



RAPPORT DE PROJET CASSIOPEE

présenté par

"Victor Gertner et Thomas Vignon"

Projet effectué entre Janvier 2023 et Juin 2023

Sujet de la mission :

"Classification non supervisée"

Tuteur école : "Marc Castella"

Table des matières

1	Presentation du Projet	2
2	Methode RANSAC	3
2.1	Presentation générale	3
2.2	Algorithme	3
2.3	Implémentation de l'algorithme	4
2.4	Limites de RANSAC	5
2.4.1	Choix de d	5
2.4.2	Mix de loi	6
2.5	Avantages et inconvénients	7
3	Méthode ICE	7
3.1	Présentation générale de la méthode ICE	7
3.2	Présentation théorique de la méthode ICE	7
4	Lien théorique entre Ransac et ICE	9
4.1	Modèle linéaire, bruit uniforme	9
5	Chaine de Markov	14
6	Synthèse de notre projet	14
6.1	Gestion de projet	14
6.2	Apport académique	15

1 Présentation du Projet

Le projet s'inscrit dans le cadre de la classification non supervisée. Dans ce cadre, on trouve des méthodes tels que l'algorithme des k-means ou des estimations de modèles de mélange de lois par EM (Expectation-Maximization). Ici, nous nous intéresserons à séparer des données issues d'une loi de probabilité absolument continue et d'une loi singulière. Un exemple de données issues d'un tel mélange de lois est donné par les points de la figure 1.

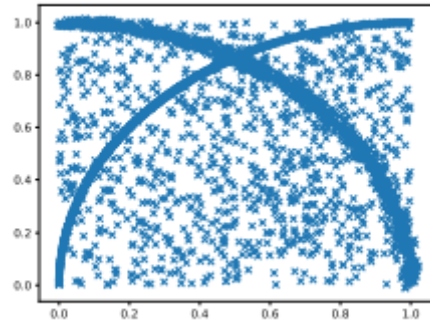


FIGURE 1 – Exemple d'un mélange de trois lois (singulière, singulière+bruit, uniforme)

Ainsi, voici le travail proposé par l'encadrant Marc Castella :

- **recherche bibliographique sur les méthodes de classification non supervisées,**
- **étude de la méthode dite «RANSAC»**
- **implantation des méthodes sous Python,**
- **test et comparaison d'algorithmes,**
- **restitution des résultats (rapport et codes de démonstration).**

2 Methode RANSAC

2.1 Presentation générale

RANSAC ou **RANdom SAmple Consensus** est une méthode itérative qui permet d'estimer un modèle mathématique à partir d'un jeu de données qui contient des outliers. Ainsi, l'hypothèse est que notre jeu de données contient des outliers et des inliers qui eux caractérisent le modèle. De plus, on fait l'hypothèse qu'un modèle existe pour expliquer ces données avant d'appliquer l'algorithme.

Son algorithme tente d'identifier les outliers du jeu de données afin d'estimer un modèle sans ceux-ci. C'est un algorithme non-déterministe dans le sens où il produit un modèle mathématique qui convient à nos données avec une certaine probabilité, et plus on laisse l'algorithme itérer, plus la probabilité augmente.

2.2 Algorithme

Algorithm 1 PseudoCode de l'algorithme RANSAC

Require: data - jeu de données

modele - un modèle qui peut être ajusté à des données

n - le nombre minimum de données nécessaires pour ajuster le modèle

k - le nombre d'itérations de l'algorithme

t - une valeur seuil pour déterminer si une donnée correspond à un modèle

d - le nombre de données minimal prises en compte dans modele pour qu'il soit acceptable

