ANNEE 2022/2023

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

0.1 Presentation 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.

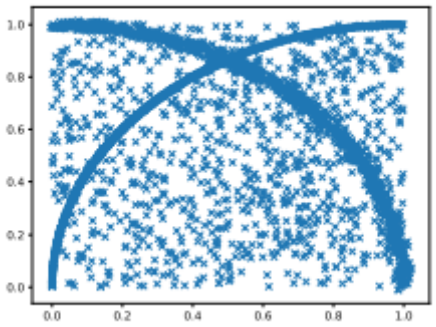


FIGURE 1 – Exemple d'un mélange de trois lois (singulière, singulière+bruit, uniforme)

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).**

0.2 Methode RANSAC

0.2.1 Presentation générale

RANSAC ou **RANdom SAmple Consensus** est une méthode itérative qui permet d'estimer un modèle mathématique a 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 algoritgme 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.

0.2.2 Algorithme

Algorithm 1 PseudoCode de l'algorithme RANSAC

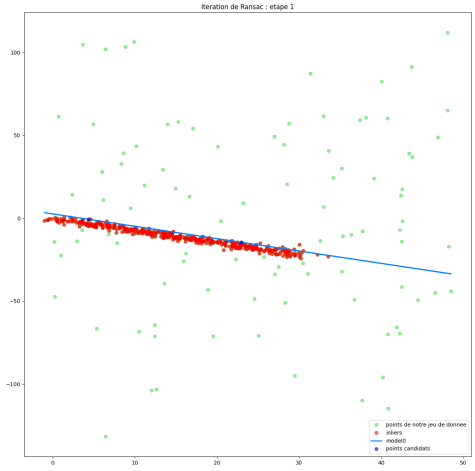
Require: data - jeu de données
modele - un modèle qui peut être ajusté à des données
n - le nombre minimum de données nécessaires pour ajuster le modèle
k - le nombre d'itérations de l'algorithme
t - une valeur seuil pour déterminer si une donnée correspond à un modèle
d - le nombre de données minimale prise 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

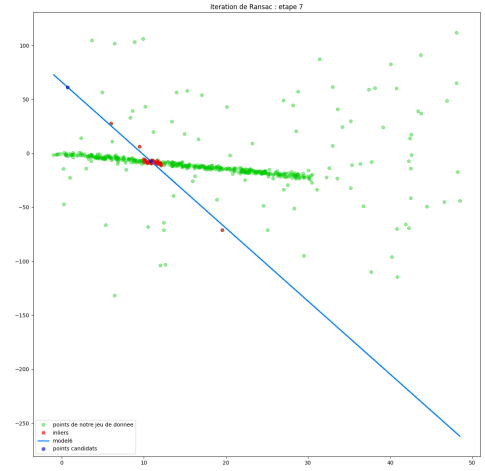
0.2.3 Implémentation de l'algorithme

Afin de bien comprendre le fonctionnement de RANSAC, nous avons décidé d'implanter RANSAC en python.(cf code : 0.3.1 Code de RANSAC en python)

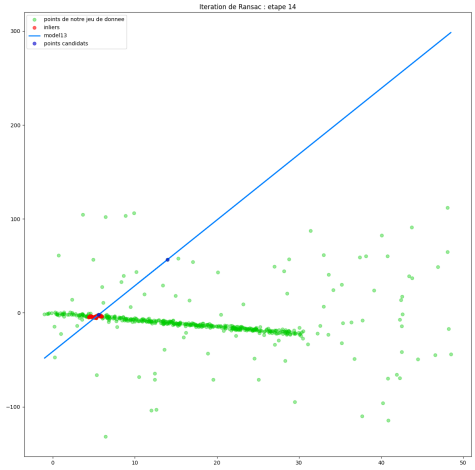
Ainsi, grâce à notre programme, nous pouvons comparer notre implémentation, celle de RANSAC classique et une simple régression linéaire.



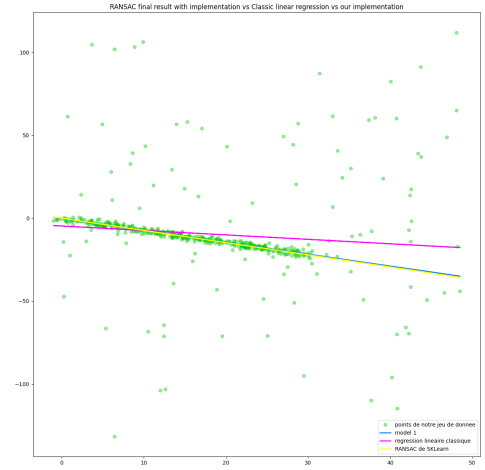
(a) 1ere étape



(b) 7eme étape



(c) 14 eme étape



(d) Resultat Final

Ainsi, on se rend compte que notre modèle se rapproche de la méthode RANSAC implantée par SkLearn et qu'il y a convergence vers celle-ci quand k-> + infini. De plus, on remarque que RANSAC permet bien de minimiser l'impact des outliers sur notre modèle, comme on peut voir que la regression linéaire classique est différente et ne represente pas bien les données.

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

0.2.4 Limites de RANSAC

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

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 donnée 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% de outliers (ce qui ne peut pas être sû en amont) et que nous demandons d'avoir un modèle qui represente 60+% de nos données, alors RANSAC telle que nous l'avons codé ne pourra pas aboutir.

