Classification supervisée & régression non-linéaire : arbre décisionnel, forêt aléatoire et méthode d'ensemble

1 Objectifs

- Classification supervisée : on dispose d'un entraînement dont les catégories (classes) sont connues. La classification peut-être binaire (2 classes, e.g. défectueux ou non, tumeur bénigne/maligne, spam/no spam, …) ou multiclasses.
- Quelques classifieurs particuliers : arbre décisionnel, apprentissage d'ensemble
- Découvrir que classification et régression ne sont pas des notions si différentes : on peut aussi concevoir un système pour la prédiction d'une catégorie (classification) que d'une valeur (prédiction/régression)

2 Arbre décisionnel & classification

Cette approche consiste à exprimer la classification comme une suite de tests conduisant à une partition de l'espace caractéristique (i.e. espace des données) en sous-régions homogènes (i.e. contenant des objets de la même classe) : voir illustration 1. A noter que cette technique est également utilisable pour de la régression (i.e. on cherche à déterminer des valeurs et non des « labels »). Nous nous limitons aux labels.

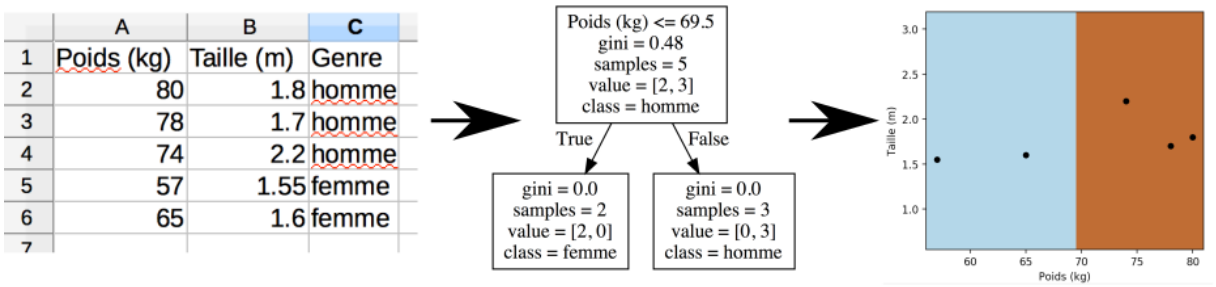

Illustration 1 Exemple : Données, Arbre décisionnel et partition de l'espace caractéristique.

La classification: La classification consiste à appliquer ces tests aux données inconnues, en partant de la racine : on va ainsi « descendre » jusqu'à une feuille dont l'étiquette indiquera la classe de l'objet inconnu.

Apprentissage (construction de l'arbre décisionnel) :

- L'arbre est construit en partant d'un nœud racine, qui produit 2 (ou plus) nœuds, et ainsi de suite jusqu'à obtenir des feuilles (nœuds terminaux) associées à une seule classe. On peut décider d'arrêter la production de feuilles selon un critère donné, notamment pour éviter le sur-apprentissage : par exemple, la profondeur de l'arbre (voir illustrations 2 et 3), le nombre d'échantillons minimum ([samples]) pour un nœud,...
- Pour construire un nœud, on s'intéresse à la « pureté » (homogénéité) des feuilles: chacune d'elle doit être plus homogène que le noeud qui la précède. Une feuille « recevant » que des échantillons ([samples]) d'une seule classe (voir [value]) est homogène (ou « pure »). Pour qualifier la pureté, il existe plusieurs métriques : par exemple l'indice de « Gini » (sans frotter), utilisé par l'algorithme de « CART » et l'entropie probabiliste, utilisé par l'algorithme « C4.5 ». Nous nous limitons à l'indice/impureté de « gini », à valeur dans [0,1] :

$$G(t)=1-\sum_j p(j|t)^2$$

où p(j|t) est la proportion d'observation de la classe j parmi toutes les observations ([samples]) au nœud t. Dans notre premier exemple, pour « t=0 » (racine) : $G(0)=1-(2/5)^2-(3/5)^2 =0.48$. Pour « t=1-» (« 1-1 » car bas-gauche) : $G(1-)=1-(2/2)^2-(0/2)^2 =0$. Ce nœud est pur (homogène) !