$i \leftarrow 0$

$bestModel \leftarrow null$

$bestPoints \leftarrow null$

$bestError \leftarrow inf$

while $i \leq k$ **do**

$randomPoints \leftarrow n$ valeurs au hasard dans data

$possibleModel \leftarrow modele(randomPoints)$

$possiblePoints \leftarrow randomPoints$

for point **not in** randomPoints **do**

if point s'ajuste à $PossibleModel$ avec une erreur inférieure à t **then**

 Ajouter point à possiblePoints

end if

end for

if $|possiblePoints| > d$ **then**

$possibleModel \leftarrow modele(possiblePoints)$

$error \leftarrow$ erreur de possibleModel

if $error < bestError$ **then**

$bestModel \leftarrow possibleModel$

$bestPoints \leftarrow possiblePoints$

$bestError \leftarrow error$

end if

end if

$i \leftarrow i + 1$

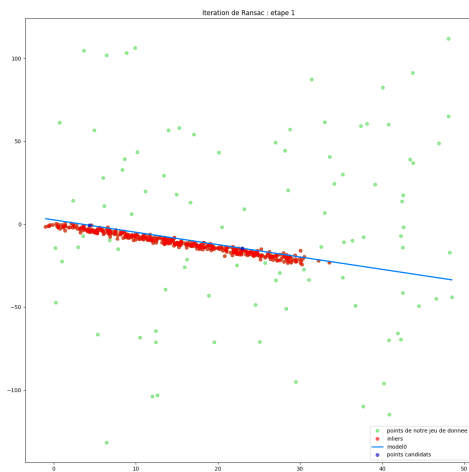
end while

return bestModel; bestPoints; bestError

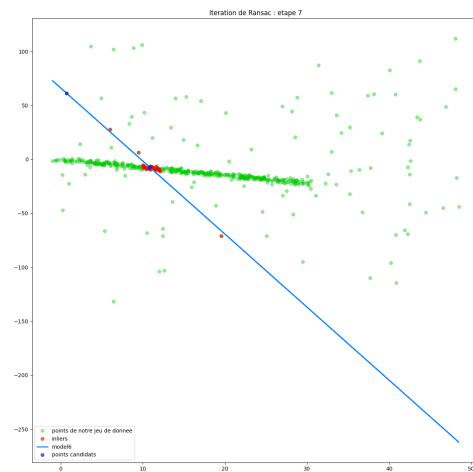
2.3 Implémentation de l'algorithme

Afin de bien comprendre le fonctionnement de RANSAC, nous avons décidé d'implémenter RANSAC en Python.

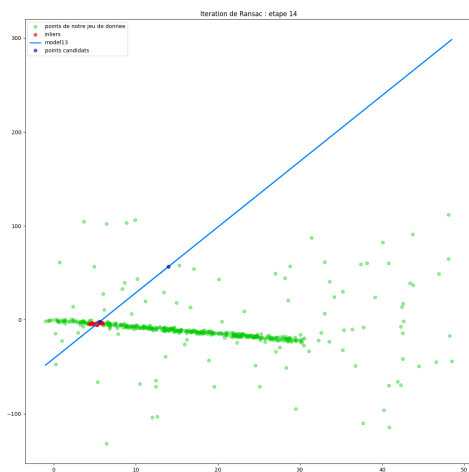
Ainsi, grâce à notre programme, nous pouvons comparer notre implémentation, celle de RANSAC classique et une simple régression linéaire.



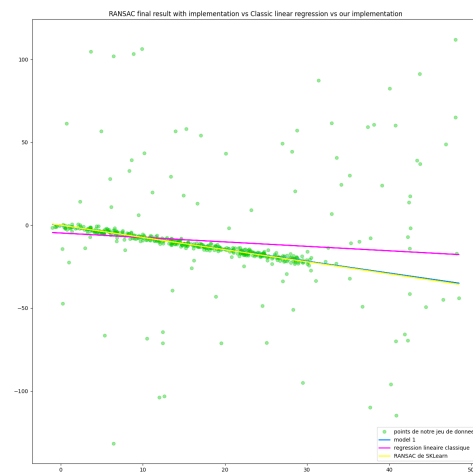
(a) 1ere étape



(b) 7eme étape



(c) 14 eme étape



(d) Resultat Final

Ainsi, on se rend compte que notre modèle se rapproche de la méthode RANSAC implémentée par SKLearn et qu'il y a convergence vers celle-ci quand $k \rightarrow +\infty$. De plus, on remarque que RANSAC permet bien de minimiser l'impact des outliers sur notre modèle, comme on peut voir que la régression linéaire classique est différente et ne représente pas bien les données.

Pour autant, on remarque que notre programme est plus coûteux en temps que celui que nous donne SkLearn. Ainsi, si nous sommes amenés à réutiliser la méthode RANSAC, nous utiliserons celle proposée par le module SKLearn.

2.4 Limites de RANSAC

Ainsi, ici nous allons utiliser le module SKLearn et notre algorithme pour trouver certaines limites de notre modèle.

2.4.1 Choix de d

On remarque que l'algorithme tel que nous l'avons présenté présente une valeur d , qui permet de demander qu'un nombre minimal de données correspondent au modèle. Pour autant, nous allons voir que cette donnée peut poser problème.

En effet, si nous prenons le cas où 60% de nos données sont des inliers et donc 40% sont des outliers (ce qui ne peut pas être sûr en amont) et que nous demandons d'avoir un modèle qui représente 60+% de nos données, alors RANSAC tel que nous l'avons codé ne pourra pas aboutir.

