# Projet Intelligence Artificielle

**Détection d'outliers par arbres de décisions**


**Siham JANATI**
**Abdelheq DELMI BOURAS**

**L3-S6 Informatique**
**Université de Strasbourg**

**Mars 2021**

# 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.
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 nonsupervisée. Une vérité terrain est fournie mais cette dernière ne sera utilisée que pour l'évaluation.

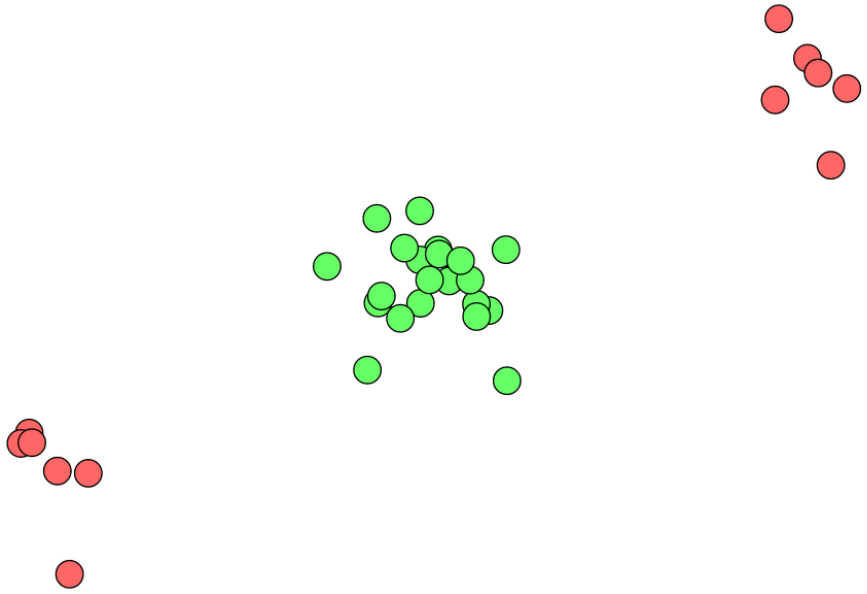


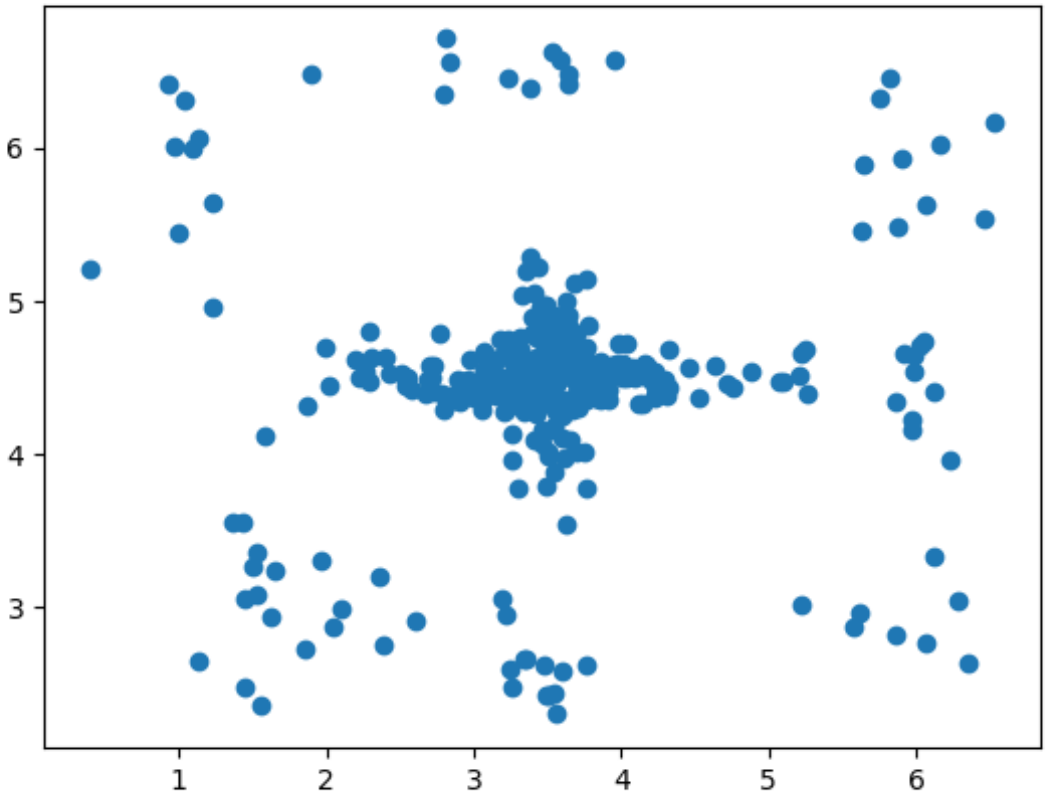
FIGURE 1 – Exemples de données. Les outliers sont représentés en rouge, les inliers en vert.