Pour un nœud donné, les deux paramètres sont la variable [X] (e.g. poids/taille) et le test [t] (e.g. seuil). Leur choix est guidé par un critère à minimiser. Dans le cas de l'algorithme CART, ce critère est :

$$I_G = \frac{n_{t-}}{n} G(t-) + \frac{n_{t+}}{n} G(t+)$$

où nt- et nt+ sont, dans ce cas binaire, le nombre d'observation pour les nœuds fils gauche et droit (e.g. 2 et 3), pour lesquels on calcule G(t-) et G(t+). « n » est le nombre d'observation au nœud considéré (e.g. 5). Dans l'exemple (illustration 1), les paramètres (X=« poids »,seuil=69.5) permettent de minimiser ce critère :

I_G (X=« poids »,t=69.5)=2/5*(1-(2/2)²-(0/2)²)+3/5*(1-(0/3)²-(3/3)²)=0. Ceci est à comparer à un autre jeu de paramètres qui produirait des feuilles moins « pures » : I_Gini (X=« poids »,seuil=61)=1/5*(1-(1/1)²-(0/1)²)+4/5*(1-(1/4)²-(3/4)²)=0.3.

Question : Que vaut I_G pour (X= « taille »,t=1.6) dans le cas de l'illustration 1 ?

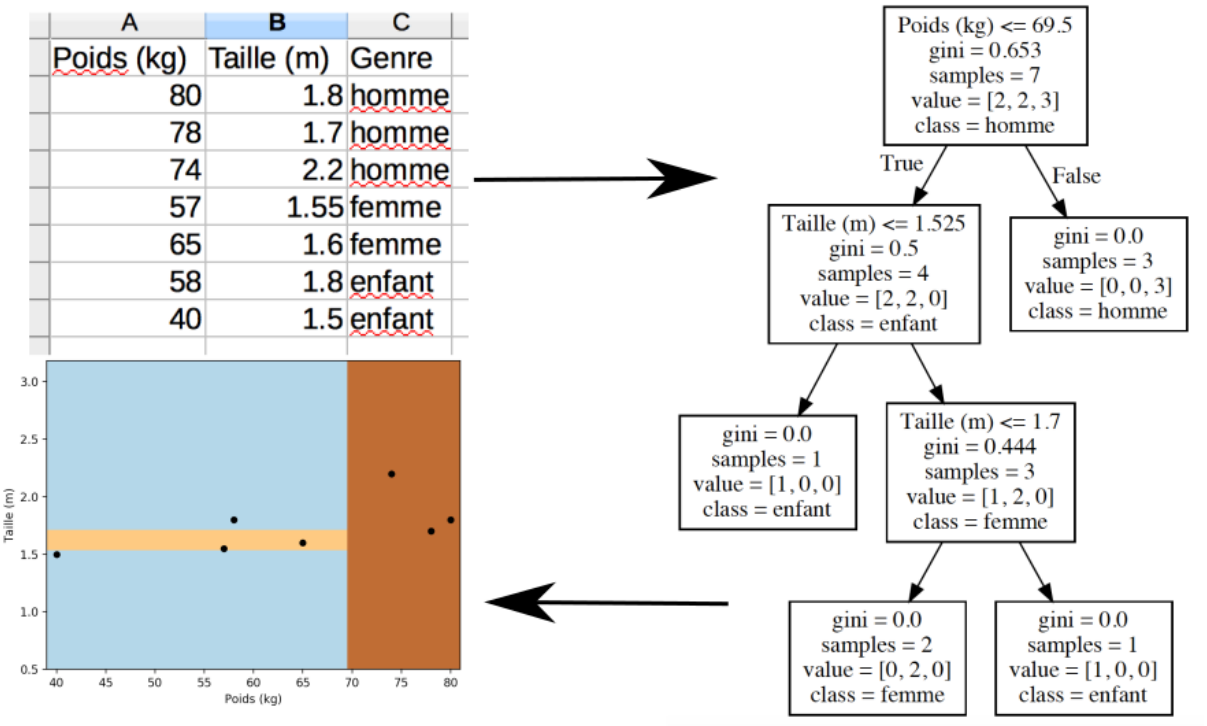

Illustration 2 Cas 3 classes sans erreur.