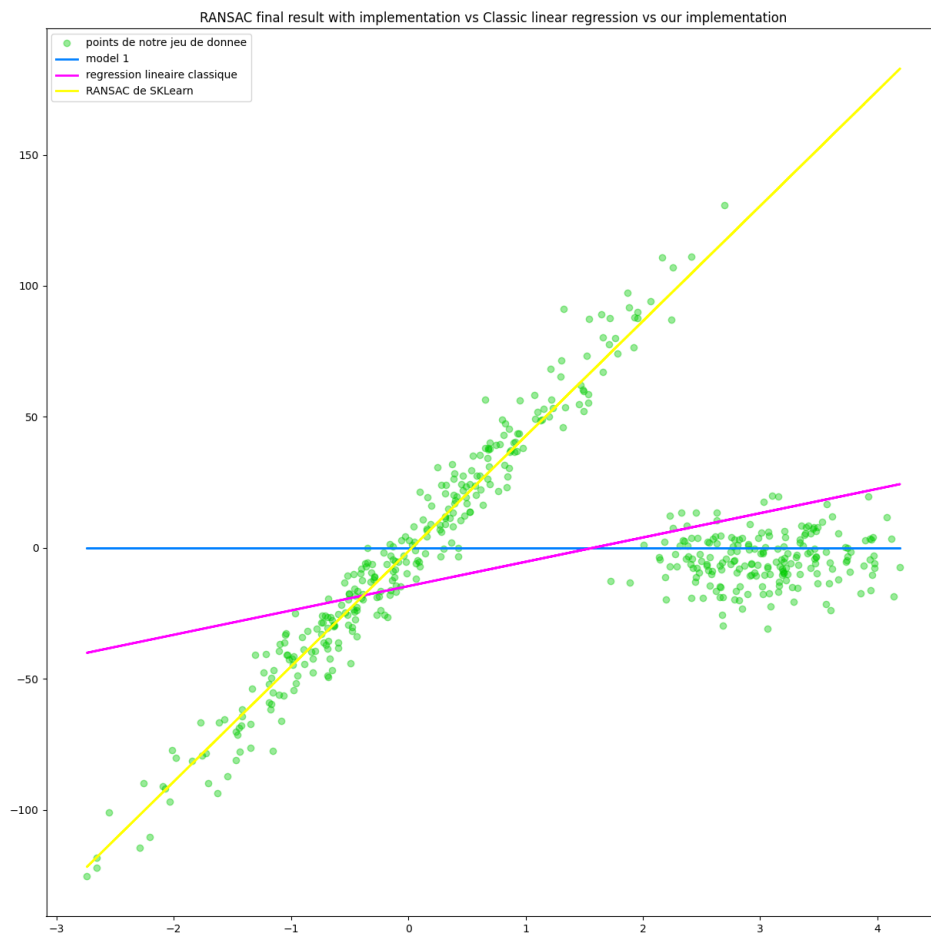
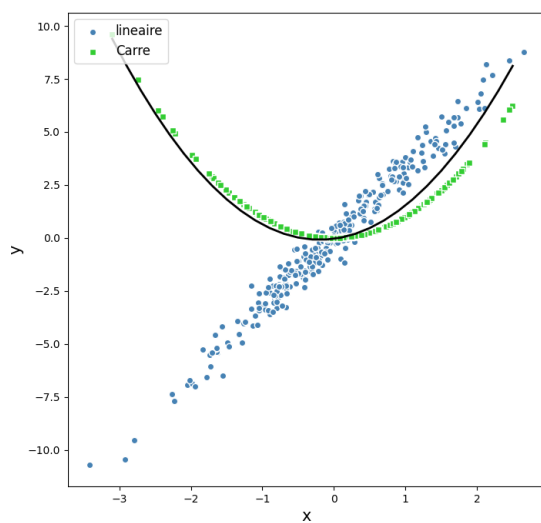


FIGURE 2 – Limitation de d

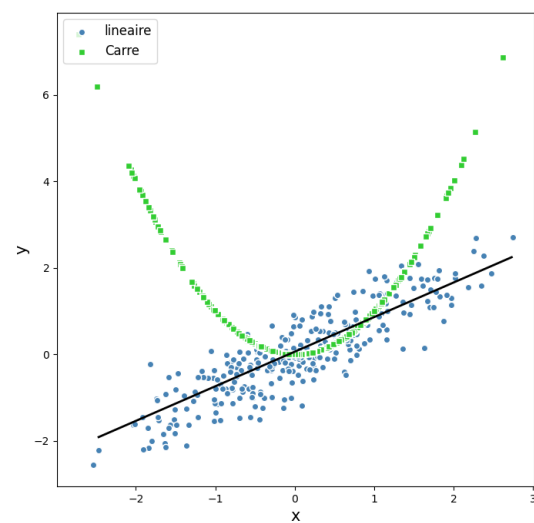
Pour autant, nous voyons ici que SKLearn a réussi à trouver un modèle qui correspond (visuellement) bien à nos données. Pour ce faire, SKLearn se soustrait du d , en effet, au lieu de vérifier si le nombre de inliers est au dessus de d , il va seulement vérifier si le nombre de nouveaux inliers est supérieur à celui du modèle envisagé à cette étape. Ainsi, on trouvera toujours un modèle dans ce cas. Pour autant, on pourra se retrouver avec un modèle qui ne représente pas assez de données par rapport à ce que l'on voudrait.

2.4.2 Mix de loi

Sachant que RANSAC est fait pour pouvoir seulement détecter un modèle à la fois, nous allons voir comment celui-ci réagit face à un mix de lois. Ici, nous avons généré un mix d'une loi linéaire et d'une loi quadratique. Voici les résultats :

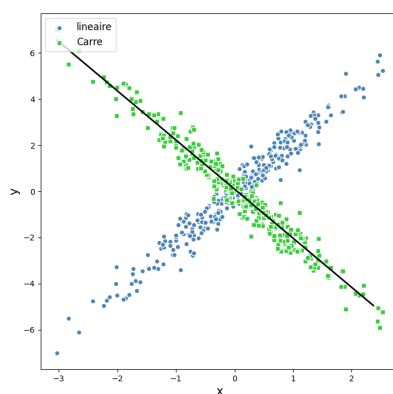


(a) Regression quadratique

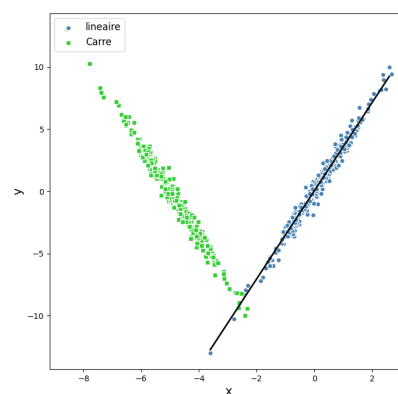


(b) Regression linéaire

On remarque que dans ce cas, il a réussi à bien distinguer les lois. Meme si l'on remarque que l'estimation de la loi quadratique est un peu influencé par les valeurs de notre loi linéaire. Pour ce faire, nous allons prendre deux lois linéaires.



