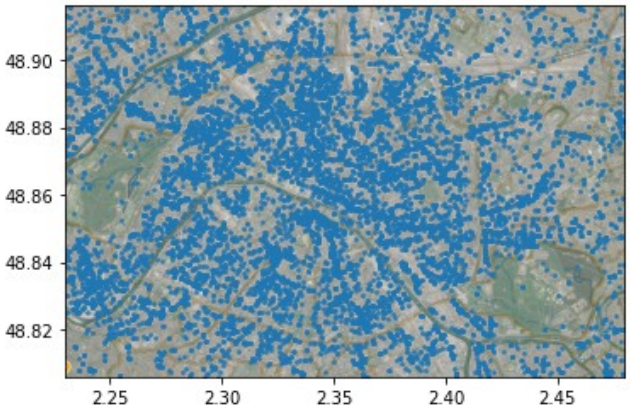
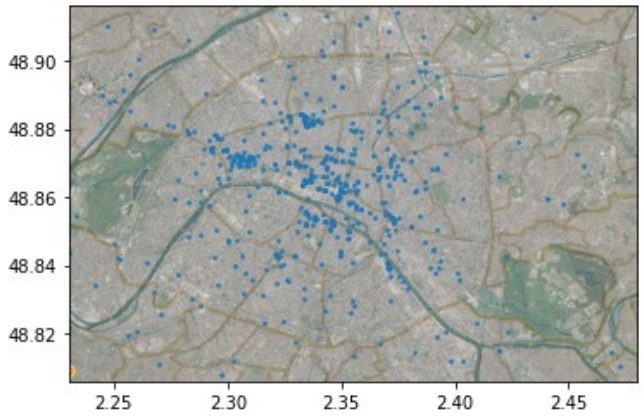
Luc STERKERS
Dao THAUVIN

[ML]-TME2-Estimation de densité

I - Données et Vraisemblance



Carte des restaurants et bars de Paris.



Carte des night clubs et bars de Paris.

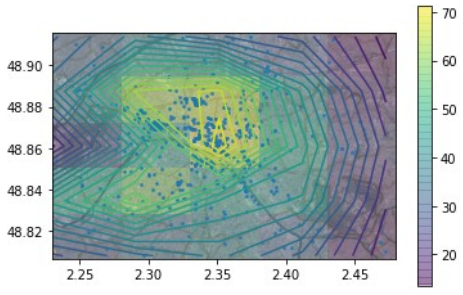
Nous n'utiliserons que la base de données des restaurants et bars de Paris (la plus part du temps) qui sont plus nombreux pour nos entrainements et tests.
L'ajout d'une valeur très faible lors du calcul de la vraisemblance permet d'éviter un log de 0 si la vraisemblance est nulle.

II - Méthode par histogramme

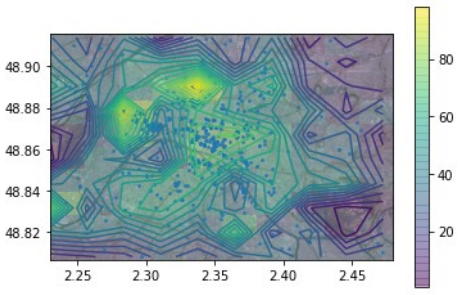
Si notre estimateur est correct, on doit vérifier que la grille retournée par get_density2D est la même que celle de notre histogramme (à une transposé près) et son intégrale doit aussi sommer à 1 (ce qui est bien le cas ici).
Test d'intégrale à 1 (res est le résultat de la fonction get_density2D) :

```
somme = np.sum(res)*((xmax-xmin)*(ymax-ymin))/(step**2)
print(somme)
```

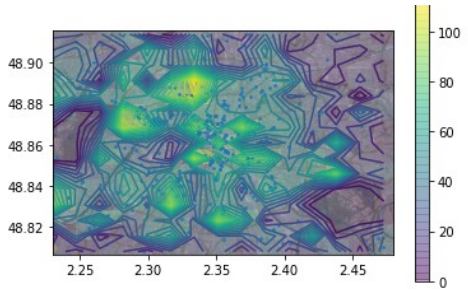
Nous avons testé nos données avec différents pas, voici quelques résultats pour commencer.



