ARBRES DE DECISION
REGRESSION

TABLE DES MATIERES

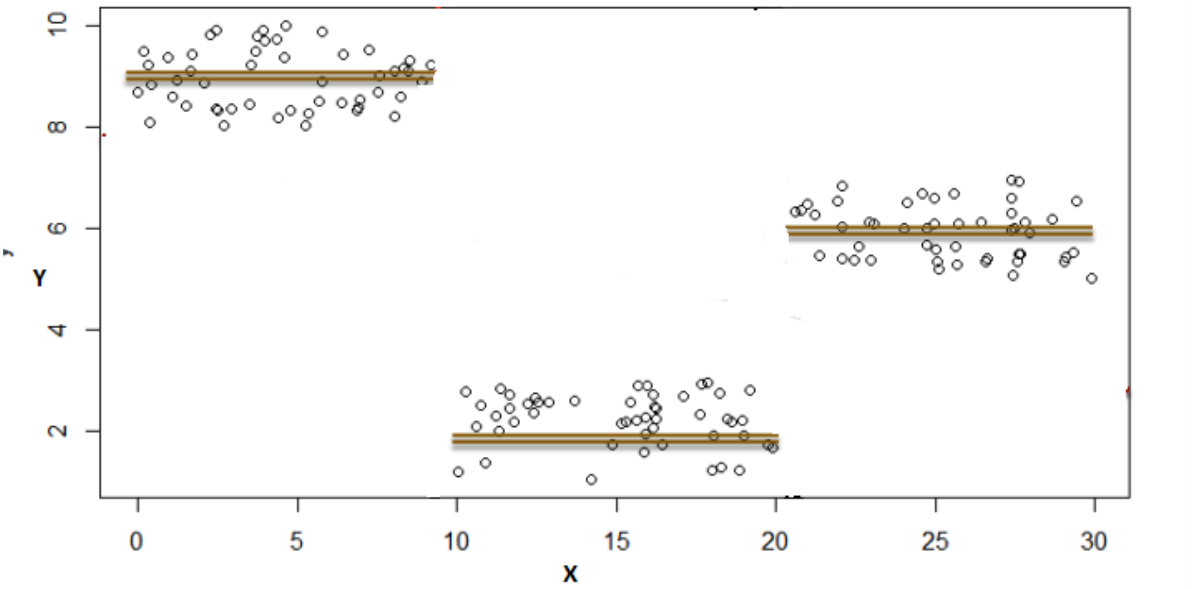
I. INTRODUCTION

Les arbres de décision sont utilisés pour répondre à des problèmes de régression où la valeur de sortie (target) est une variable continue.

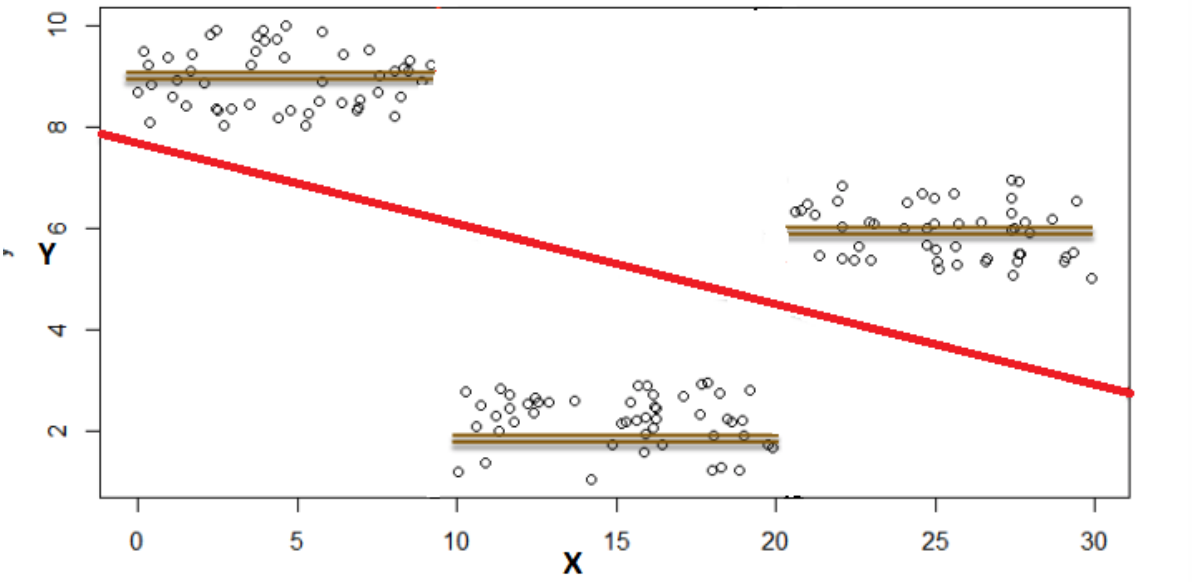
Pour prédire une variable continue, l'étiquette de chaque feuille de l'arbre est la moyenne des valeurs.