(a) Regression linéaire



(b) Regression linéaire

2.5 Avantages et inconvénients

RANSAC permet de calculer de manière robuste (i.e non perturbé par les outliers) les paramètres du modèle. Pour autant, il n'y a pas de limite de temps avec laquelle on est sûrs d'avoir le meilleur modèle possible, ainsi lorsque l'on entre la limite k , nous ne sommes pas sûrs d'avoir la solution optimale pour notre jeu de données. Pour pallier à ceci, on peut décider non pas de stopper notre algorithme avec une limite k , mais demander un taux d'erreur à notre modèle, afin d'augmenter les chances d'avoir les paramètres optimaux. De plus, RANSAC ne peut estimer qu'un seul modèle par rapport à un jeu de données. Ainsi lorsque plusieurs modèles coexistent, cette méthode peut arriver à ne trouver aucun modèle.

3 Méthode ICE

3.1 Présentation générale de la méthode ICE

Maintenant que nous avons étudié la méthode RANSAC, nous allons regarder une autre méthode de classification qui nous permettra de mieux répondre à nos attentes. Pour cela, nous allons étudier la méthode ICE ou **Iterative Conditional Estimation**. Cet algorithme est une méthode d'estimation en contexte non supervisé. A partir d'observation de plusieurs sources, elle estime les paramètres du mélange (taux de mélange, degrés d'indépendance). Nos données sont aussi dites incomplètes, car le degré d'indépendance de nos observations entre elles est inconnu, il va donc être estimé à chaque itération.

3.2 Présentation théorique de la méthode ICE

Nous sommes dans le cadre où nous avons un modèle linéaire :

X : le vecteur des observations

S : le vecteur des classes

Tel que

Observation : $x = (x_1, \dots, x_N)$

classes : $s = (s_1, \dots, s_N)$ étant l'appartenance à des distributions probabilistes.

Dans ce cadre, s nous est inconnu, et nous voulons l'estimer.

On introduit le processus stochastique r appelé processus caché tel que :

— r soit à valeur dans $\{0,1\}$

— 1. $r(t) = 0 \Rightarrow$ on suppose que nos classes sont indépendantes, et au plus une des distribution est gaussienne.

2. $r(t) = 1 \Rightarrow$ On suppose que nos classes sont dépendantes.

Dans le cadre d'ICE, r n'est pas connu, ainsi celui ci va être estimé itérativement.

On introduit :

$$\theta = (\eta)$$

η caractérise la distribution de r

Ainsi θ est le paramètre à estimer.

On note

(r, X) : l'ensemble des données complètes
 X : Observations dites données incomplètes
 r : Processus caché

Afin d'utiliser ICE, il y a besoin de certains prérequis :

- Il existe un estimateur $\hat{\theta}(r, X)$ à partir des données complètes
- On peut calculer $\mathbb{E}[\hat{\theta}(r, X)|X; \theta]$ ou accéder à la mesure $\mathbb{P}[r|X; \theta]$

Ainsi, en partant d'une première estimation des paramètres, ICE trouve itérativement les paramètres de telle sorte :

$$\hat{\theta}^{(q)} = \mathbb{E}[\hat{\theta}(r, X)|X; \hat{\theta}^{(q-1)}]$$

Si cette expression n'est pas calculable, on peut la remplacer par la moyenne empirique de telle sorte que :

$$\hat{\theta}^{(q)} = \frac{1}{K} \sum_{k=1}^K \hat{\theta}(r^{(k)}, X)$$

Où $K \in N$ fixé, et chaque $r^{(k)}$ est une réalisation de la loi a posteriori $\mathbb{P}[r|X; \hat{\theta}^{(q-1)}]$. En pratique, on prendra généralement $K=1$.

On remarque donc que les conditions d'application d'ICE sont faibles, d'autant plus que la 1ère découle logiquement du fait que si il n'existe pas de tel estimateur, il n'y a pas d'espoir à trouver un estimateur sans la connaissance que nous apporte les données complètes.

Dans l'algorithme, on estime les dépendances de nos sources par rapport aux observations. Pour autant, cela est équivalent à les définir par rapport aux sources, en effet :

$$\mathbb{P}[r|X; \eta] = \frac{\mathbb{P}[r, X; \eta]}{\mathbb{P}[X; \eta]} = \frac{\mathbb{P}[r, S; \eta]}{\mathbb{P}[S; \eta]}$$

$$\text{Ainsi } \mathbb{P}[r|X; \eta] = \mathbb{P}[r|S; \eta]$$

Cette dernière est accessible car on suppose un modèle pour la loi de $\mathbb{P}[r, S; \eta]$

4 Lien théorique entre Ransac et ICE

Dans cette partie, nous appellerons la ou les droites diagonales nos données dites inliers, et les autres données seront donc du bruit considéré comme des outliers. Le but sera donc de faire une classification de ces données. De plus, une donnée outlier classée en tant qu'inlier sera appelée faux positif, et une donnée inlier classée outlier sera un faux négatif.