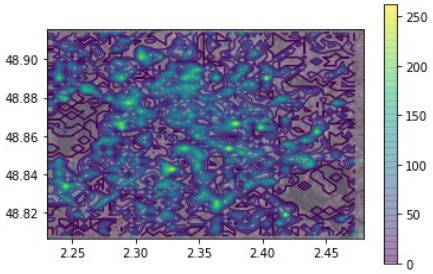
*Densité des bars et restaurants de Paris avec une discrétisation de **5***



*Densité des bars et restaurants de Paris avec une discrétisation de **10***



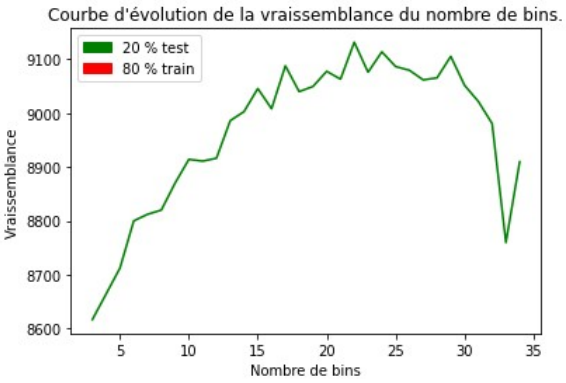
*Densité des bars et restaurants de Paris avec une discrétisation de **15***

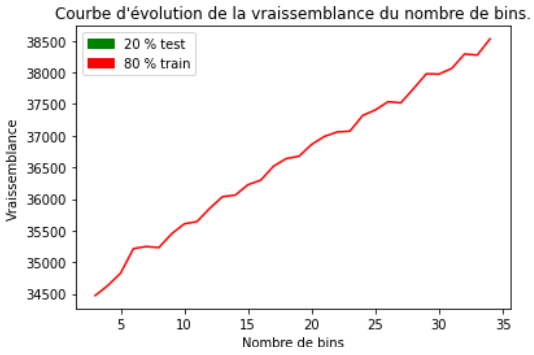


*Densité des bars et restaurants de Paris avec une discrétisation de **25***

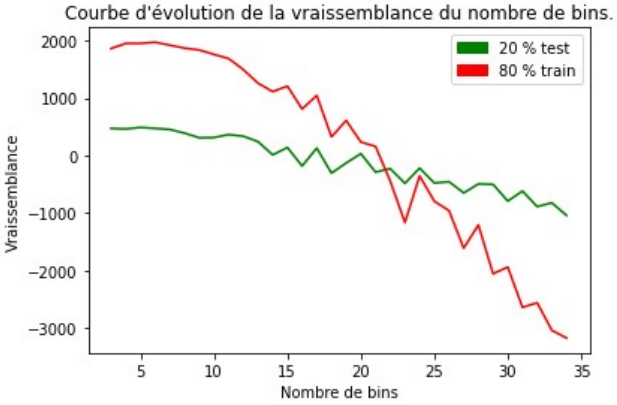
On remarque sur ces différentes cartes de densités qu'une faible discrétisation ne permet pas de différencier les lieux géographiques entre eux.
Au contraire, une trop forte discrétisation entraîne un sur-apprentissage, les lieux géographiques de fortes densités sont seulement un bar ou un restaurant, il ne s'agit pas d'une zone. On n'a donc pas d'informations géographiques globales sur la présence des bars et des restaurants. Une discrétisation moyenne (comme 15 dans notre exemple), nous permet d'avoir à la fois, des lieux de forte et de faible densité et des informations géographiques globales sur la présence des bars et restaurants.

Nous avons alors séparé nos données en deux sous-ensembles, un correspond à un ensemble aléatoire de 80% de nos données pour l'apprentissage de nos modèles et un autre de 20% pour les tests.
Ces ensembles sont tirés aléatoirement pendant chaque test de modèle.
Voici les courbes obtenues avec la méthode par histogramme sur un pas de 3 à 35 :





On observe du sur-apprentissage à un pas d'environ 25 (baisse de vraisemblance en test). Un pas de 15 ou même 20 est donc un bon pas de discrétisation.
Nous réalisons la même expérience avec les données des night clubs beaucoup moins nombreuses.



