Approches par modèles de substitution

Le compte-rendu de ce TME est à rendre avant la séance du mardi 21 novembre, sous la forme d'un notebook jupyter.

Exercice 1 Construction de règles contre-factuelles

Pour cet exercice, vous pouvez utiliser soit l'algorithme de construction d'arbres de décision que vous avez réalisé lors du TME précédent, soit utiliser l'algorithme de scikit-learn.

1. Choisir un ensemble de données et le séparer en une base d'apprentissage et une base de test.
2. Construire un arbre de décision avec la base d'apprentissage.
3. Choisir un exemple de la base de test comme exemple de référence $\mathbf{x}$. Le classer avec l'arbre de décision et afficher la règle activée pour son classement.
4. Afficher toutes les règles contre-factuelles r' ainsi que $n_{r'}$ leur nombre de tests invalidés pour $\mathbf{x}$.
5. Proposer une explication pour la classification de $\mathbf{x}$ par l'arbre de décision.
6. Proposer une méthode de votre choix pour présenter à l'utilisateur l'explication fournie à la question précédente.

Exercice 2 Génération de bases d'apprentissage et d'explications

1. En vous inspirant des implémentations que vous avez déjà réalisées, ou de l'approche LORE, écrire une fonction qui, étant donné un classifieur f (celui que vous voulez, appris sur votre base d'apprentissage de l'exercice précédent) et un exemple de référence $\mathbf{x}$, génére une base d'apprentissage pour construire un modèle de substitution.
2. Combiner la fonction précédente avec ce que vous avez écrit dans l'exercice précédent pour écrire un programme de génération d'explications de la classification par f pour les exemples de la base de test.

Exercice 3 Expérimentations

Expérimenter votre programme des exercices précédents sur au moins 2 bases d'exemples de complexité variable (chevauchement des classes, nombre d'attributs, attributs catégoriels,...).

Annexes

Quelques indications pour `scikit-learn`

Pour manipuler des `DecisionTreeClassifier` une fois construits :

```
import sklearn
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
arbre = clf.tree_
```

- `tree.plot_tree`
- `clf.predict`
- `clf.decision_path`
- `arbre.node_count`, `arbre.feature`