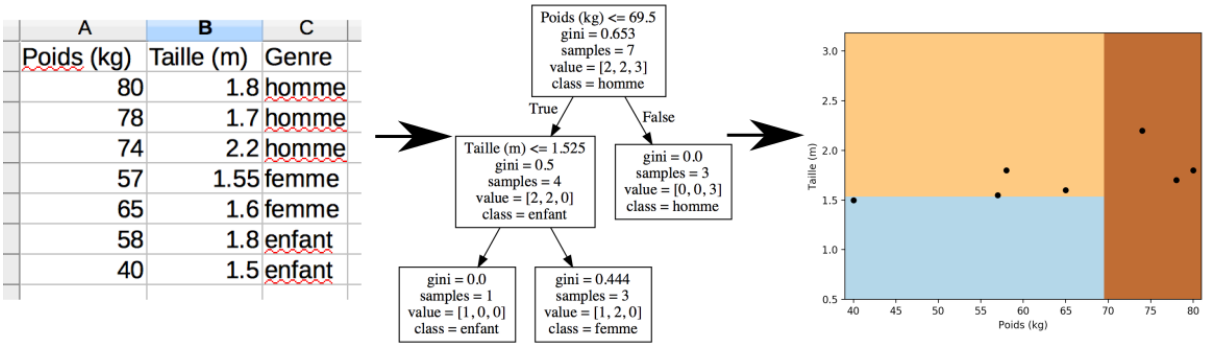

Illustration 3: Cas 3 classes en limitant la profondeur à 2 → classification imparfaite (taux de 0.85).

Travail à faire : Ecrivez le script testant l'efficacité de cette technique ([sklearn.tree.DecisionTreeClassifier]) dans le cas des iris : essayez en considérant les 3 classes et seulement les descripteurs associés à la taille du sépale (largeur et hauteur), puis du pétale (largeur et hauteur), et enfin des 2 informations. On utilisera les paramètres par défaut du classifieur.

Question : Quel est le taux de bonne classification pour chaque cas de figure ?

Question : Quel est le taux de bonne classification pour une profondeur d'arbre de 2 seulement (en utilisant les informations sépale et pétale ? Quelle était la profondeur initiale ?

Remarque : on peut exporter l'arbre dans un format graphique (nécessite une librairie particulière) :

```
import graphviz
dot_data = tree.export_graphviz(clf)
graph = graphviz.Source(dot_data)
graph.render("HF")
```

Dans le cas « imparfait » (e.g. illustration 3), la dernière feuille, conduit à la classe « femme », avec 2 chances sur 3 selon l'apprentissage. Le vote majoritaire classifierait donc un échantillon de type [poids=60,taille=1.55] comme étant de la classe femme. On peut néanmoins également considérer des probabilités : le cas [poids=60,taille=1.55] conduit à « femme » (avec une probabilité de 0.666) ou « enfant » (avec une probabilité de 0.333) mais pas « homme » (probabilité de 0).

Question : En adaptant le script précédent pour les données « homme-femme-enfants », en considérant une profondeur de 2, vérifiez que vous obtenez bien ces probabilités (méthode [predict_proba]). Etant donné que peu de données sont disponibles, on ne séparera pas en données entraînement/test : on utilisera l'intégralité des données pour entraîner l'arbre.

