Universite de Strasbourg

Intelligence artificielle

Licence 3 - UFR Mathématique - Informatique / Printemps 2021

Projet – détection d'outliers par arbres de décisions

Instructions :
- Vous pouvez travailler en binôme (pas de trinôme et plus tolérés)
- Vous devrez rendre votre code et un compte rendu de 2 pages maximum au format `pdf` avant le 23 avril 2021 à 20h CET, dans l'espace Moodle prévu à cet effet (1 rendu par groupe)

Barème (donné à titre indicatif)
- Préparation des données : 3 points
- Questions sur les métriques : 4 points
- Arbre réduit à une feuille : 4 points *(+ 1 point bonus)*
- Arbre superficiel : 4 points
- Arbre généralisé : 5 points

1 Introduction

Les outliers sont les données aberrantes présentes dans le jeu de données, c'est-à-dire les données dont la valeur s'éloigne significativement de la tendance majoritaire obtenue à partir des autres données (voir figure 1).

En data science, les outliers biaisent défavorablement les modèles s'ils ne sont pas pris en compte. Une manière (radicale !) de les gérer consiste à les supprimer du jeu de données pour ne travailler qu'avec des valeurs régulières, aussi appelées inliers.

Dans ce projet, on se propose de détecter les outliers avec une méthode non supervisée. Une vérité terrain est fournie mais cette dernière ne sera utilisée que pour l'évaluation.

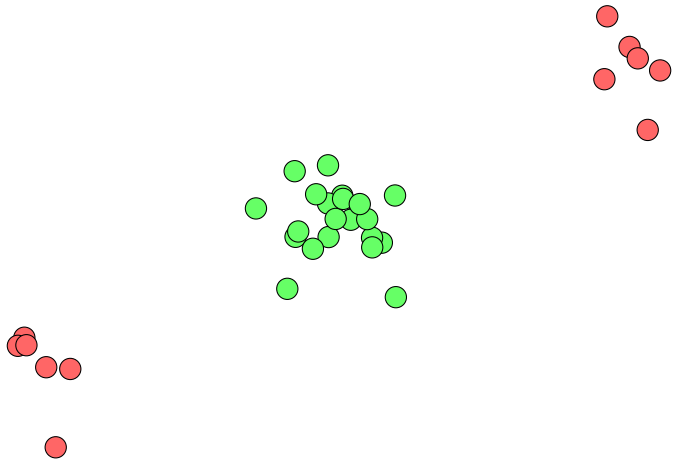


FIGURE 1 – Exemples de données. Les outliers sont représentés en rouge, les inliers en vert.

2 Données

Pour ce projet, vous utiliserez les données présentes dans `data.csv`. Dans ce fichier, chaque ligne décrit un exemple. Les deux premières colonnes correspondent aux attributs de l'exemple en question tandis que la troisième et dernière renseigne la vérité terrain (0 : inlier ; 1 : outlier).

Travail à faire

1. Combien y a-t-il de données dans chacune des classes ?
2. Écrivez une fonction pour visualiser ces données (en utilisant `seaborn` ou autre). Joignez une capture d'écran du résultat dans votre rapport.
3. Décrivez brièvement la forme des données.

3 Évaluation

À chaque fois qu'il sera question d'évaluer un modèle, ou encore de comparer plusieurs modèles, on s'appuiera sur certains indicateurs déduits de la matrice de confusion. De tels indicateurs, il en existe une myriade [1]. On choisit donc de se concentrer sur les trois indicateurs suivants :

— L'exactitude pondérée (*weighted accuracy*)
— La précision (*precision*)
— Le rappel (*recall*)

Concrètement, en définissant la matrice de confusion par

$$\begin{pmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{pmatrix} = \begin{pmatrix} \text{inliers prédits comme inliers} & \text{inliers prédits comme outliers} \\ \text{outliers prédits comme inliers} & \text{outliers prédits comme outliers} \end{pmatrix}$$

on a

$$\begin{aligned} \text{exactitude pondérée} &= \frac{\frac{TN}{TN+FP} + \frac{TP}{FN+TP}}{2} \\ \text{précision} &= \frac{TP}{TP+FP} \\ \text{rappel} &= \frac{TP}{FN+TP} \end{aligned}$$

