

école —
normale —
supérieure —
paris — saclay —

MASTER MVA

GÉNÉRATION DE DONNÉES EN IA PAR TRANSPORT ET DÉBRUITAGE
CHALLENGE DATA ENS 2025
RAPPORT

Challenge par Egis

"Prédisez les infractions de stationnement !"

Élèves : Colin COËRCHON & Chloé HABASQUE

Enseignant : Stéphane MALLAT

Président du Jury : Enzo DALBY

Providers : Daniel RAYBAUT & Raphael VIENNE



19 mars 2025

Table des matières

1	Introduction	2
1.1	Entreprises	2
1.2	Contexte et Enjeux	2
1.3	Objectif du Challenge	2
2	Découverte et traitement des données	2
2.1	Exploration des données	2
2.2	Analyse des données	3
2.3	Analyse des données en fonction de la cible	5
2.4	Transformation des Features	7
2.5	Nettoyage des Données	8
2.5.1	Gestion des valeurs manquantes	8
2.5.2	Suppression des lignes dupliquées	8
2.5.3	Filtrage des valeurs extrêmes de <code>total_count</code>	8
2.5.4	Analyse de données de position mise à l'échelle	8
2.6	Choix du nombre de voitures	9
2.7	Normalisation des données	10
3	Clustering des données	10
3.1	Heuristique 1 : K-means simple	12
3.2	Heuristique 2 : Délimitation de bordure et K-means	12
3.3	Heuristique 3 : Variante de l'heuristique 2	15
3.4	Résumé de nos 3 heuristiques	16
3.5	Corrélation finale	17
4	Modèles et évaluation des performances	17
4.1	Métrique d'évaluation	17
4.2	Modèles utilisés	18
4.2.1	Régressions linéaires (Ridge et Lasso)	18
4.2.2	Support Vector Machine (SVM)	18
4.2.3	Random Forest	18
4.2.4	XGBoost	19
4.2.5	Moyenne des modèles	19
5	Résultats	19
5.1	Influence du filtrage des données	19
5.2	Résultats sur le dataset public	19
5.3	Résultats sur le dataset privé	20
5.4	Classement final	20
6	Idées d'améliorations	20
6.1	Utilisation de RANSAC	20

1 Introduction

1.1 Entreprises

Ce challenge a été fourni par la société Egis en collaboration avec DataCraft.

1.2 Contexte et Enjeux

Le stationnement urbain constitue un enjeu majeur pour les municipalités, notamment en ce qui concerne le respect des réglementations et la gestion des espaces disponibles. Egis, groupe international d'ingénierie et d'exploitation, joue un rôle central dans l'exploitation du contrôle du stationnement sur voirie. Les systèmes de contrôle automatique s'appuient sur des véhicules équipés de caméras pour capturer des données essentielles telles que les plaques d'immatriculation, la localisation et l'horodatage. L'analyse de ces données est cruciale pour optimiser l'efficacité des contrôles et réduire le taux d'infraction.

L'objectif de ce challenge est de développer un modèle de Machine Learning capable d'identifier les zones et les horaires les plus propices aux infractions de stationnement. Cela permettrait d'optimiser les tournées des véhicules de contrôle, réduisant ainsi le coût d'exploitation et améliorant l'efficacité des interventions.

1.3 Objectif du Challenge

Le but de cette étude est de proposer un modèle de régression permettant de prédire le taux d'infraction futur en fonction des données historiques et des facteurs externes, notamment les conditions météorologiques. L'évaluation des modèles se fait à l'aide de la corrélation de Spearman, une métrique adaptée à l'identification des zones de forte infraction relative.

2 Découverte et traitement des données

2.1 Exploration des données

Les données mises à disposition comportent les informations suivantes pour chaque contrôle effectué :

- **Taux d'infraction constaté (Variable cible)** : Proportion de véhicules en infraction dans la zone concernée.
- **Nombre total** : Nombre total de véhicules contrôlés dans la zone.
- **Latitude et longitude** : Localisation géographique du contrôle mise à l'échelle.
- **Horodatage** : Mois (de 1 à 12), jour de la semaine (de 1 à 6) et heure du contrôle (de 6 à 19).