On va donc regarder le fonctionnement en parallèle de Ransac et ICE, en jouant sur le modèle, ou encore le bruit (origine et intensité).

4.1 Modèle linéaire, bruit uniforme

Commençons donc par regarder un modèle linéaire, avec un bruit uniforme.

Cas ou 50% des données sont du bruit : Voici le résultat :

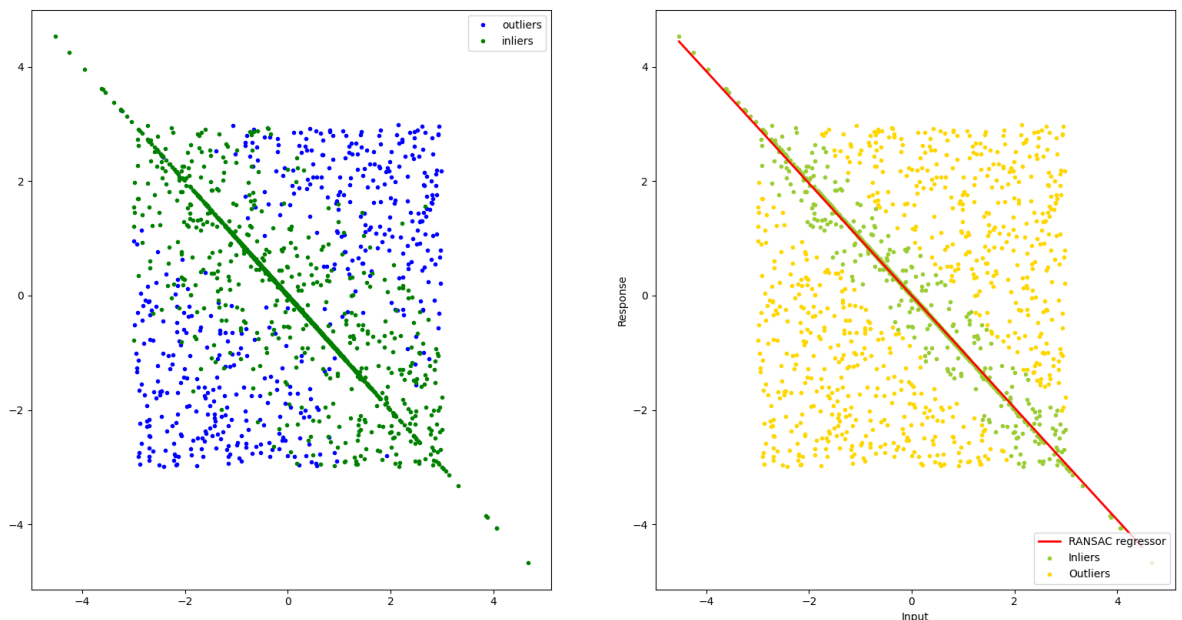


FIGURE 3 – Uniforme + 50% de bruit

On remarque dans ce cas que Ransac va toujours bien classifier les données de notre modèle, et les faux positifs seront dans un tube autour de notre modèle. Pour autant, nous pouvons modifier les paramètres de Ransac pour pouvoir réduire ce tube (dans la possibilité qu'un biais apparaisse). De l'autre côté, ICE ici arrive de même à classifier toutes nos bonnes données, il n'y a pas de faux négatifs. Pour autant, on remarque que le comportement n'est pas du tout le même en ce qui concerne les faux positifs, en effet sachant que le modèle se base sur un choix probabiliste, nous n'avons pas de tube mais une répartition plus ou moins centrée autour de notre modèle. Ainsi, l'erreur quadratique de ce modèle pourrait être plus forte. On peut aussi penser que grâce à cela, en appliquant un filtre, nous pourrions plus facilement enlever les faux positifs que pour RANSAC.

Mais dans le cadre de notre approche, nous avons codé à la main le fonctionnement de ICE dans ce cadre, mais nous prenons une méthode déjà faite pour RANSAC. Or, dans notre implémentation, nous pouvons jouer sur l'un des coefficient de notre modèle recherché pour avoir quelque chose de moins dispersé. Voici donc le résultat :