Questions

Une autre métrique couramment utilisée est l'exactitude (*accuracy*), définie par

$$\text{exactitude} = \frac{TN+TP}{TN+FP+FN+TP}$$

On considère un modèle dont la matrice de confusion est

$$\begin{pmatrix} 1000 & 2 \\ 30 & 5 \end{pmatrix}$$

1. À la seule lecture des coefficients, le modèle associé vous semble-t-il bon ? Justifiez.
2. Calculez puis comparez l'exactitude et l'exactitude pondérée de cette matrice. Commentez.
3. Pourquoi l'exactitude donne un score aussi bon ?
4. En déduire pourquoi elle n'est pas pertinente dans notre cas.

4 Algorithmes

Pour la suite, l'idée générale est de construire et d'évaluer différents types d'arbres de décisions pour déterminer quels sont les outliers parmi les données.

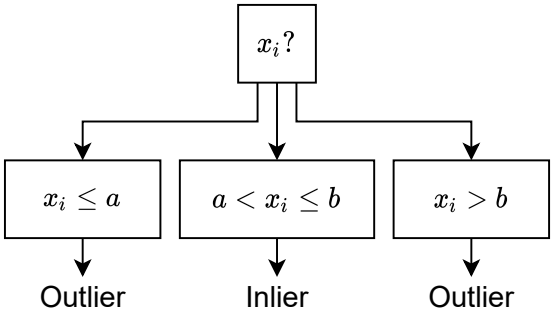
1. Pour en avoir un aperçu, il suffit de jeter un œil sur le tableau de droite de la page `https://en.wikipedia.org/wiki/Confusion_matrix`.

Les arbres auront tous la particularité d'être ternaires, c'est-à-dire que chacun de leurs nœuds présentera trois issues.

Les structures de données à utiliser sont les mêmes pour l'ensemble des questions. Seul l'algorithme de construction de l'arbre change.

4.1 Arbre réduit à 1 feuille (classe *DecisionLeaf*)

Pour commencer, on va considérer le cas le plus simple : celui d'un arbre composé d'une unique feuille. Concrètement, l'arbre produit ressemblera à ceci



Ce modèle comporte 3 paramètres, qu'il faut donc déterminer :

— L'attribut utilisé (l'indice i sur le schéma)

— Les deux seuils a et b

Pour choisir l'attribut à utiliser, on ne peut pas utiliser l'entropie (qui nécessite d'être dans une configuration d'apprentissage supervisé), il nous faut un autre critère. Instinctivement, on cherche l'axe sur lequel on peut trouver les plus grands écarts entre les données, les plus grands « vides ». Une bonne indication de ce phénomène se retrouve à travers l'écart-type, qui mesure la disparité entre données. On retient donc l'attribut qui présente le plus grand écart-type. L'écart-type de l'attribut x_i est noté σ_{x_i}.

Quant aux seuils, on les calcule par clustering. On commence par extraire l'attribut d'intérêt de chaque donnée. Ensuite, on réalise un k-means sur ces extractions, avec $k = 2$. Contrairement à l'algorithme original, les centroïdes de départ ne sont pas choisis au hasard mais initialisés aux deux valeurs extrêmes : la plus petite et la plus grande. Une fois la convergence atteinte, on récupère les centroïdes finaux, qui deviennent les seuils de décision.

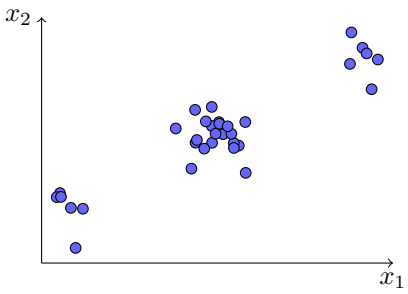
La construction de ce modèle est résumée dans la figure 2.