- **Données météorologiques** : Conditions climatiques à partir des données open data Météo France, incluant les précipitations, la hauteur de neige, la température et la force du vent.

Description	Nombre de données	Valeurs manquantes	Nombre de variables
X_{train}	6 076 546	170 098	10
Y_{train}	6 076 546	0	1
X_{test}	2 028 750	7 906	10

TABLE 1 – Résumé des caractéristiques des données utilisées

2.2 Analyse des données

L'analyse exploratoire des données permet d'examiner leur distribution et d'identifier les principales caractéristiques de chaque variable.

Les données sont constituées de plusieurs types de variables :

- **Variables numériques** : `total_count`, `longitude_scaled`, `latitude_scaled`, `Precipitations`, `HauteurNeige`, `Temperature` et `ForceVent`.
- **Variables temporelles** : `day_of_week`, `month_of_year` et `hour`.

Répartition des variables temporelles :

- `day_of_week` et `month_of_year` sont uniformément distribués, bien que `day_of_week` ne prenne que six valeurs.
- `hour` couvre une plage de 5h à 19h, avec une répartition relativement uniforme entre 8h et 17h. Cependant, on observe une légère baisse du nombre d'observations sur les créneaux 5h-8h et 17h-19h.

Répartition des variables numériques :

- `total_count` : La médiane est de 4, et la quasi-totalité des valeurs se situe entre 0 et 100. Quelques observations extrêmes atteignent des valeurs jusqu'à 1669.
- `Precipitations` : La grande majorité des valeurs sont nulles, avec seulement environ 13% des observations contenant des valeurs non nulles.
- `HauteurNeige` : Une variable quasi-binaire, où 99.6% des valeurs sont nulles et seulement 0.4% des observations présentent une hauteur de neige de 1.
- `Temperature` et `ForceVent` suivent une distribution gaussienne :

- Temperature est centrée autour de 15°C.
- ForceVent est centrée autour de 3.2.

L'histogramme ci-dessous illustre ces distributions :

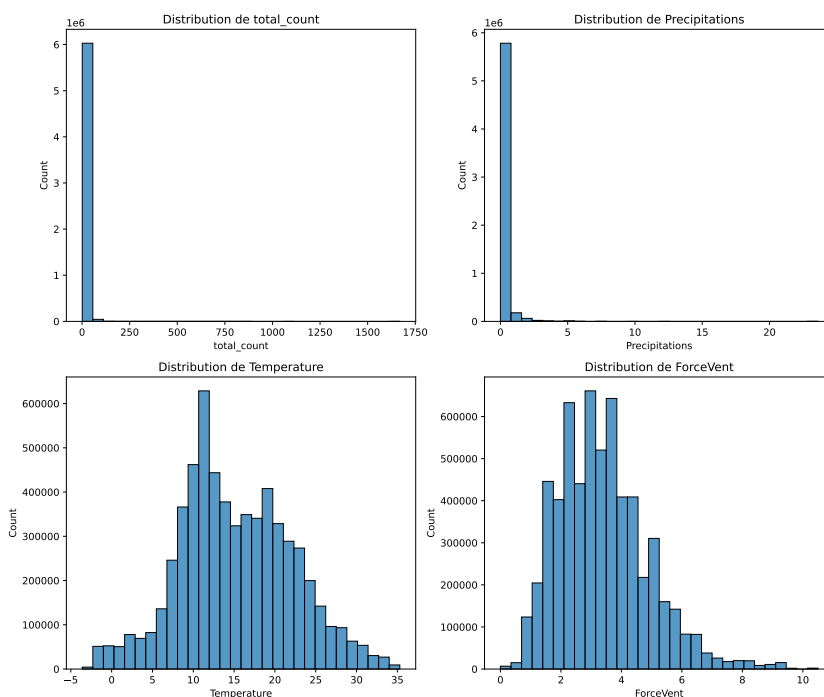


FIGURE 1 – Histogrammes des variables `total_count`, `Precipitations`, `Temperature` et `ForceVent`.

Corrélation entre les variables Enfin, nous observons que, à l'exception de `total_count`, aucune variable ne présente une forte corrélation avec la cible. Cela signifie que la prédiction du taux d'infraction nécessitera des méthodes avancées pour capturer d'éventuelles relations non linéaires.