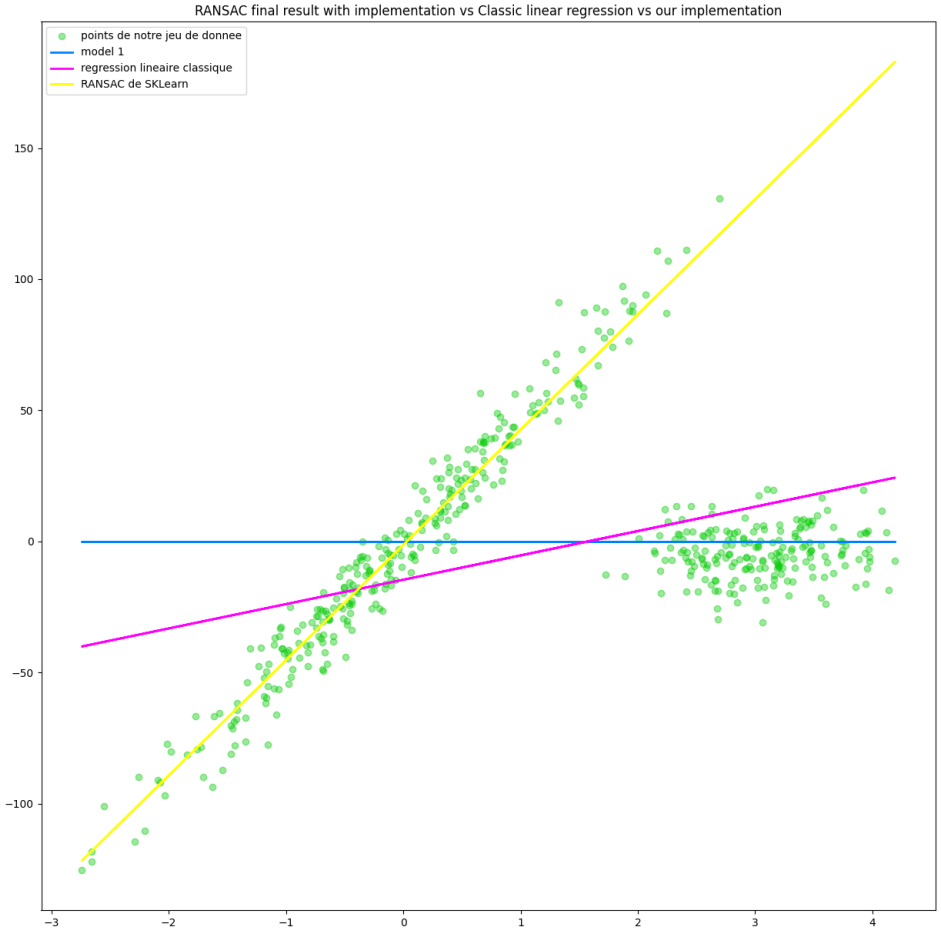
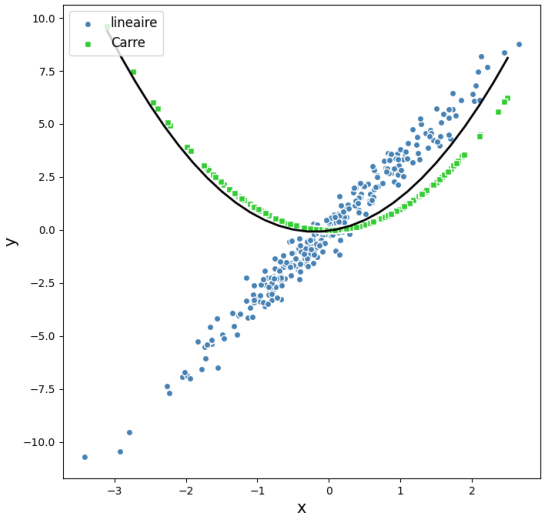


FIGURE 2 – Limitation de d

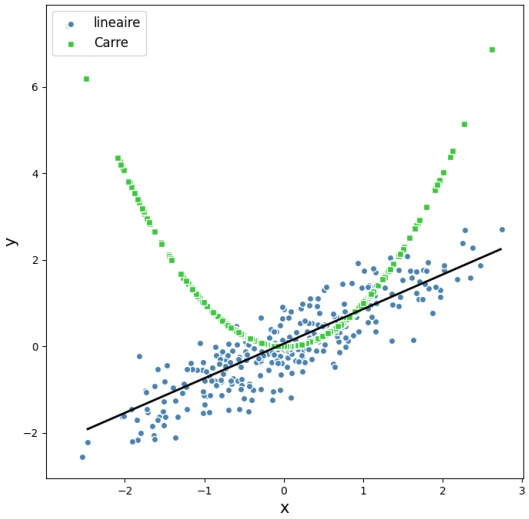
Pour autant, nous voyons ici que SKLearn à réussi à trouver un modèle qui correspond (visuellement) bien à nos données. Pour ce faire, SKLearn se soustrait du d, en effet, au lieu de vérifier si le nombre de inliers est au dessus de d, il va seulement verifier si le nombre de nouveaux inliers est supérieur 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 represente pas assez de donnée par rapport à ce que l'on voudrait.

Mix de loi

Sachant que RANSAC est fait pour pouvoir seulement detecter 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 resultats :

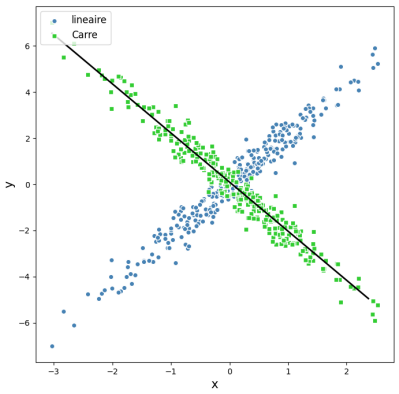


(a) Regression quadratique

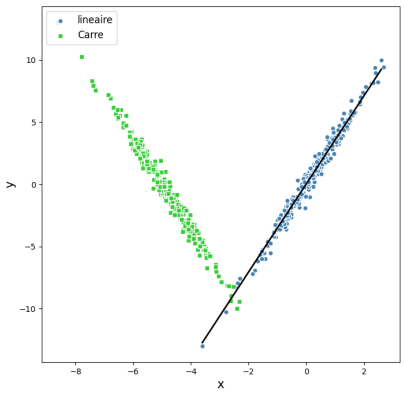


(b) Regression linéaire

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

0.2.5 Avantages et inconvenients

RANSAC permet de calculer de manière robuste (ie non perturbé par les outliers) les paramètres du modèle. Pour autant, il n'y a pas de limite de temps avec laquelle on est sur d'avoir le meilleur modèle possible, ainsi lorque l'on entre la limite k, nous ne sommes pas sûr d'avoir la solution optimale pour notre jeu de données. Pour palier 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 raport à un jeu de données. Ainsi lorsque plusieurs modèles coexsistent, cette méthode peut arriver à trouver aucun modèle.