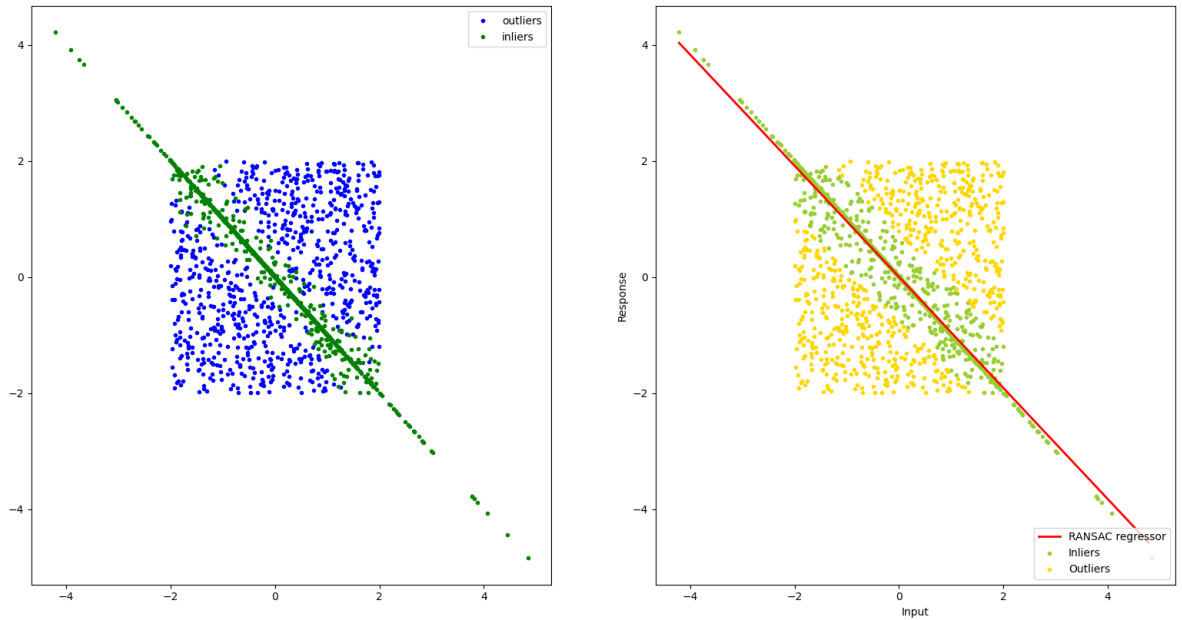


FIGURE 4 – Uniforme + 50% de bruit, lambda ajusté pour RANSAC

On remarque donc de suite que nos données sont bien moins dispersées pour ce qui est des faux positifs. Cet ajustement est donc bénéfique.

Il est aussi important de regarder une autre chose : l'estimation de la proportion d'outliers de notre algorithme ICE, dans ce cas ci il trouve en moyenne 55%.

Cas où 20% des données sont du bruit Voyons donc l'influence du bruit :

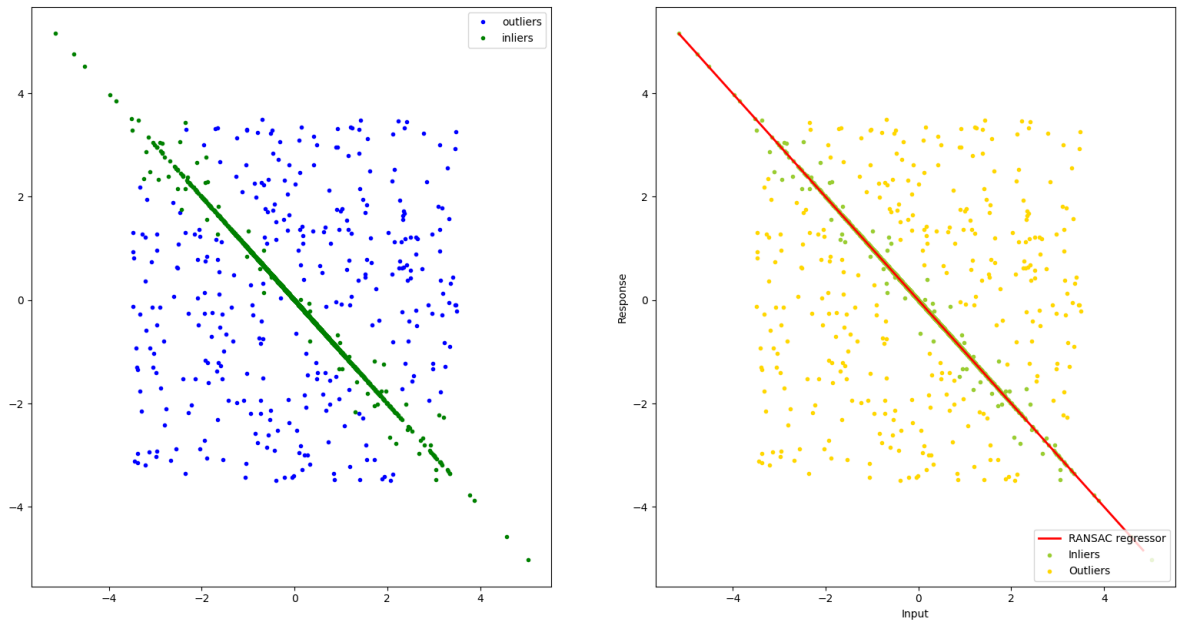


FIGURE 5 – Uniforme + 20% de bruit

Logiquement, lorsque nous avons moins de bruit, RANSAC marche de la même façon et on tire la même conclusion. On voit de même qu'avec un bon ajustement d'ICE il est cette fois-ci non pas meilleur que RANSAC mais similaire. Il faut savoir qu'ICE peut être plus coûteux en temps que RANSAC, ainsi dans ce cas il faudrait peut-être simplement utiliser RANSAC

Nous allons ensuite regarder les cas limites de ces modèles, c'est à dire beaucoup de bruit et très peu à aucun bruit.