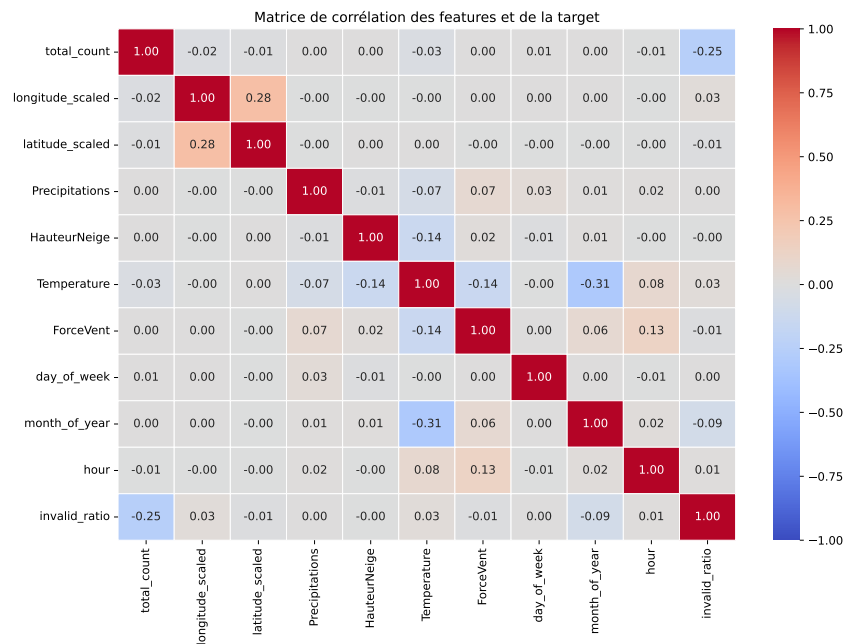


FIGURE 2 – Matrice de corrélation des données

2.3 Analyse des données en fonction de la cible

L'objectif de cette section est d'étudier la relation entre la variable cible et les autres variables du dataset. Globalement, l'analyse visuelle montre une **quasi-absence de corrélation** entre la cible et les variables explicatives, à quelques exceptions près.

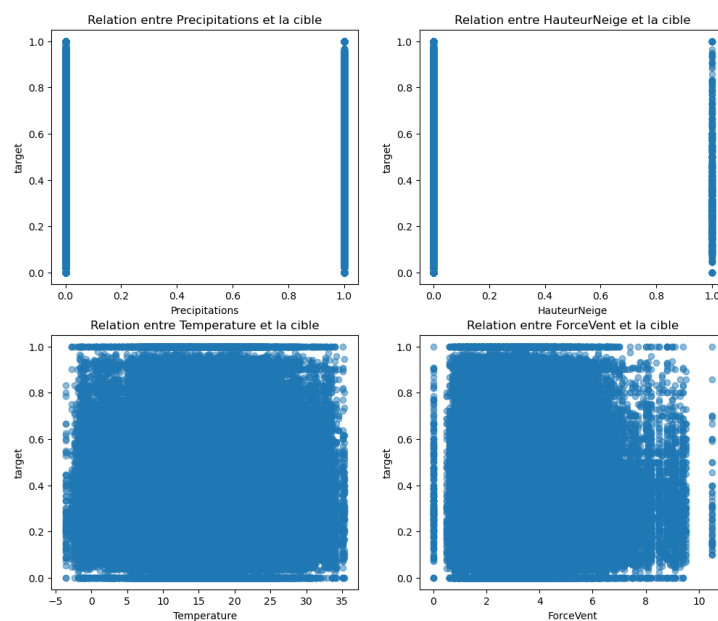


FIGURE 3 – Relation entre la variable cible et Precipitations, HauteurNeige, Temperature et ForceVent.

La figure 3 illustre cette absence de lien : aucune tendance claire ne se dessine entre la

cible et les variables `Precipitations`, `HauteurNeige`, `Temperature` ou `ForceVent`. On observe notamment que la présence ou l'absence de précipitations ne semble pas influencer les valeurs de la cible.