Bilan sur cette méthode (non exhaustif):

- **Quelques avantages:** simplicité de compréhension, très rapide pour la classification
- **Quelques inconvénients:** apprentissage long (volume > quelques milliers), et risque de sur-apprentissage.

3 Apprentissage d'ensemble : combinaison de classifieurs

Chaque classifieur a ses avantages et ses inconvénients. Une approche moderne consiste à combiner différents classifieurs : on va ainsi considérer un « ensemble » de classifieurs, d'où le terme « apprentissage d'ensemble ». On peut combiner des classifieurs hétérogènes ou des classifieurs de même type (e.g. forêt aléatoire combinant des arbres décisionnels). On peut également, lors de la combinaison de classifieurs, jouer sur l'entrainement : par exemple tirer aléatoirement des sous-ensembles du jeu d'entraînement pour entraîner chaque classifieur. Dans ce document, nous nous limitons à deux cas de figure :

- Classification par vote (classifieurs hétérogènes)
- Forêt aléatoire (classifieurs de même type mais on joue sur l'entraînement)

1 Classifieurs par vote

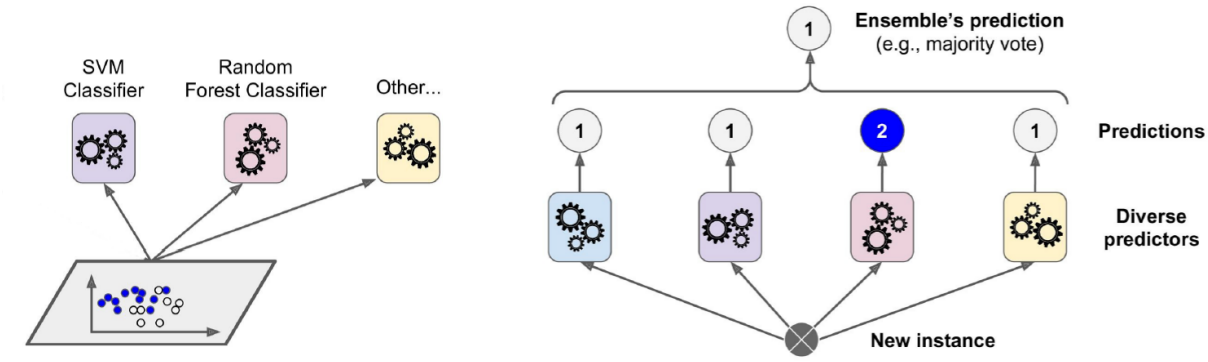

Illustration 4: Apprentissage d'ensemble: classifieur par vote (apprentissage à gauche, classification à droite). Source: [Machine Learning avec scikit-learn]

En utilisant la classe [sklearn.ensemble.VotingClassifier] (avec vote majoritaire : « hard »), combiner un classifieur de type 'arbre de décision' (avec profondeur de 2) et de type 'plus proche voisin' (essayer avec 3 puis 5 voisins) pour classer les iris (3 classes), en n'utilisant que l'information relative à la sépale. On consultera la documentation en ligne. A noter que l'on pourrait considérer un vote « souple » (soft), qui raisonne sur la base des probabilités d'appartenance estimées par chaque classifieur : une condition est que les classifieurs fournissent des probabilités !

Question : Quel est le taux de classification de chaque classifieur? De leur combinaison (en «hard» et «soft»)?

2 Forêt aléatoire (ou forêt d'arbres décisionnels)

Ce type de classifieur combine plusieurs arbres de décision, chacun étant entraîné sur un sous-ensemble de l'ensemble d'entraînement (tirage aléatoire). Pour augmenter la diversité, au lieu de chercher la meilleure variable pour partager les nœuds sur l'ensemble des variables, cette meilleure variable est cherchée sur un sous-ensemble des variables. Ce classifieur est fourni par la classe [sklearn.ensemble.RandomForestClassifier] de scikit-learn.