Cas où 0% des données sont du bruit : Voici les résultats :

On commence par le cas extrême, si nous n'avions que des bonnes données, dans ce cas, RAN-

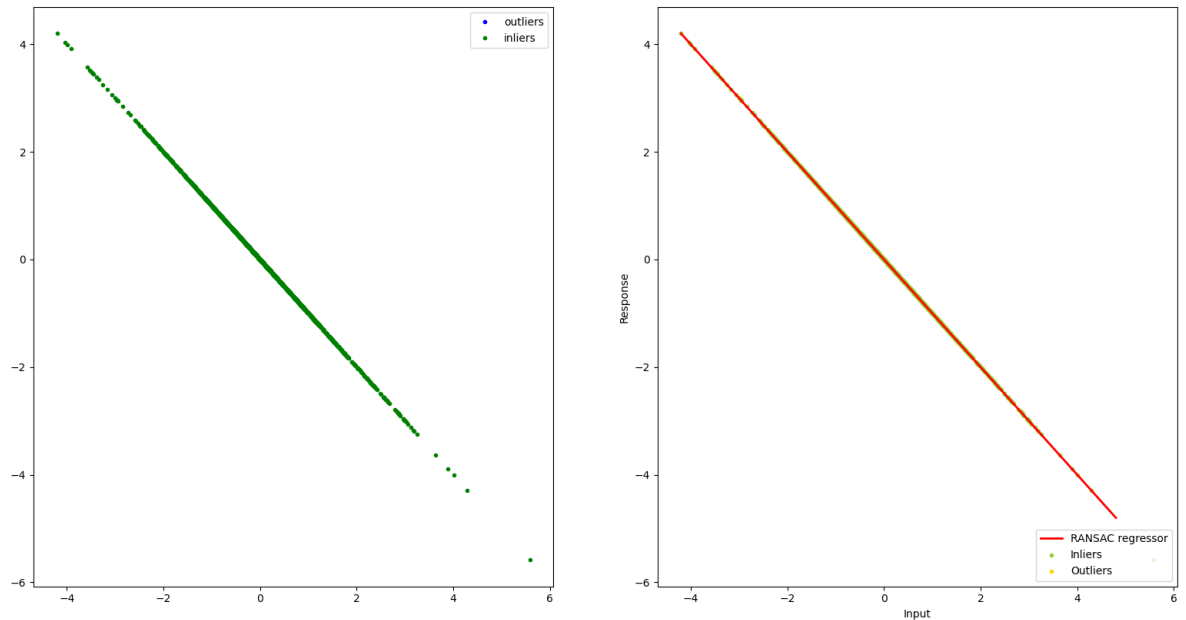


FIGURE 6 – Uniforme + 0% de bruit

SAC marche toujours aussi bien et classe toutes nos données comme bonnes. On remarque de même qu'ICE fonctionne parfaitement. C'est ici que l'on remarque que l'estimation de la proportion de bruit est fondamentale, en moyenne nous trouvons 1% de bruit avec l'algorithme. Ainsi, c'est grâce à cette bonne estimation que l'algorithme permet d'avoir un résultat fiable.

Cas où 5% des données sont du bruit : On regarde donc juste avant le cas limite et on prend 5% de bruit dans nos données. Ce qui est un cas faisable. Les deux méthodes permettent

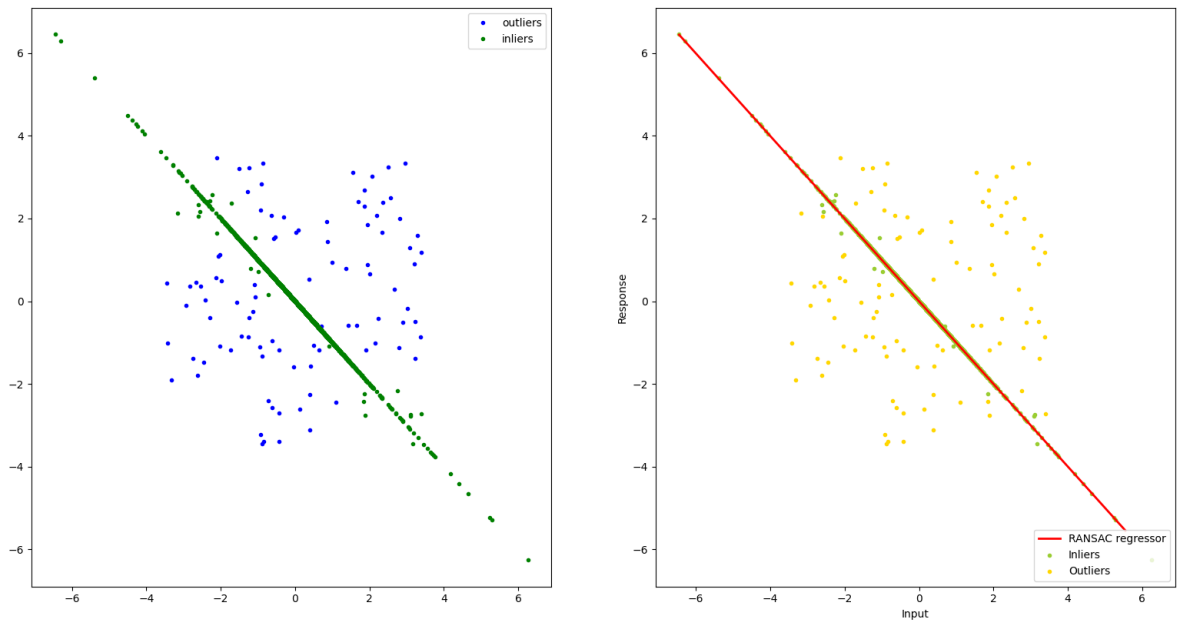


FIGURE 7 – Uniforme + 5% de bruit

de faire une classification sans faux négatifs. On remarque que RANSAC est un peu plus performant qu'ICE et de plus, les faux positifs de RANSAC sont inclus dans ceux d'ICE.

