Veille algorithmique

Le but de l'algorithme est de trouver si oui ou non il y aura des dégâts après la location d'une maison ou d'un appartement.

La target est donc binaire, on peut parler de classe.

J'ai donc choisi d'utiliser un algorithme de classification, plus précisément un algorithme en random forest. Le RandomForestClassifier peut gérer les données de grandes dimensions, le dataset possède plus d'une centaine de colonnes. En effet, les random forest utilisent plusieurs arbres pour ensuite donner son "verdicte" pour la donnée testée. Ici nous pourrons évaluer nos données de tests si la location est rendue en bon état ou non.

Cet algorithme fonctionne avec les paramètres n_estimators, qui donnent le nombre d'arbre aléatoire qui vont être créés. Ainsi nous aurons n_estimators arbres.

Sur chacun de ces arbres, nous pouvons également régler avec les paramètres max_depth la profondeur maximal que peut avoir nos arbres.

On pourra ensuite entraîner nos arbres avec des données d'entraînement
(80% des données pour conserver 20% des données pour les tests).

Suite à l'entraînement de notre algorithme, nous pourrons évaluer son score avec la méthode cross_val_score() permettant de mesurer l'accuracy de notre algorithme.

On pourra donc visualiser l'accuracy (la précision) de notre algorithme. Mais cela ne sera pas suffisant, nous devrons donc également évaluer notre algorithme avec une matrice de confusion.

Pour cela il est la fonction classification_report permet de faire un rapport sur la précision, le recall, le f1-score sur la prédiction qu'a fait notre algorithme, comparé aux données de test ("réponses").