Question : En ne considérant que l'information « sépale », quel est le taux de classification avec une forêt aléatoire, par exemple pour 10 estimateurs (à comparer avec un simple arbre de décision) ?

4 Régression non-linéaire et arbre décisionnel

Les figures 1 et 2 montrent le cas d'une sortie (Output) reliée à l'entrée (Input) par une relation non linéaire (régression non-linéaire), en s'appuyant sur un arbre décisionnel.

	Input	Output
0	0.0	101.44636852574773
1	1.0	478.8669313293561
2	2.0	970.3199020853608
3	3.0	594.5073103806795
4	4.0	904.0471065754723
5	5.0	312.7326183495886
6	6.0	208.40111679967984
7	7.0	919.5123952014224
8	8.0	408.4757607247182
9	9.0	188.92737012315865
10	10.0	603.1647718972051
11	11.0	245.82690128612606
12	12.0	520.2307106509456

Illustration 1: Données synthétiques (synthetic.csv)

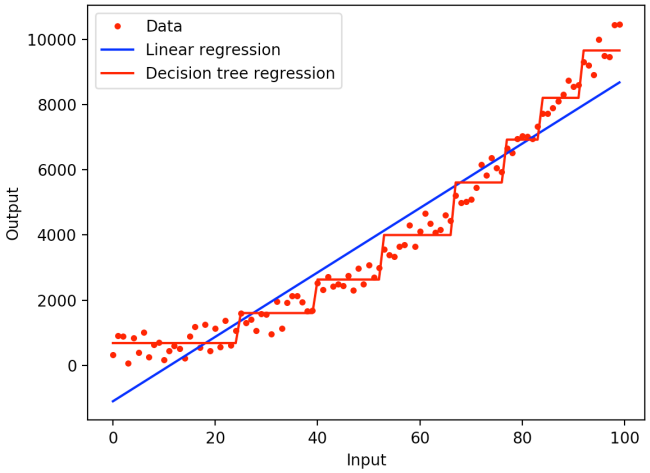

Illustration 2: Modèle de prédiction : par régression linéaire et par régression à l'aide d'un arbre décisionnel

1 Etude des données synthétiques

En utilisant les données synthétiques (« synthetic.csv »), on va essayer de construire un modèle prédictif basé sur la régression linéaire (classe [sklearn.linear_model.LinearRegression]) et un autre modèle basé sur un arbre décisionnel (classe [sklearn.tree.DecisionTreeRegressor]).

Question : Calculer et donner les erreurs de prédiction commises sur les données d'entraînement et sur les données de test (20 %), dans le cas de la régression linéaire et dans le cas de l'utilisation de l'arbre décisionnel. Pour mesurer l'erreur, on utilisera le « root mean squared error ». On utilisera les paramètres par défaut pour l'arbre décisionnel. Comment expliquez-vous les résultats ?

Question : Essayer d'améliorer la régression non-linéaire par arbre décisionnel en jouant sur la profondeur de l'arbre. Tracer la courbe de l'évolution des erreurs (entrainement et test) en fonction de ce paramètre. Quelle est la valeur optimale de ce paramètre (i.e. pour laquelle l'erreur sur les données de test est la plus faible) ?

Question : l'arbre décisionnel est-il plus performant que la régression linéaire ? De combien est le bénéfice (en terme d'erreur) ?

2 Prédiction du prix de maison

Répendre l'étude sur l'estimation des prix des biens immobiliers (« housing.csv ») - en considérant les différentes transformations préliminaires (ajout d'un attribut, retrait de l'attribut catégoriel, normalisation,...) :

Question 1: Calculer et donner les erreurs de prédiction commises sur les données d'entraînement et sur les données de test (20 %), dans le cas de la régression linéaire et dans le cas de l'utilisation de l'arbre décisionnel. On utilisera les paramètres par défaut pour l'arbre décisionnel.

Question 2: Tracer la courbe de l'évolution des erreurs (entrainement et test) en fonction de la profondeur. Quelle est la valeur optimale de ce paramètre ?