On observe que la valeur la plus intéressante est 3, cela pourrait être dû à 2 raisons :

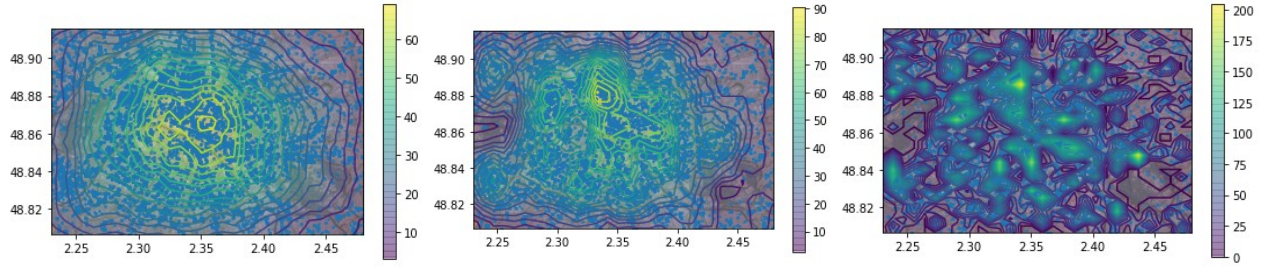
- Le nombre de données en test est trop faible, nos tests ne sont donc pas valables.
- Le nombre de données d'apprentissage est trop faible pour pouvoir avoir une bonne approximation des nouvelles données

Une cross validation serait surement plus adapté à cette situation.

III - Méthodes à noyaux

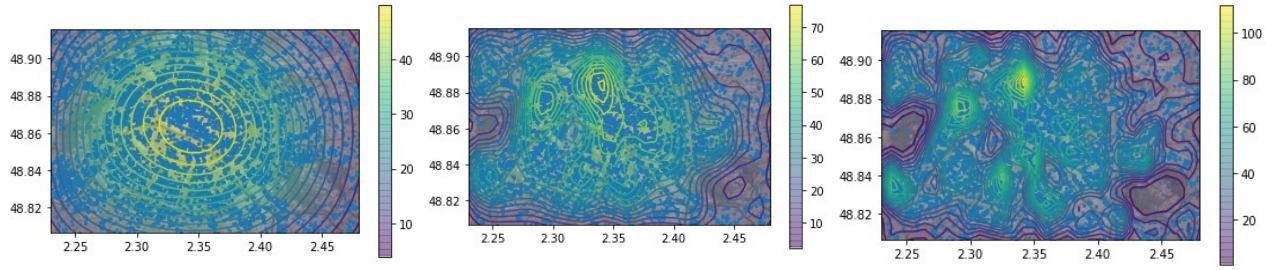
Visualisation :

- Noyau Uniforme :



Densité des bars et des restaurants avec un sigma de 0.06	Densité des bars et des restaurants avec un sigma de 0.03	Densité des bars et des restaurants avec un sigma de 0.005

- - Noyau Gaussien



Densité des bars et des restaurants avec un sigma de 0.03	Densité des bars et des restaurants avec un sigma de 0.01	Densité des bars et des restaurants avec un sigma de 0.005

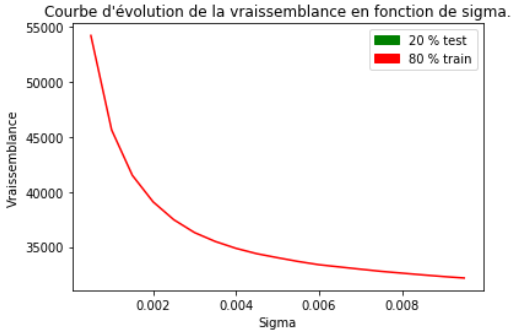
On observe que plus sigma est faible, plus on est proche de nos données.