# 2 Préparation des données :

## 2.1 Recensement des données

| | |
|---|---|
| 0 (inlier) | 250 |
| 1 (outlier) | 80 |

## 2.2 Visualisation des données




## 2.3 description des données :

dans la figure présentée ci-dessus, on aperçoit de manière approximative, que les inliers se présentent dans la section x[1,7 :5,2] et y[3,4 :5,2]

# 3 Évaluation

## 3.1

Le modèle semble bien détecter les inliers ( 1000/1002 ), mais pas les outliers ( 5/35 ). Comme le but est de détecter les outliers, le modèle ne semble pas si bon.

## 3.2

$$exactitude = \frac{1000+2}{1000+2+30+5} = 0.97$$

$$exactitudepondre = \frac{\frac{5}{35}+\frac{1000}{1002}}{2} = 0.57$$

### 3.3

Parce que le modèle détecte bien les négatifs mais pas les positifs. Et comme les données dont nous disposons comprennent surtout des négatifs (inliers), l'exactitude semblera bonne.

### 3.4

Elle n'est pas pertinente car elle ne renseigne pas sur la capacité du modèle à détecter les outliers, ce qui est notre but.

# 4 Algorithmes

## 4.1 Arbre Réduit à une feuille

— **DecisionLeaf** : une feuille de decision avec comme attributs
  — **a** `higher_split`
  — **b** `lower_split`
  — **currentAttrib** `l'attribut d'intérêt`
  — **D** `notre dataset`

```
class DecisionLeaf():
      def __init__(self, data=None, currentAttrib=None):
           self.currentAttrib = currentAttrib
           self.D = data
           clusters, cluster_centers = Functions.Functions.k_means(self.D.T[self.currentAttrib])
           self.a = cluster_centers[0]
           self.b = cluster_centers[1]
```

## 4.2 Arbre Artificiel

### 4.2.1 Structures

Pour notre arbre superficiel, il nous faut créer 2 classes supplémentaires, qui sont implémentées dans le fichier `'Node.py'`

— **DecisionDirect** : Une classe de decision directe avec comme attributs
  — **D** `notre dataset`
  — **nb** `le nombre des exmples reste à traiters`
  — **outier** `type Boolean`

```
class DirectDecision:
    def __init__(self, D, nb, outlier):
        self.outlier = outlier
        self.nb = nb
        self.D = D
```

— **Node** : Une classe pour l'arbre artificiel, avec comme attributs
  — **a** `higher_split`
  — **b** `lower_split`
  — **R** `sous-arbre/branche droit`
  — **L** `sous-arbre/branche gauche`
  — **C** `sous-arbre/branche du centre`

```
class Node:
    def __init__(self, currentAttrib = None, a = None, b = None, L = None, M = None, R = None):
        self.currentAttrib = currentAttrib
        self.a = a
        self.b = b
        self.R = R
        self.L = L
        self.M = M
```