0.3 Méthode ICE

0.3.1 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 plus 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, degres d'independance). . Nos données sont aussi dites incompletes, car le degrès d'indépedance de nos observations entre elle est inconnu, il va donc être estimer à chaque itération.

0.3.2 Présentation théorique de la méthode ICE

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

X : le vecteurs des observations
S : le vecteur des classes
Tel que
Observation : $x = (x_1, ..., x_N)$
classes : $s = (s_1, ..., s_N)$ Etant l'appartenance à des distributions probabilistes.

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

On introduit le processus stochastique r appélé 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 éstimeé 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 : Observation 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 acceder 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 postériori $\mathbb{P}[r|X; \hat{\theta}^{(q-1)}]$. 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 dépédances de nos sources par rapport aux observation. 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 accéssible car on suppose un modèle pour la loi de $\mathbb{P}[r, S; \eta]$

0.4 Lien théorique entre Ransac et ICE

Dans cette partie, nous appelerons 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 outliers classée en tant qu'inlier sera appelée faux positifs, 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).

0.4.1 Modèle linéaire, bruit uniforme

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

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

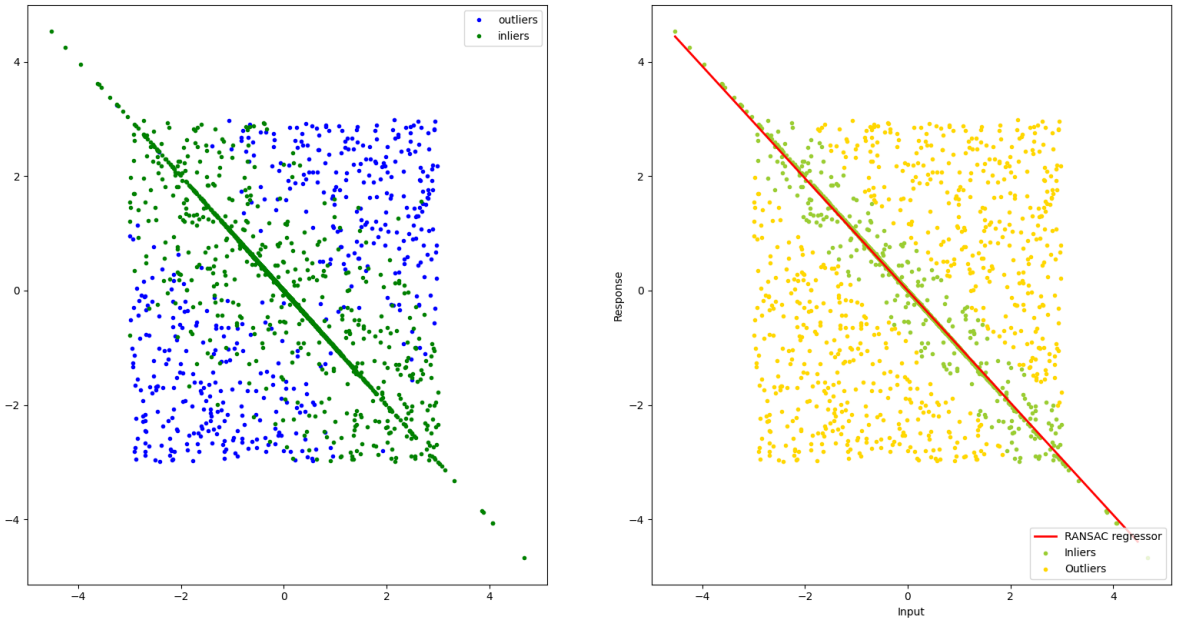


FIGURE 3 – Uniforme + 50 de bruit

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 paramètre de Ransac pour pouvoir réduire se tube (dans la possibilité qu'un biais apparaisse). De l'autre coté, 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 dutout 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 repartitions plus ou moins centrée autour de notre modèle. Ainsi, l'érreur quadratique de ce modèle pourrait être plus forte. On peut aussi penser que grâce a 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 a la main le fonctionnement de ICE dans ce cadre, mais nous prennons une méthode deja faite pour RANSAC. Or, dans notre implentation, nous pouvons jouer sur l'un des coéficient de notre modèle recherché pour avoir quelque chose moins dispersé. Voici donc le resultat :