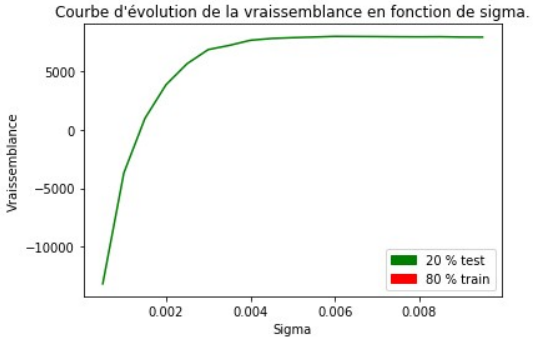
Le noyau gaussien converge d'ailleurs moins vite vers nos données quand sigma diminue.

Séparation Données Test/Entrainement :

Avec la même méthode que précédemment nous avons séparé nos données en apprentissage et entrainement.

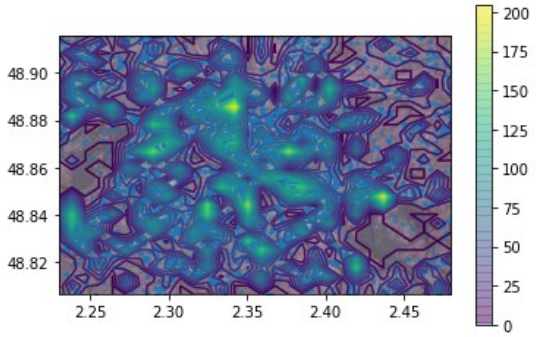
- Noyau Uniforme





On observe que la valeur 0.005 semble être une bonne valeur de sigma.
Ici, le surapprentissage ce fait à gauche comme nous l'avons vu précédemment, on est proche de nos données lorsque sigma est faible.

La carte de densité (avec un pas de 30) obtenue correspond à :

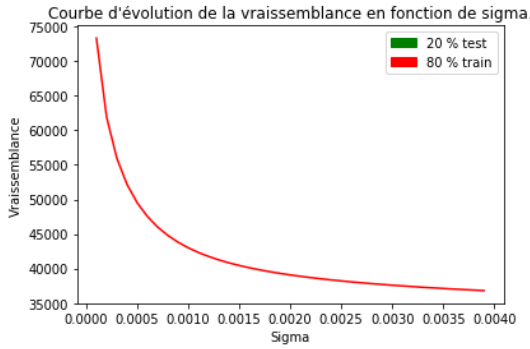


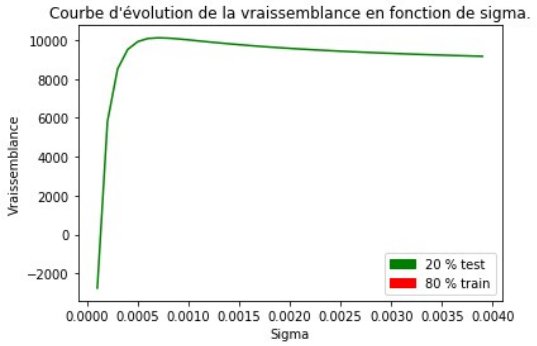
Test d'intégrale à 1 (res est le résultat de la fonction get_density2D) :

```
somme = np.sum(res)*((xmax-xmin)*(ymax-ymin))/(step**2)
print(somme)
```