### 4.2.2 Évaluation de l'arbre artificiel

Feuille de Décision

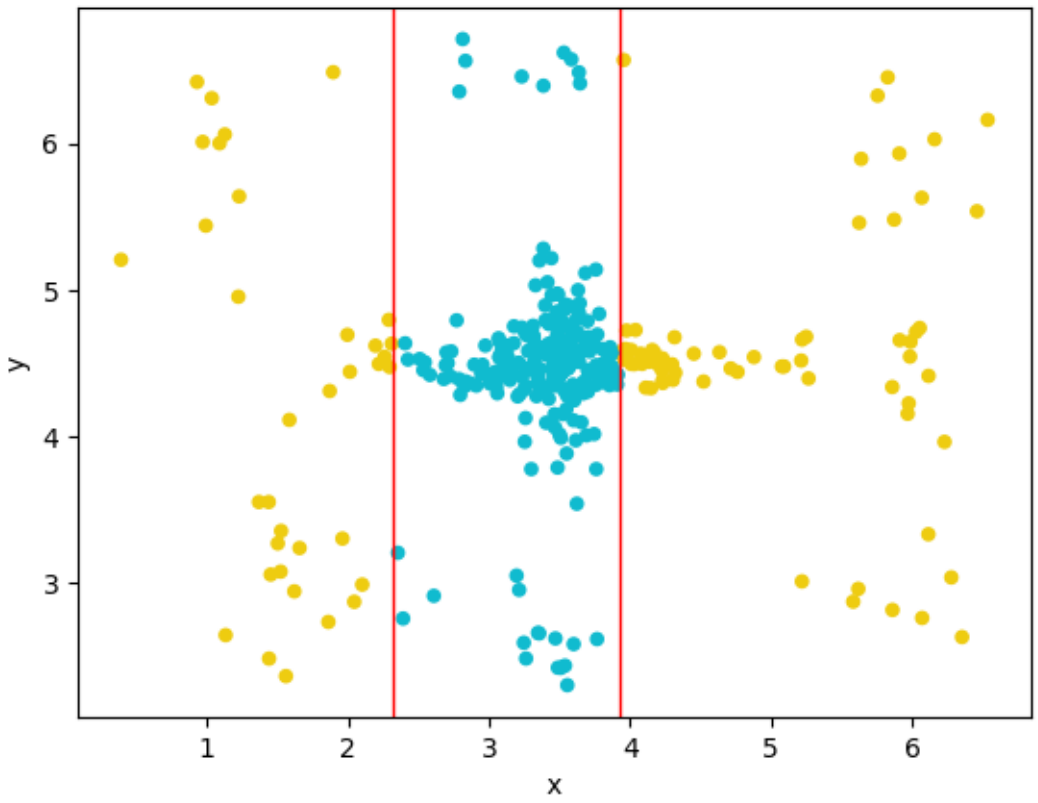



FIGURE 1 – Decision Leaf

```
Exactitude: 0.7818181818181819
Exactitude pondérée: 0.74975
Précision: 0.5392156862745098
Rappel: 0.6875
```