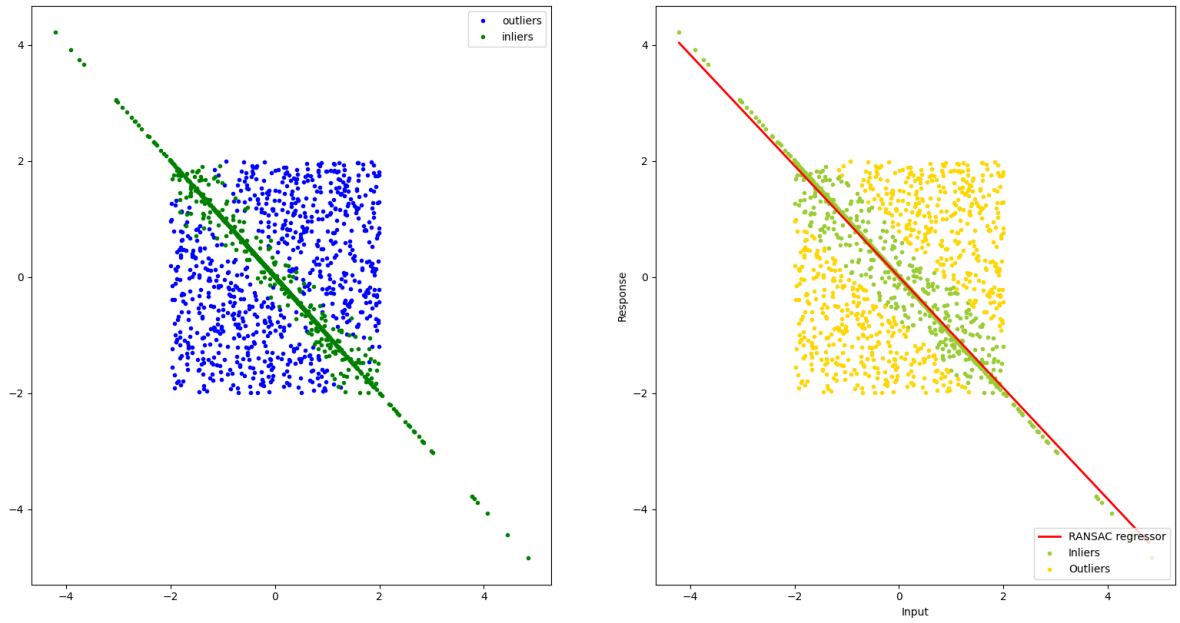


FIGURE 4 – Uniforme + 50% de bruit, lambda ajusté pour RANSAC

On remarque donc de sutie que nos données sont bien moins dispersés 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'outlier de notre algorithme ICE, dans ce cas ci il trouve en moyenne 55

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

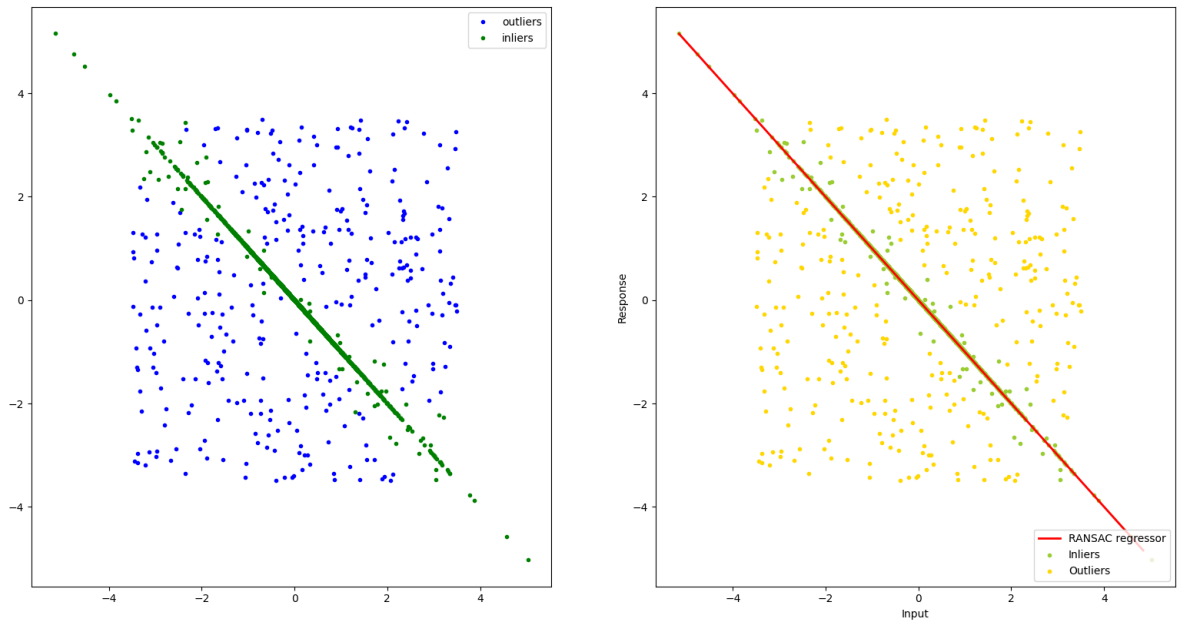


FIGURE 5 – Uniforme + 20% de bruit

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 couteux 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 a dire beaucoup de bruit et très peu à aucun bruit.

Cas où 0% des donneés sont du bruit : Voici les resultats :
On commence par le cas extreme, si nous avions que des bonnes donneés, dans ce cas, RANSAC

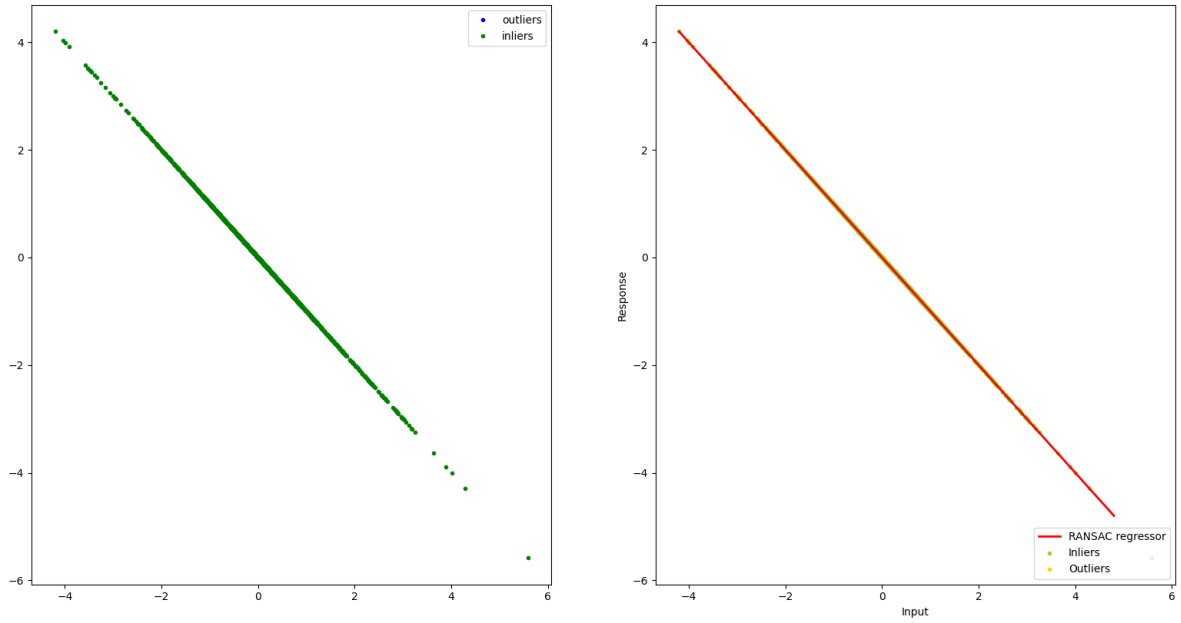


FIGURE 6 – Uniforme + 0% de bruit

marche toujours aussi bien et classifie toute 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 grace à cette bonne estimation que l'algorithme permet d'avoir un resultat fiable.