Travail à faire

1. Définissez une structure de données adaptée pour représenter une telle feuille.
2. Implémentez l'algorithme de construction de ce modèle. Pour le calcul de l'écart-type, vous pouvez utiliser `numpy.std`. Pour k-means, vous pouvez

soit vous tourner vers le module dédié de scikit-learn [2], soit recoder la méthode par vous-même. **Si vous optez pour cette seconde option et que vous proposez votre propre implémentation (sans bugs !) de k-means, vous aurez 1 point bonus sur la note finale.**

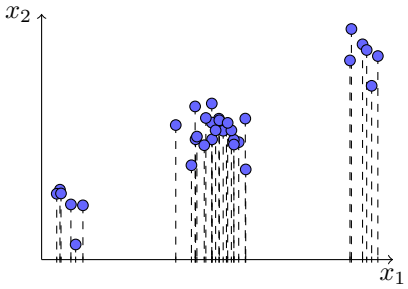
3. Évaluez le modèle appris.



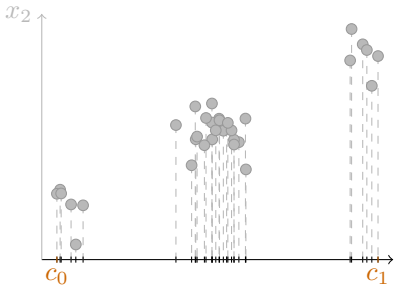
(a) Situation initiale

$$\sigma_{x_1} = 0.5897454498724192$$
$$\sigma_{x_2} = 0.3162639157410175$$
$$\sigma_{x_1} > \sigma_{x_2} \implies \text{on choisit } x_1$$

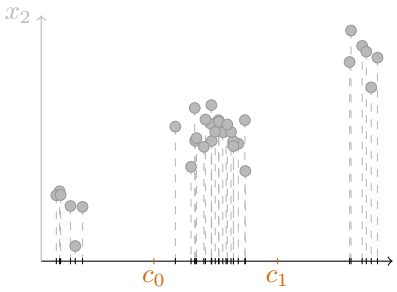
(b) Calcul des écart-types



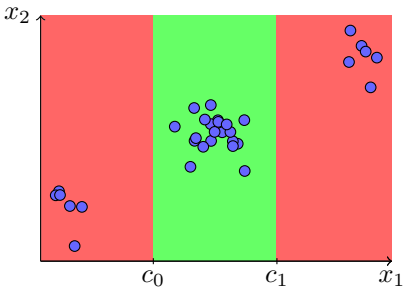
(c) Extraction de l'attribut



(d) Initialisation du k-means



(e) Résultat du k-means



(f) Régions de classification

FIGURE 2 – Illustration de l'algorithme de construction d'une feuille de décision.

2. Pour la doc, voir `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`. **Attention, pensez bien à renseigner les centroïdes initiaux via la paramètre *init* du constructeur, sinon vous n'aurez pas le bon résultat !**

4.2 Arbre superficiel

Une fois qu'on est capable de construire une feuille de décision, on peut chercher à obtenir des modèles plus expressifs en créant des arbres. Dans un premier temps, on procède de façon analogue à l'algorithme ID3 vu en cours. Comme pour la construction d'une feuille, chaque nœud aura trois issues. Trois types d'issues seront possibles :

- un sous-arbre, tant qu'il reste au moins 2 attributs non utilisés et au moins 4 exemples (comme pour ID3, un attribut ne peut servir qu'une seule fois).
- une feuille de décision, s'il reste exactement 1 attribut à utiliser et au moins 4 exemples.
- une décision directe, s'il ne reste que 3 exemples ou moins à traiter. Dans ce cas, on considère que le k-means n'est plus pertinent pour déterminer des seuils et on arrête la récursivité. La décision prise dépendra du chemin pris depuis le nœud parent : si on vient de la branche centrale, on renverra « inlier », si on vient d'une des deux branches lattérales, on renverra « outlier ».