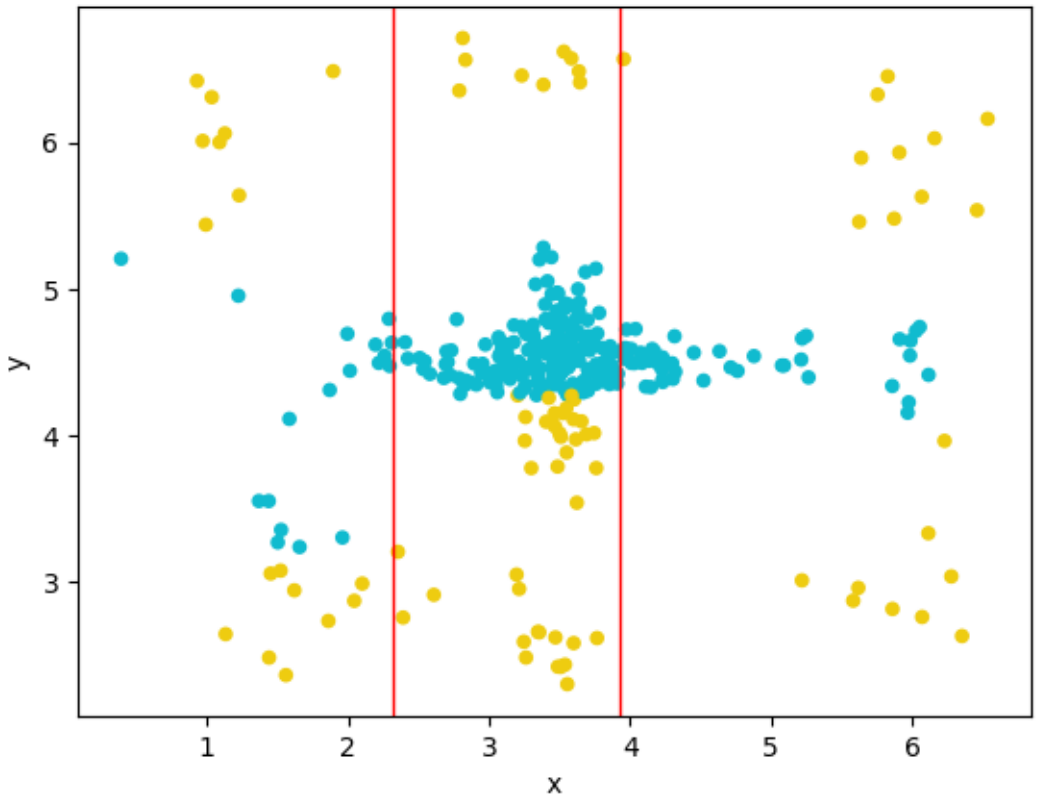



FIGURE 2 – build Decision Tree

---

Arbre de décision :

```
Exactitude: 0.8393939393939394
Exactitude pondérée: 0.8005
Précision: 0.651685393258427
Rappel: 0.725
```

---

**Commentaires :**
Les résultats se sont bien améliorés de la première évaluation avec un **arbre réduit** à une feuille, avec un rappel de `0.76`, le modèle **arbre artificiel** a réussi a bien détecter les (in/out)liere à 80%.

# 5 Arbre Généralisé

## 5.1 Adaptation de l'algorithme

## 5.2 Évaluation :

Feuille de décision :

| exactitude pondérée | Précision | Rappel |
|---|---|---|
| 0.74975 | 0.5392156862745098 | 0.6875 |

Arbre Superficiel :

| exactitude pondérée | Précision | Rappel |
|---|---|---|
| 0.8005 | 0.651685393258427 | 0.725 |

Arbre Généralisé :

| Hauteur | exactitude pondérée | Précision | Rappel |
|---|---|---|---|
| 1 | 0.74975 | 0.5392156862745098 | 0.6875 |
| 2 | 0.804 | 0.5714285714285714 | 0.8 |
| 3 | 0.5714999999999999 | 0.2987012987012987 | 0.575 |
| 4 | 0.5175 | 0.2541436464088398 | 0.575 |

## 5.3 Meilleure hauteur!?

Une hauteur de 2 donne les meilleurs résultats. Mais celle-ci n'est pas meilleure en précision que les résultats de l'arbre superficiel de la question 4-2, elle lui est égale en exactitude pondérée et est meilleure en rappel. Donc elle est moins précise dans la détection des outliers (plus de faux positifs), par contre elle détectera plus d'outliers que l'arbre superficiel qui en manquera certains. Et ceci est tout à fait normal, en améliorant le rappel on perdra un peu de précision.