Cas où 5% des donneés 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

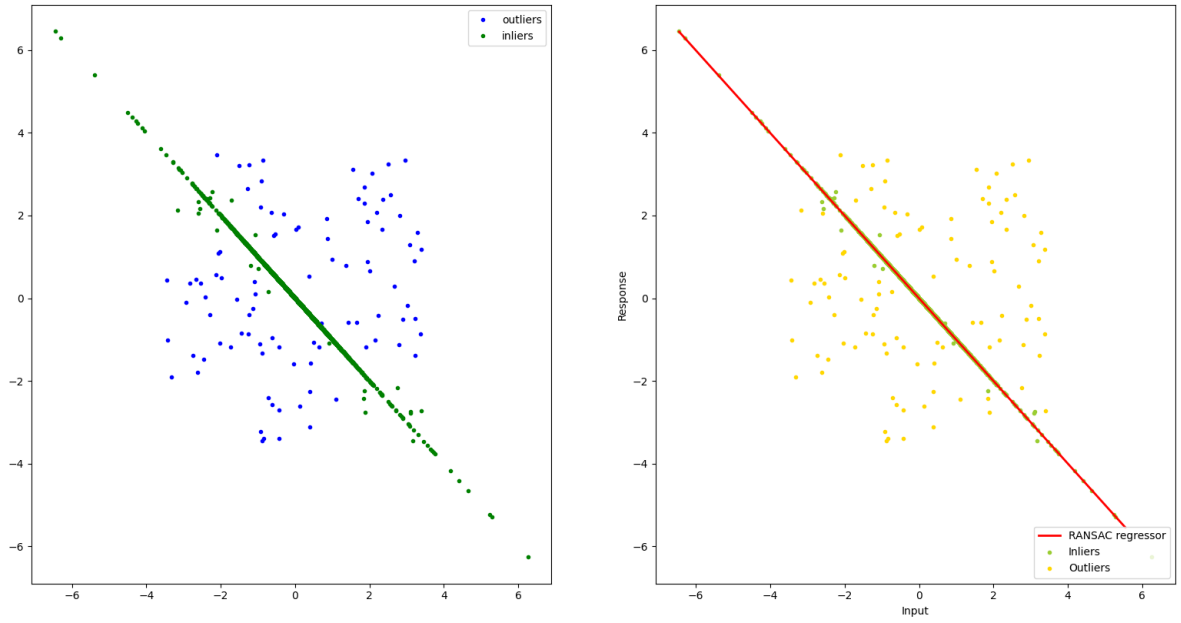


FIGURE 7 – Uniforme + 5% de bruit

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.

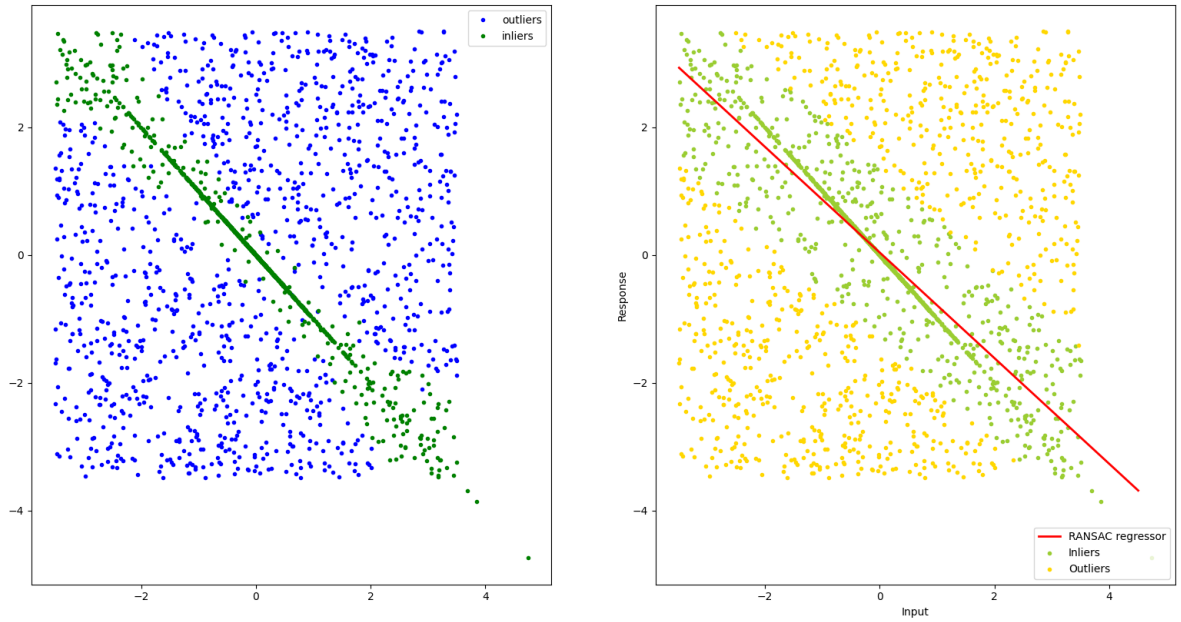


FIGURE 8 – Uniforme + 70% de bruit

Cas où 70% des donneés sont du bruit Dans ce cas, on remarque clairement qu'ICE est supérieur à RANSAC. ICE suit la même dynamlique 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 robustèce 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.

0.4.2 Modèle linéaire, bruit gaussien

Nous avons donc modifié la distribution de notre bruit, pour savoir si de nouveau comportement allait pouvoir permettre de rapprocher ou non RANSAC et ICE.