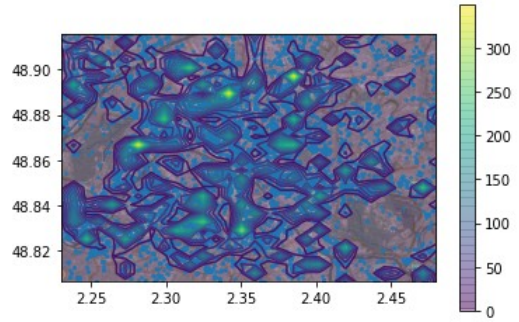
On observe une somme de 0.95.
Remarque :
On utilise la fonction get_density2D qui récupère la densité de rectangles en récupérant la densité d'un point.
Pour l'histogramme, la densité d'un point correspond à la densité du rectangles ou il se situe, on avait donc un très bon résultat car get_density2D fait le calcul de densité sur le même rectangle.
Les noyaux ne sont pas forcement rectangle ce qui peut créer une marge d'erreur (notamment avec le noyau gaussien. Même avec le noyau uniforme, il est impossible de faire la même chose que pour la méthode par histogramme car nos noyaux sont carrés et non des rectangles.).
Il faut aussi avoir une taille de noyau proche de la taille du rectangle de get_density2D pour avoir une valeur proche de 1 (sinon des zones ne seront pas pris en compte ou pris en compte plusieurs fois).
On a ici choisi un pas de 30 qui nous donne une assez bonne valeur en étant assez rapide.

- Noyau Gaussien





Cette fois ci la valeur optimale semble être sigma = 0.0006.
On obtient alors la carte de densité (avec un pas de 30) :



Test d'intégrale à 1 (res est le résultat de la fonction get_density2D) :

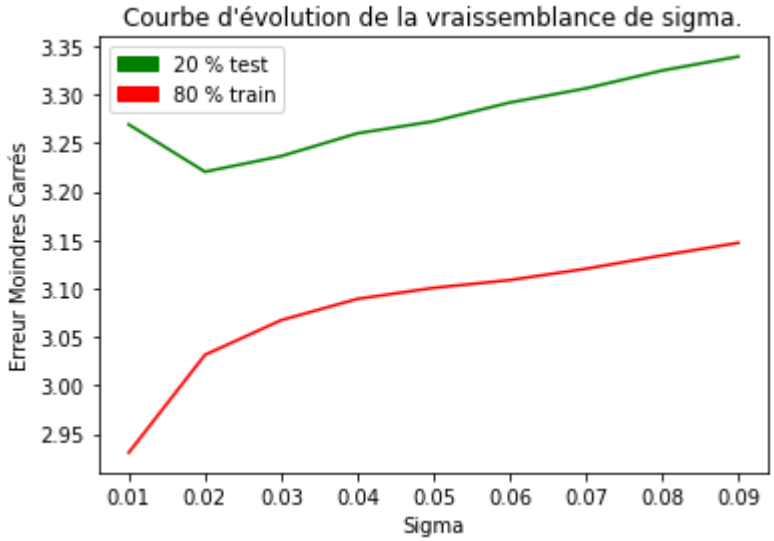
```
somme = np.sum(res)*((xmax-xmin)*(ymax-ymin))/(step**2)
print(somme)
```

On observe une somme de 0.95 ce qui est plutôt bien (pour la même raison que le noyau uniforme).

IV – Regression par Nadaray-Watson

Voici notre courbe en faisant varié sigma :

- Noyau Uniforme



On obtient une erreur minimum sur les tests avec sigma de 0.02.
On essaye de prédire les notes des 10 premières notes des bars/restaurants avec le reste des bars/restaurants avec un sigma de 0.02.
Notes de bases :

```
[4.1 -1. 4.7 3.5 3.5 2.4 4.3 4.3 3.4 3.7]
```

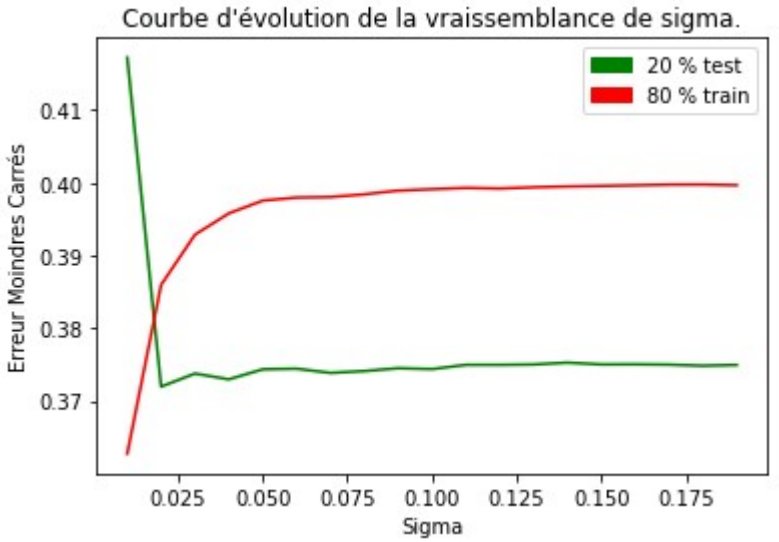
Nos notes :

```
[3.98 -1. 4.2 1.8 3.4 -1. 3.7 3.9 3.85 3.8]
```