Exemple :

Dans la figure ci-dessous, y est la variable à prédire et x est une variable explicative.

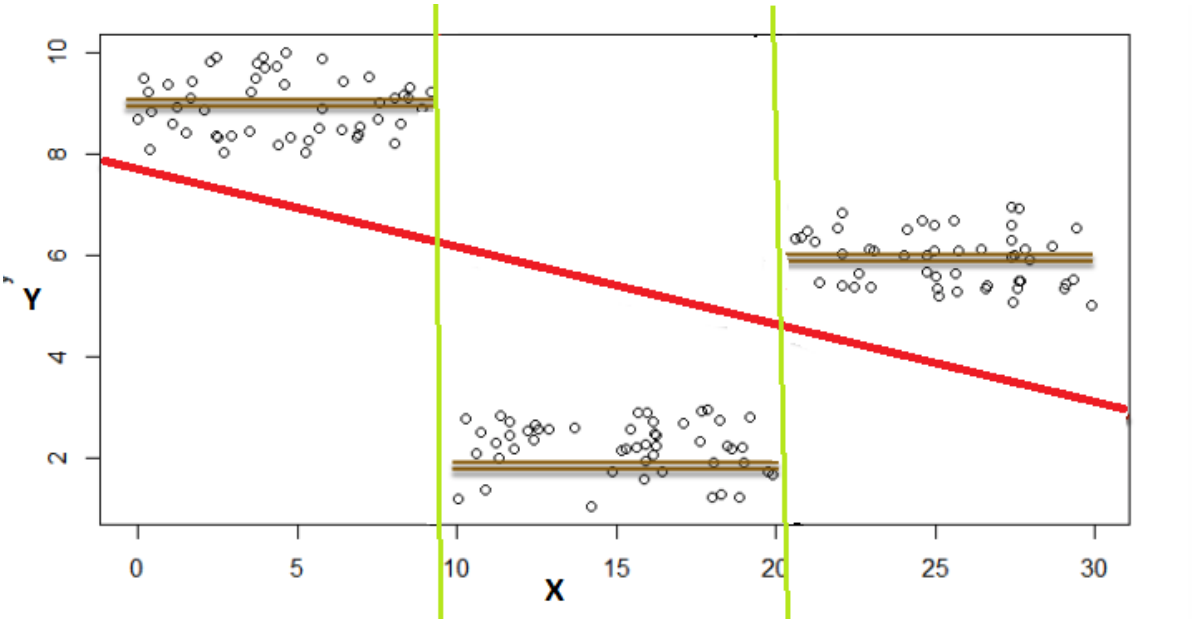


Si on applique une régression linéaire à ce jeu de données, nous obtenons le résultat suivant :



Nous remarquons que le modèle de régression linéaire ne décrit pas pertinemment ce jeu de données.

Cependant, nous remarquons que les données se répartissent en trois groupes différents comme le montre la figure suivant :



Un modèle basé sur un arbre de décision peut s'adapter facilement à ce type de problème.

- $x \leq 10$, la valeur de sortie sera la moyenne de ses valeurs
- $10 < x \leq 20$, la valeur de sortie sera la moyenne de ses valeurs
- $20 < x \leq 30$, la valeur de sortie sera la moyenne de ses valeurs.

II. IMPLEMENTATION

1. SKLEARN

Pour la régression avec les arbres de décision, scikit-learn offre la classe ***DecisionTreeRegressor***.

Le constructeur de la classe DecisionTreeRegressor peut prendre plusieurs paramètres qui influent sur le modèle de régression qui sont listés ici.

Comme pour la classification, la méthode ***fit()*** prend en paramètre X (attributs des observations).

Attention : les y ne sont pas des étiquettes de classes mais des valeurs réelles.

Decision Tree Regression in 6 Steps with Python

2. R

Pour construire un arbre de régression avec R, il faut utiliser le package ***rpart***.

Il est possible d'afficher l'arbre de décision avec R en utilisant la commande ***prp***.

Pour diviser le jeu de données en jeu de test et jeu d'entrainement, il faut utiliser la librairie **caTools**.

III. PRATIQUER

1. PYTHON

CHARGEMENT DES DONNEES

1. Charger la base de données **Diabetes** du module sklearn.datasets.

2. Faire une partition aléatoire en partie apprentissage et partie test (70% apprentissage, 30% test)

CONSTRUCTION DE L'ARBRE DE DECISION

1. Construire un modèle d'arbre de régression sur cette base en utilisant les paramètres par défaut de la classe.
2. Visualiser l'arbre de régression.
3. Evaluer les performances de ce modèle.
4. Changer la valeur du paramètre max_depth.
    a. Que se passe-t-il si on prend une valeur trop grande ?
    b. Trop petite ?
    c. Conclure

2. R

CHARGEMENT DES DONNEES

1. Charger le jeu de données Position_Salaries.
2. Faire une partition aléatoire en partie apprentissage et partie test (70% apprentissage, 30% test)

CONSTRUCTION DE L'ARBRE DE DECISION

5. Construire un modèle d'arbre de régression sur cette base pour prédire la variable Salary.
6. Visualiser l'arbre de régression.
7. Evaluer les performances de ce modèle.