Cas où 20% des donneés sont du bruit Voici le resultat. Ainsi lorsque notre bruit est

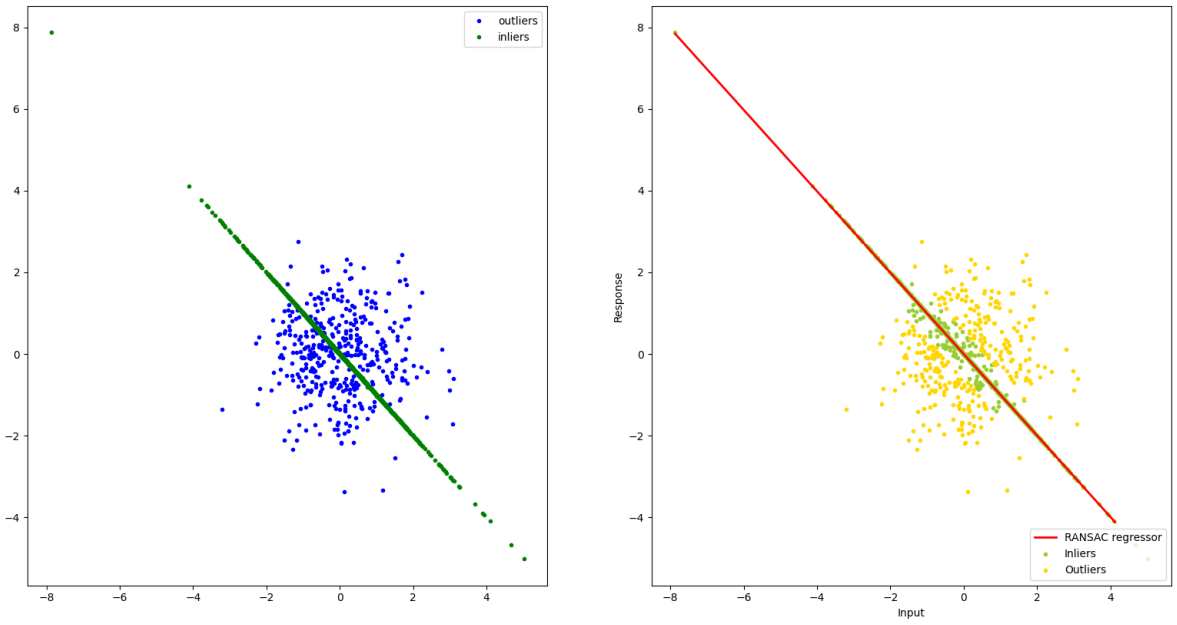


FIGURE 9 – Gaussien + 20% de bruit

faible, nous remarquons que les deux ont un comportement similaire avec toujours plus d'outliers pris en compte pour RANSAC. Nous sommes dans une configuration ou nous avons un lambda adapté pour bien avoir la distribution.

Cas où 50% des donneés sont du bruit Voici le resultat. Même conclusion que pour 20

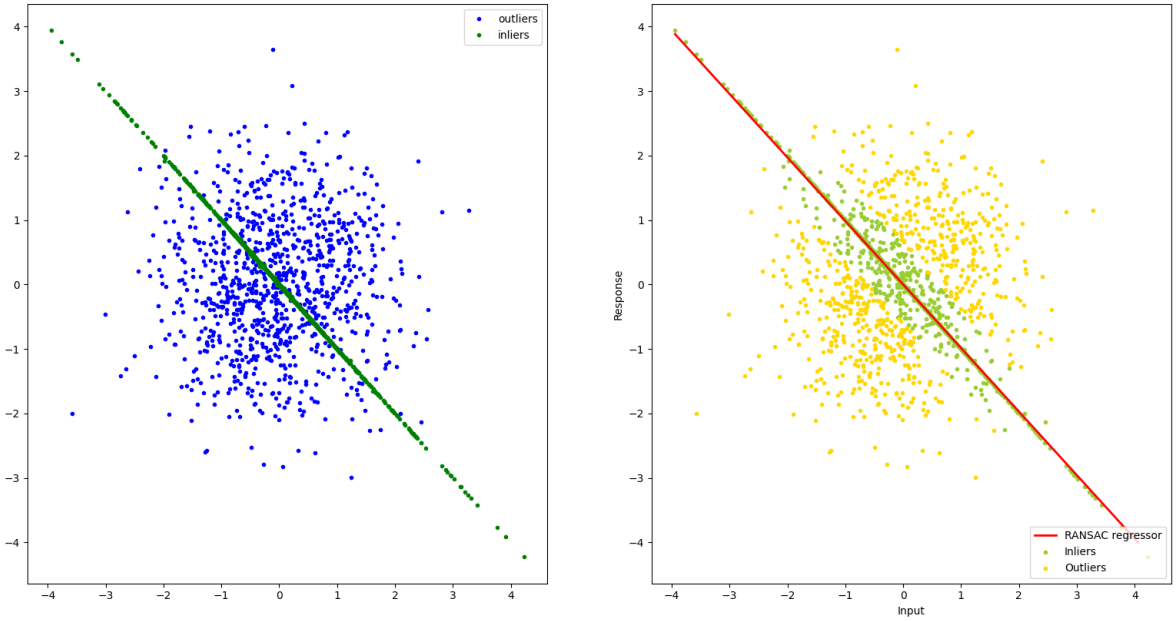


FIGURE 10 – Gaussien + 50% de bruit

Cas où 70% des donneés sont du bruit Voici le resultat.

