_train)` quelque soit l'algorithme d'apprentissage. Pour la prédiction il faut utiliser la fonction `classifier.predict(x_test)`. Pour mesurer les performances, celà dépend si on est en régression ou en classification.

En régression :

- `from sklearn.metrics import mean_absolute_error` pour la MAE,
- `from sklearn.metrics import mean_squared_error` pour la MSE,
- `from sklearn.metrics import r2_score` pour le score r^2

En classification :

- `classifier.score()` pour le score en prédiction,
- `from sklearn.metrics import confusion_matrix` pour la matrice de confusion,
- `from sklearn.metrics import classification_report` pour avoir le score en précision, en recall et le score f_1.

Nous allons tester plusieurs algorithmes d'apprentissage :

- `from sklearn import tree`
- `from sklearn.ensemble import RandomForestClassifier`
- `from sklearn.neighbors import KNeighborsClassifier`
- `from sklearn.neural_network import MLPClassifier`

Effectuer un apprentissage à l'aide de ces algorithmes, mesurer leurs performances grace à `classification_report`.

Certains algorithmes ont un très mauvais recall, pourquoi ?

Correction des données

Pour corriger le déséquilibre présent dans les données, il existe deux méthodes :

- La première consiste à supprimer des données de la classe la plus présente (utilisée lorsqu'on a un grand nombre de données).
- La seconde consiste à dupliquer des données de la classe la moins présente (utilisée lorsqu'on a un petit nombre de données).

Utiliser ces deux méthodes et comparez les.