IV. VALIDATION CROISEE

1. PRINCIPE

La validation croisée, ou cross validation, est une procédure d'échantillonnage utilisée pour évaluer les modèles d'apprentissage automatique sur un échantillon de données limité.

Cette procédure est également appelée k-fold cross-validation où k fait référence au nombre de groupes dans lesquels un set de données doit être divisé.

La validation croisée est principalement utilisée en apprentissage automatique pour estimer la performance d'un modèle sur des données « invisibles ». En effet, la méthode utilise un échantillon limité afin d'estimer comment le modèle devrait fonctionner en général lorsqu'il est utilisé pour faire des prédictions sur des données non utilisées pendant la phase d'apprentissage.

Cette méthode est largement utilisée vu qu'elle est simple à comprendre et qu'elle aboutit généralement à une estimation non biaisée des performances des modèles contrairement à une simple répartition jeu d'apprentissage et jeu de test.

2. PROCEDURE

La procédure générale de la validation croisée est comme suit :

1. Mélanger le jeu de données aléatoirement.
2. Diviser le jeu de données en k groupes.
3. Pour chaque groupe :
    a. Considérer ce groupe comme jeu de test.
    b. Considérer les autres groupes comme un jeu d'apprentissage.

   c. Entrainer le modèle sur le jeu d'apprentissage et l'évaluer avec le jeu de test.
   d. Garder en mémoire le score d'évaluation
4. Résumer les performances du modèle

Il est important de noter que pour chaque observation de l'échantillon de données est affectée à un groupe individuel et reste dans ce groupe pendant la durée de la procédure. Cela signifie que chaque échantillon a la possibilité d'être utilisé dans le jeu de test une fois et k-1 fois dans le jeu d'entraînement.

Les résultats d'un k-fold cross-validation sont souvent résumés avec la moyenne des scores des performances du modèle. Il est recommandé d'inclure une mesure de la variance comme l'écart-type.

3. CONFIGURATION DU K

Le choix de la valeur de k est primordial. Une valeur mal choisie peut entrainer une idée erronée sur les performances du modèle comme par exemple un score avec une variance élevée ou encore une surestimation des performances.

Pour choisir la valeur de k trois stratégies sont communément utilisées :

1. Une valeur représentative : la valeur de k est choisie de telle sorte que chaque jeu d'entrainement et de test soit suffisamment grand pour être statistiquement représentatif de l'ensemble des données.
2. $k = 10$: La valeur de k est fixée à 10. Il s'agit d'une valeur qui a été trouvée pas l'expérimentation pour aboutir généralement à une bonne estimation des performances du modèle avec un biais faible et une petite variance.
3. $k = n$: la valeur de k est fixée à n, où n est la taille de l'ensemble du jeu de donnée pour donner à chaque échantillon d'être utilisé dans le jeu de test. Cette approche est appelée leave-one-out- cross-validation.

V. OPTIMISATION DES HYPER PARAMETRES

Le modèle d'arbre de régression est paramétrable. En effet, le choix des sous-arbre (criterion), la profondeur de l'arbre (max_depth), le nombre maximum de features à utiliser (max_features) ou encore le diviseur (splitteur) peut influer sur les performances du modèle.

Il faut alors sélectionner les paramètres qui optimise les performances du modèle de régression.

Sklearn offre une classe qui permet de tester toutes les combinaisons pour trouver un optimum parmi les hyper paramètres : GridSearchCV. Elle utilise la validation croisée.

L'attribut best_estimator_ permet de récupérer le meilleur arbre de décision.

VI. PRATIQUER

Répondre aux questions 1 et 2 en utilisant sklearn ET R.

1. CHARGEMENT DES DONNEES

1. Charger la base de données **Boston** du module sklearn.datasets.
2. Faire une partition aléatoire en partie apprentissage et partie test (70% apprentissage, 30% test)

2. CONSTRUCTION DE L'ARBRE DE DECISION

1. Construire un modèle d'arbre de régression sur cette base en utilisant les paramètres par défaut de la classe.

2. Visualiser l'arbre de régression.
3. Evaluer les performances de ce modèle.

3. OPTIMISATION DES HYPER PARAMETRES

1. Trouver les hyper paramètres *criterion* et *max-depth* optimaux en utilisant *GridSearchCV*
2. Interpréter

VII. POUR LA CULTURE

How to optimize hyper parameters of a Decision Tree model using Grid Search

A Guide to Machine Learning in R for Beginners: Decision Trees

Decision Tree Regression in 6 Steps with Python