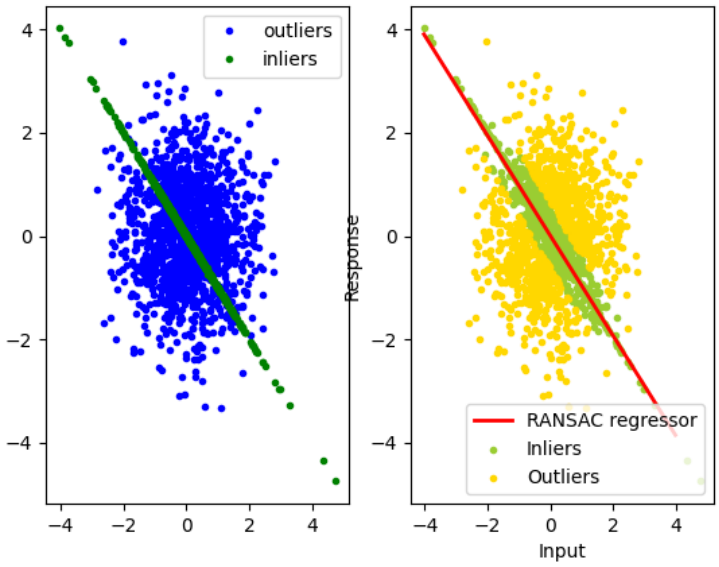


FIGURE 11 – Gaussien + 70% de bruit

Dans ce cas-ci, nous montrons ce qu'il se passe lorsque le lambda d'ICE est mal ajusté, dans ce cas, on remarque que le comportement est similaire que RANSAC. Pour autant, il est bien plus simple et fiable d'ajuster ICE que RANSAC. En effet, pour ICE, nous ajustons la loi que nous demandons, et c'est une connaissance que nous pouvons recuperer empiriquement, alors

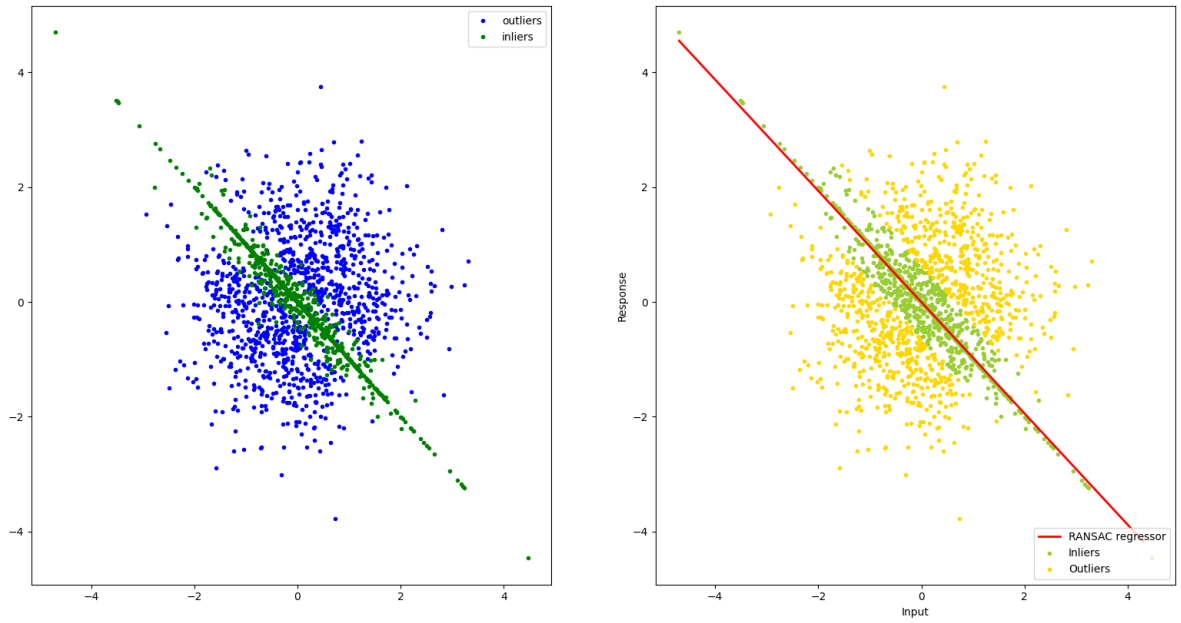


FIGURE 12 – Gaussien + 70% de bruit non ajusté

que si l'on veut ajuster RANSAC, il faut ajuster l'erreur que nous acceptons. Ainsi ICE trouve un modèl qui va avec notre loi, mais RANSAC regarde juste l'erreur et donc il peut nous donner totalement autre chose que la loi de nos inliers. C'est pourquoi en terme de fiabilité, on peut dire qu'ICE est largement en tête.

0.4.3 Double modèle linéaire, bruit uniforme

Nous avons deja vu que RANSAC n'était pas optimal quand deux modèles sont mis ensemble. Pour autant nous allons quand même faire les mêmes test pour bien comprendre nos méthodes.

Cas où 20% des donneés sont du bruit : Voici le resultat.

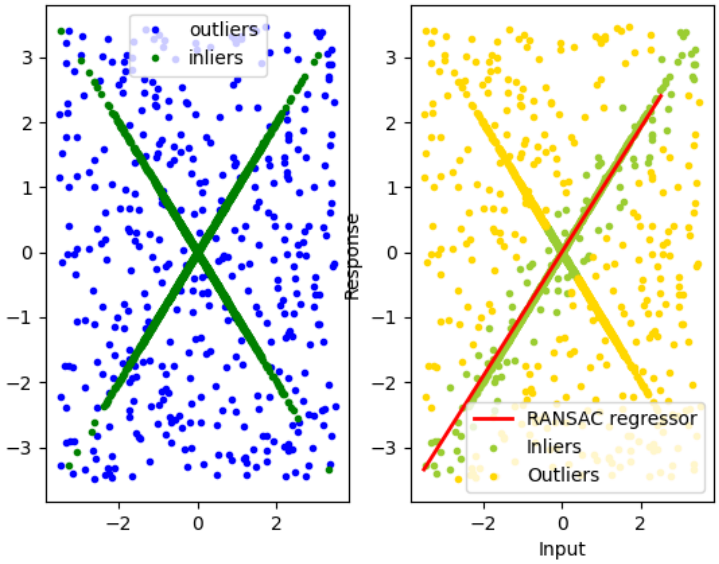


FIGURE 13

On remarque qu'ici ICE trouve les deux modèles sans soucis, et RANSAC réussi à trouver au moins un des des modèles et donc les inliers de celui-ci.