Effet de `total_count` sur la cible :

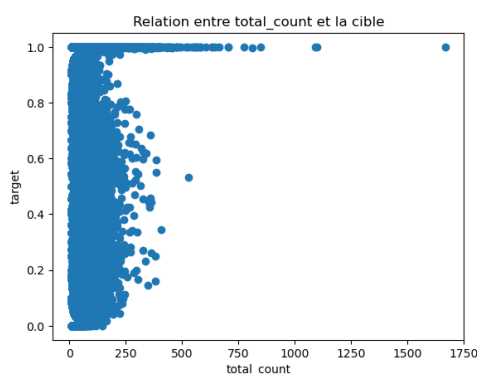


FIGURE 4 – Influence de `total_count` sur la cible.

Seule la variable `total_count` présente une légère corrélation avec la cible. Plus la valeur de `total_count` est élevée, plus la cible tend à prendre des valeurs importantes. Cela explique pourquoi cette variable est la seule à avoir une **corrélation initiale non négligeable** avec la cible.

Comme le montre la figure 4, cette corrélation n'est visible que pour les très hauts `total_count`. En dehors de ces cas particuliers, l'effet de `total_count` reste limité.

Influence des variables temporelles :

En étudiant les tendances temporelles, on constate que ni les jours ni les heures n'ont d'effet significatif sur la variable cible. Toutefois, une **tendance mensuelle décroissante** est visible : la cible diminue progressivement au fil des mois.

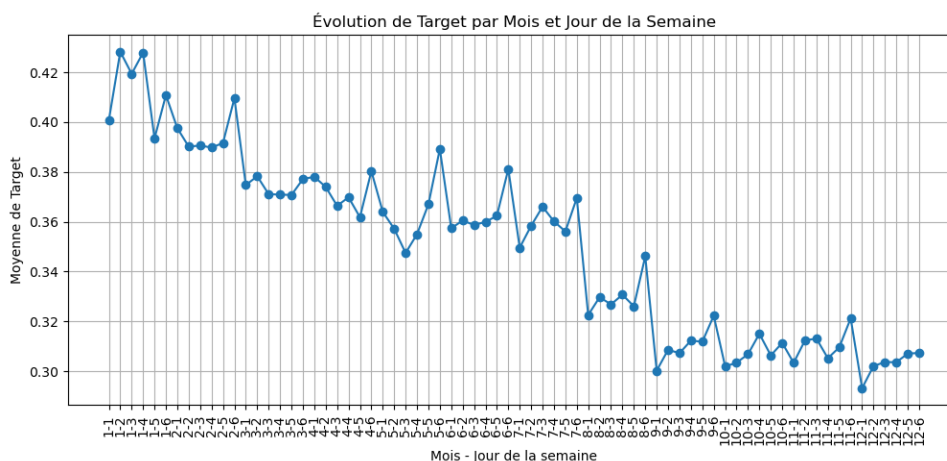


FIGURE 5 – Évolution de la moyenne de la cible au fil des mois.

Comme le montre la figure 5, la cible passe d'une moyenne de **0.42 lors du premier mois** à **0.30 au dernier mois**. Cependant, cette information est **peu exploitable** car l'ensemble `X_test` ne contient que 6 mois de données, contre 12 mois pour `X_train`. Il est donc risqué d'exploiter directement cette tendance, car elle ne serait **pas généralisable** sur l'ensemble de test.

2.4 Transformation des Features

Afin de rendre les données exploitables pour les modèles de Machine Learning, plusieurs transformations ont été appliquées aux variables.

Encodage des variables catégoriques :

Certaines variables catégoriques ont été transformées en indicateurs binaires grâce au `one-hot encoding` :

- `day_of_week` : Initialement représentée par un nombre entre 1 et 6, elle a été convertie en six colonnes binaires (`day_of_week_2`, `day_of_week_3`, etc.).
- `month_of_year` : Cette variable a été regroupée en catégories saisonnières : `season_Spring`, `season_Summer`, `season_Winter` et `season_Fall`.
- `hour` : Les heures ont été transformées en plages horaires significatives : `time_of_day_Morning`, `time_of_day_LateMorning`, `time_of_day_Afternoon`, etc.

Ces transformations permettent d'éviter de favoriser un horaire plutôt qu'un autre (comme dans du *label encoding* par exemple), et ainsi d'améliorer l'interprétabilité des modèles.