Cas où 70% des données sont du bruit : Voici les résultats.

Dans ce cas, on remarque clairement qu'ICE est supérieur à RANSAC. ICE suit la même dynamique qu'avant alors que RANSAC lui se voit biaisé. En effet, on remarque qu'il prend beaucoup d'outliers, d'autant plus que la droite générée par le modèle n'est pas bonne. Le bruit a donc un fort impact sur la robustesse de la méthode.

Cela nous pose donc la question de regarder la distribution de ce bruit, en effet nous avons donc ici mis un bruit uniforme. Mais que ce passerait-il avec un bruit gaussien qui serait donc bien plus dans le tube de RANSAC.

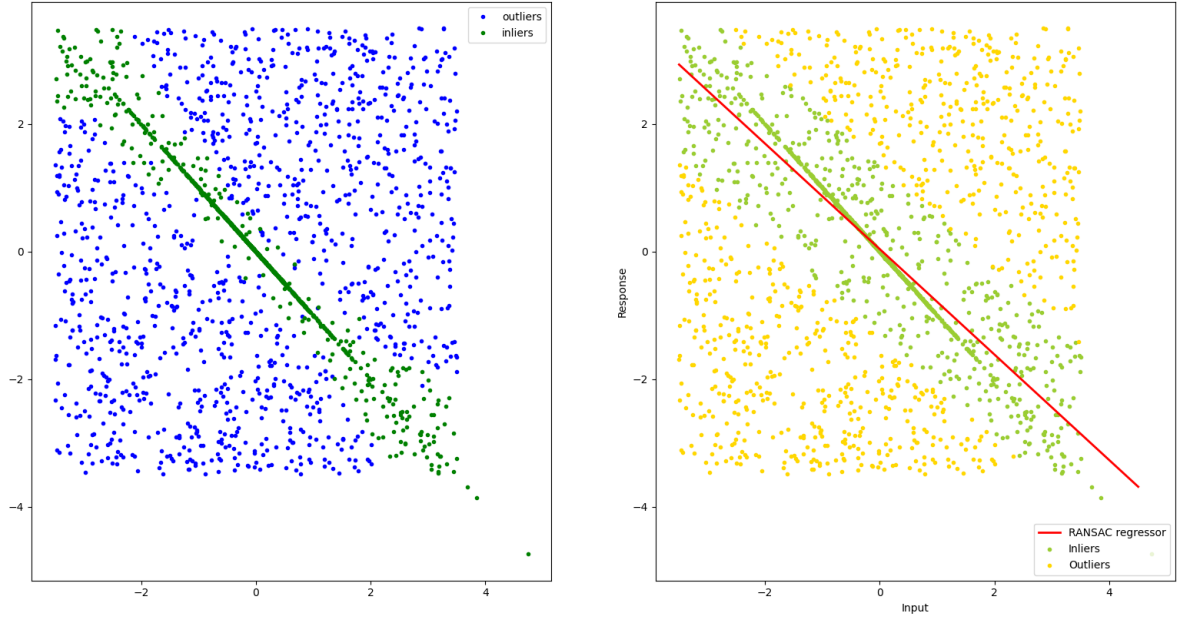


FIGURE 8 – Uniforme + 70% de bruit

5 Chaîne de Markov

Jusqu'ici nous considérons r comme un processus i.i.d, mais on pourrait aussi le considérer comme un processus Markovien, c'est à dire :

$$\mathbb{P}(r) = P(r(1)) \prod_{t=2}^T \mathbb{P}(r(t)|r(t-1))$$

où les probabilités de transitions de r seraient données. Dans ce cas, le paramètre η consiste en les probabilités de transition et la probabilité initiale de r . L'avantage principal de considérer r comme un processus Markovien plutôt que i.i.d est que la loi a posteriori de r sachant les observations X pourront alors être calculées avec l'algorithme "Forward-Backward". Il est alors possible d'échantillonner cette loi à posteriori et donc appliquer ICE.

Précédemment, l'état du processus r à la position $n-1$ n'impactait pas son état à la position n , ce qui était probablement la cause de certains problèmes évoqués plus haut (problème de classification aux extrémités). L'utilisation du processus Markovien pourrait permettre de régler ce problème en ajoutant de l'information supplémentaire à l'estimation de r .

Ceci n'a pas pu être fait dans notre Cassiopée, mais c'est une perspective d'avenir d'avenir pour un projet futur.

6 Synthèse de notre projet

6.1 Gestion de projet

Dans le cadre de la gestion de projet, il a fallu faire des plannings et pouvoir s'y tenir. Dans notre cadre, on s'est rendu compte que cela pouvait être difficile, car étant dans un projet de recherche, il est compliqué d'estimer avec justesse le temps nécessaire à chaque tâche. Pour autant, notre encadrant était compréhensif entre les phases plus lentes et les phases avec beaucoup d'avancement.

6.2 Apport académique

Notre projet Cassiopée nous a permis de rentrer dans l'état de l'art de méthode statistique de classification non supervisée. Cela a été stimulant de pouvoir faire face à des défis techniques, et d'avoir de nouvelles connaissances. Cela a été intéressant de pouvoir se plonger dans la peau d'un chercheur pendant quelques moments.