Cas où 50% des donneés sont du bruit : Voici le resultat. Dans ce cas, ICE fonctionne toujours aussi bien, pour autant on remarque qu'ICE n'arrive pas à trouver au moins un modèle il y a des faux négatifs pour le modèle qu'il pense avoir trouvé.

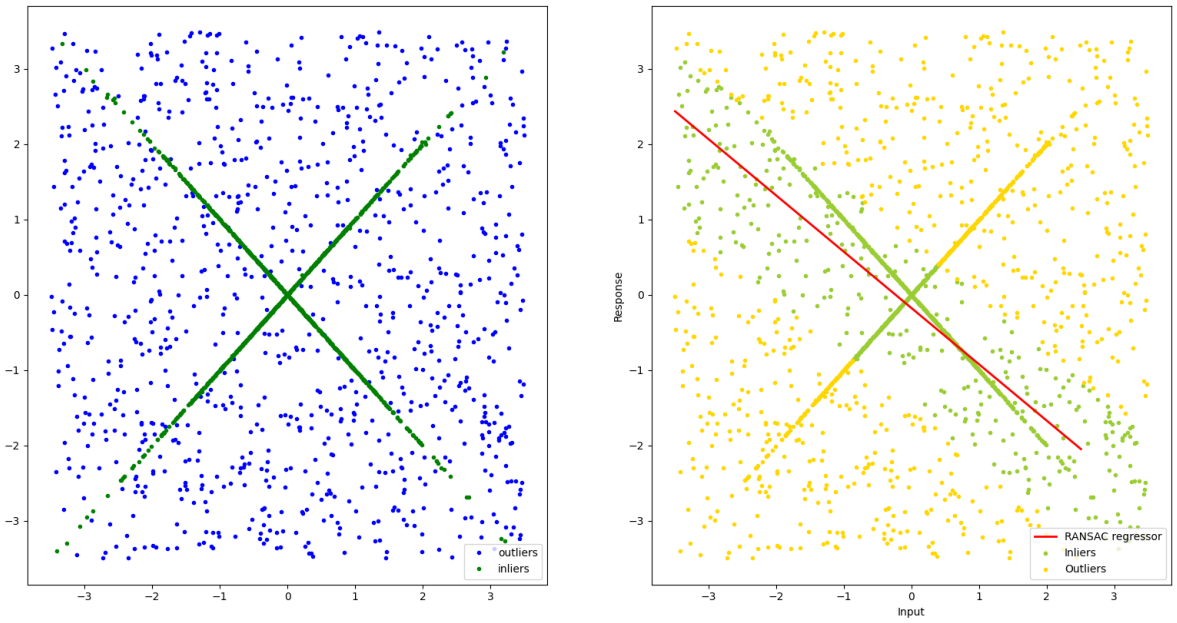


FIGURE 14

Cas où 70% des donneés sont du bruit : Voici le resultat.

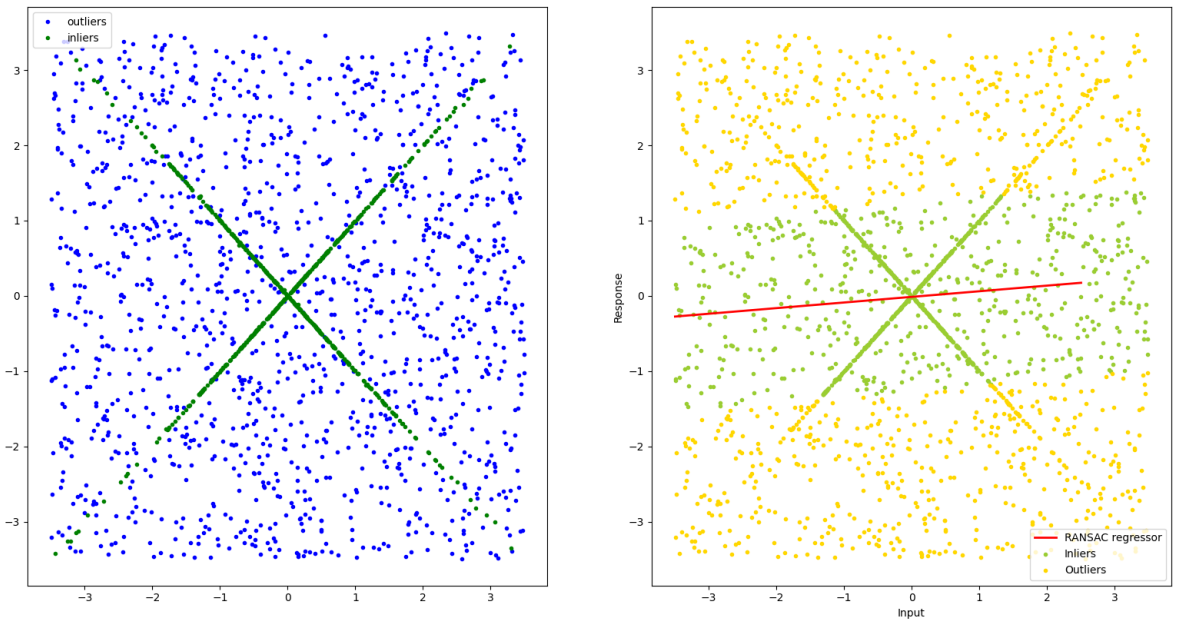


FIGURE 15

RANSAC n'arrive plus dutout a trouver l'un des deux modèl. Pour autant, ICE a lui aussi du mal avec les valeurs extrèmes, celle qui sont au bout de nos droites, comme l'on peut le voir, des valeurs qui pouvaient être prise en compte dans le cas 20% ne sont pas prise en compte.

0.5 Chaine de Markov

0.5.1 Introduction des chaines

Jusqu'ici nous considérions 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 probailités 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'échnatillonner 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 extrêmités). L'utilisation du processus Markovien pourrait permettre de régler ce problème en ajoutant de l'information supplémentaire à l'estimation de r.

0.5.2 Resultat de l'implémentation