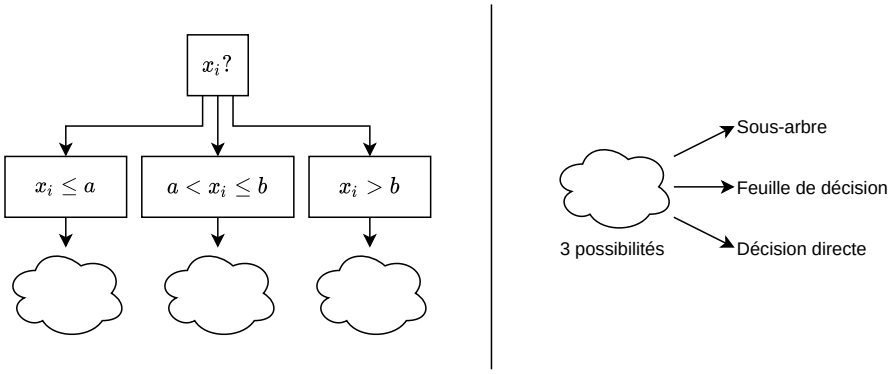


FIGURE 3 – Schéma d'un arbre de décision

Algorithme 1 : Algorithme de construction de l'arbre de décision superficiel. Le paramètre « central » de buildDecisionTree indique si le noeud en cours de création provient de la branche centrale de son parent ou non. Par convention, central vaut **True** pour la racine. La fonction « getSplitParameters » correspond à la détermination de l'attribut d'intérêt et au calcul des seuils vus en section 4.1.

```
Data : Un ensemble de données D de taille Card(D). Chaque donnée a
       n attributs, numérotés de 1 à n
Result : Un arbre de décision
function buildDecisionTree(D, central, attribIdx):
    if Card(D) ≥ 4 then
        if Card(attribIdx) ≥ 2 then
            currentAttrib, a, b := getSplitParameters(D) ;
            D_l, D_m, D_r := {x | x_i ≤ a}, {x | a < x_i ≤ b}, {x | b < x_i} ;
            attribIdx := attribIdx \ {currentAttrib} ;
            L := buildDecisionTree(D_l, False, attribIdx) ;
            M := buildDecisionTree(D_m, True, attribIdx) ;
            R := buildDecisionTree(D_r, False, attribIdx) ;
            return Node(currentAttrib, a, b, L, M, R) ;
        else
            return DecisionLeaf(D, attribIdx) ;
        end
    end
    else if Card(D) < 4 then
        if central == True then
            return DirectDecision(outlier=False)
        else
            return DirectDecision(outlier=True)
        end
    end
end
buildDecisionTree(D, True, {1; ...; n});
```

Travail à faire

Mêmes questions qu'à la section 4.1 :

1. Précisez les structures de données utilisées
2. Implémentez la méthode proposée
3. Évaluez votre modèle et discutez les résultats.

4.3 Arbre généralisé

La principale limitation de la méthode précédente était qu'on ne pouvait sélectionner un attribut qu'une seule fois. En supprimant cette contrainte, on supprime par la même occasion un des critères d'arrêt de la récursivité puisque la hauteur de l'arbre ne pouvait pas dépasser le nombre d'attributs. Comme il n'est pas forcément souhaitable que la hauteur des arbres soit exclusivement déterminée par le nombre d'exemples (risque de sur-apprentissage), on ajoute explicitement une hauteur maximale que l'arbre ne peut pas dépasser. Lorsque cette profondeur est atteinte, on complète l'arbre avec une feuille de décision s'il reste suffisamment d'exemples. Dans le cas contraire, on opte pour une décision directe.

Travail à faire

1. Adapter l'algorithme 1 pour pouvoir réutiliser le même attribut plusieurs fois (et ajoutez le nouveau critère d'arrêt).
2. Évaluez le modèle avec des hauteurs maximales de 1 à 4 inclus et réalisez un tableau reprenant l'ensemble des métriques demandées : exactitude pondérée, précision et rappel. Vérifiez que le cas $h = 1$ correspond à une feuille de décision (vous devriez obtenir exactement le même modèle qu'à la section 4.1 : mêmes seuils, même indice d'attribut, mêmes métriques, etc.).
3. Quelle hauteur vous semble donner les meilleurs résultats ?