Transformation de Precipitations :

La variable `Precipitations`, initialement continue, a été convertie en une variable binaire indiquant simplement s'il y a eu des précipitations (`True`) ou non (`False`).

Création de nouvelles features :

- `high_total_count` : Une variable binaire a été ajoutée pour signaler les zones où `total_count` dépasse 500.
- **Rééquilibrage des catégories après encodage** : Les colonnes manquantes dans `X_test` après encodage ont été rajoutées avec une valeur de 0 pour assurer la cohérence avec `X_train`.

2.5 Nettoyage des Données

Avant l'entraînement des modèles, un prétraitement des données a été effectué afin de garantir leur qualité.

2.5.1 Gestion des valeurs manquantes

Le dataset d'entraînement contient **164 016 lignes** avec au moins une valeur manquante, soit **2.70%** de l'ensemble. Les variables les plus touchées sont :

Variable	Nombre de valeurs manquantes	Pourcentage
HauteurNeige	164 016	2.7%
ForceVent	6 082	0.1%

TABLE 2 – Résumé des valeurs manquantes

Comme la valeur `HauteurNeige` est presque toujours nulle, les valeurs manquantes ont été imputées à 0.

2.5.2 Suppression des lignes dupliquées

Un total de **29 lignes dupliquées** ont été supprimées du dataset.

2.5.3 Filtrage des valeurs extrêmes de `total_count`

Un filtre a été appliqué sur la variable `total_count` afin de supprimer les observations jugées non pertinentes. Initialement, un seuil de 45 était recommandé, garantissant un minimum de 45 véhicules par image. Cependant, bien que ce seuil ait produit de très bons résultats sur l'ensemble de validation, il a **négativement impacté** les performances sur l'ensemble de test réel.

Ainsi, nous avons ajusté ce paramètre, avec une performance optimale observée à `threshold = 10`, mais pour nos derniers tests, nous avons décidé de ne pas filtrer les données (`threshold = 0`).

2.5.4 Analyse de données de position mise à l'échelle

Le dataset comprend les informations de longitudes et latitudes de chaque relevé. Il est donc intéressant d'analyser cela afin de comprendre si il existe un lien entre le taux d'infraction et le lieu de contrôle.

Les données de positions ont été mises à l'échelle, on ne peut pas savoir précisément où les contrôles ont été effectués mais en affichant le graphique `longitude_scaled` x `latitude_scaled`, on observe rapidement une différence notable entre 3 endroits.

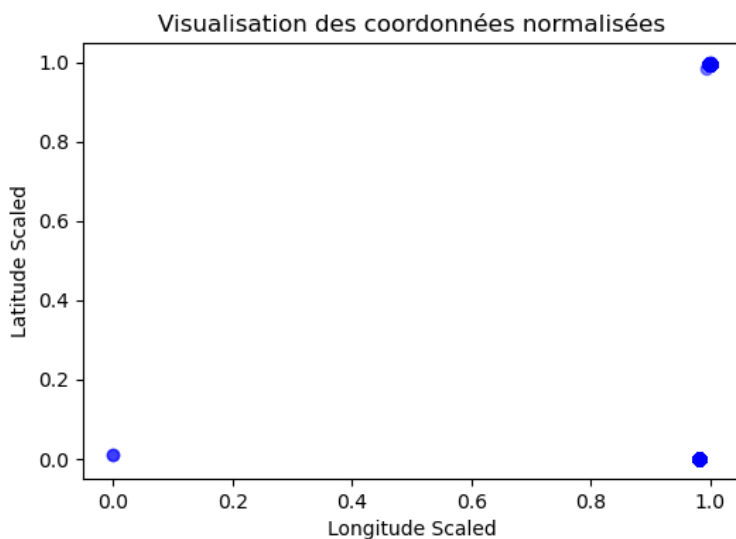


FIGURE 6 – Position des contrôles

Les positions avec une latitude ou une longitude inférieure à 0.1 sont négligeable, représentant même pas 0.001% des données.

Dans la suite, on va se concentrer sur les données de position supérieur à 0.9. En zoomant sur cette zone, on peut supposer qu'on obtient une ville ressemblant fortement à Paris. On s'est également assuré que les enregistrements de `x_test`, on également les mêmes propriétés.