On est assez proche des notes attendues.
Remarque : on note -1 quand aucune donnée d'entrainement n'est trouvée proche de notre point (la somme des kernels sur la base d'entrainement donne 0).

On essaye maintenant en enlevant les notes -1 qui correspond surement à des notes inexistantes et qui peuvent fausser nos résultats.

On obtient alors la courbe :



On observe une erreur beaucoup plus faible qu'avant (on passe de 3.2 en test à 0.37), le sigma optimal ne change pas beaucoup, 0.025 semble une bonne valeur de sigma.
Avec le même test on obtient :

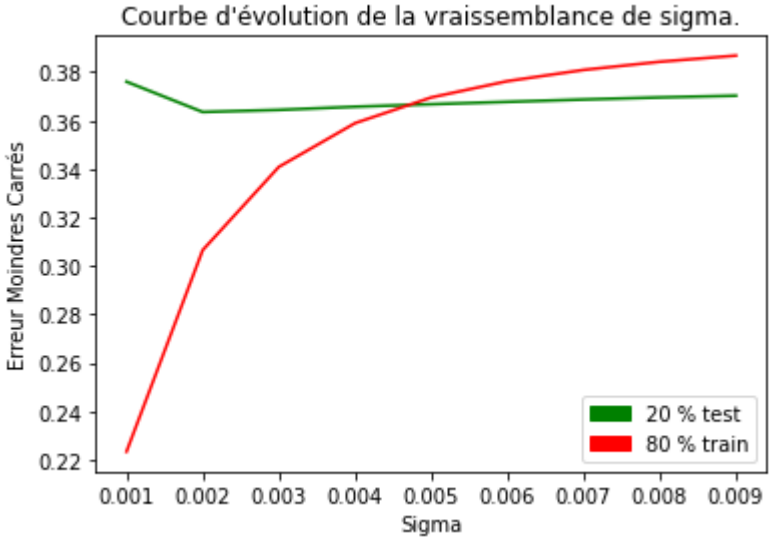
```
[4.0364532  4.0561753  3.81428571 4.02884615 3.9409901  3.96299435 3.98166667
4.03122677 3.85545024 3.8982684]
```

Pour les notes de bases :

```
[4.1 4.7 3.5 3.5 2.4 4.3 4.3 3.4 3.7 3.8]
```

On est beaucoup plus proche qu'avant avec nos données !
Les -1 étaient bien la cause d'une erreur supplémentaire.
On essaye de prédire des données avec

- Noyau Gaussien
  Cette fois-ci on ne test nos données que sur des données vraiment notées.
  On obtient la courbe :



Cette fois-ci la valeur la plus cohérente pour sigma semble 0.002 si on minimise l'erreur en test et en train ou 0.005 si on veut un même taux d'erreur en test et en train.
Avec 0.005 on obtient :

```
[4.07285526  3.99671866  3.89827207  3.91470461  3.84976171  3.91578743
3.97952594 4.002758 3.71323434 3.90240326]
```

Avec 0.002 on obtient :

```
[4.06418109  3.96236548  3.95151621  3.63802376  3.87549742  3.94724
3.81388183 4.03524526 3.43808776 4.03212127]
```

Les notes de bases sont :

```
[4.1 4.7 3.5 3.5 2.4 4.3 4.3 3.4 3.7 3.8]
```

Là aussi on est très proche de nos notes mais avec une erreur légèrement plus faible qu'avec le noyau uniforme (0.36 au lieu de 0.37).