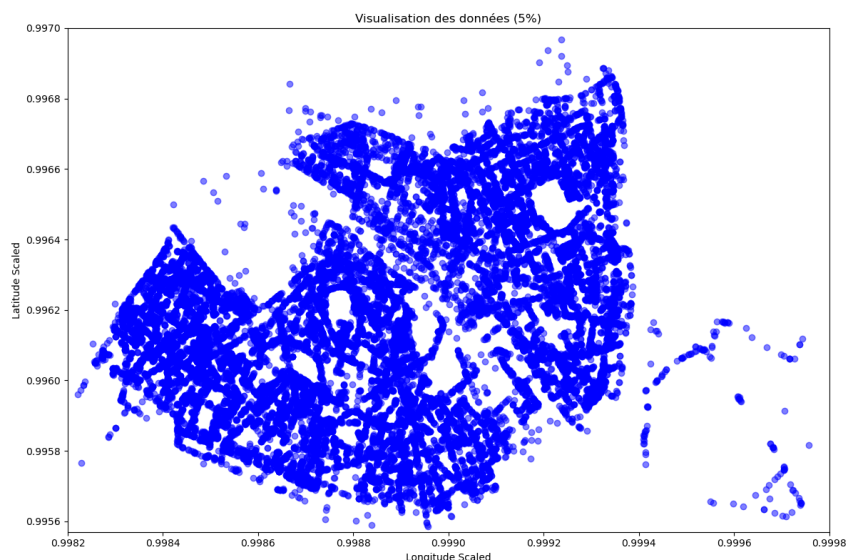


FIGURE 7 – Zoom sur les données de position retenues

2.6 Choix du nombre de voitures

Initialement, nous avons filtré les observations avec un `total_count` supérieur à 45, pensant améliorer la pertinence des données. C'était ce seuillage qui était proposé dans

l'introduction du sujet et utilisé dans le *benchmark*. Ce filtrage a entraîné une meilleure corrélation sur l'ensemble de validation, lui aussi filtré.

Cependant, sur l'ensemble de test (forcément sans filtrage), les performances se sont dégradées. Le modèle, entraîné sur des valeurs élevées de `total_count`, avait du mal à prédire les cas où ce nombre était faible, représentant pourtant une part importante des données réelles.

Un seuil trop élevé améliore l'apprentissage mais pénalise la généralisation. Après tests, `threshold = 10` semblait un bon compromis, mais pour éviter tout biais, nous avons finalement opté pour **l'absence de filtrage** (`threshold = 0`).

2.7 Normalisation des données

Afin d'améliorer la stabilité et la performance des modèles, une normalisation des données a été appliquée. Cette étape est particulièrement importante pour la variable `total_count`, qui joue un rôle clé dans la prédiction.

- **Transformation logarithmique** ($\log(1 + x)$) sur `total_count` : Cette variable présente une distribution très asymétrique, avec des valeurs extrêmes pouvant atteindre 1669. L'application d'un logarithme permet de **réduire l'effet des valeurs extrêmes**, de rendre la distribution "plus gaussienne" et d'améliorer l'apprentissage du modèle.
- **Standardisation** sur `Temperature` et `ForceVent` : Ces variables étant continues et de nature gaussienne, elles sont centrées-réduites ($\mu = 0$, $\sigma = 1$) pour faciliter leur intégration dans les modèles.
- **Variables binaires laissées inchangées** : Toutes les autres variables, déjà sous forme de catégories encodées en indicatrices, sont conservées telles quelles.

Ces transformations assurent une meilleure homogénéité des données et limitent l'influence des grandes valeurs sur les modèles de régression et les arbres de décision.

3 Clustering des données

Nous avons vu précédemment (cf. partie précédente du document) que l'immense majorité des points se situe dans la zone $[0.9, 1] \times [0.9, 1]$. Plus précisément, nous nous focalisons sur un sous-carré $(0.9982, 0.9998) \times (0.99557, 0.9970)$ qui, de façon assez inattendue, recouvre la grande majorité des données « propres ». Visuellement, cette zone ressemble fortement au plan de **Paris**, comme on l'a observé dans les graphiques présentés (voir figure 7).

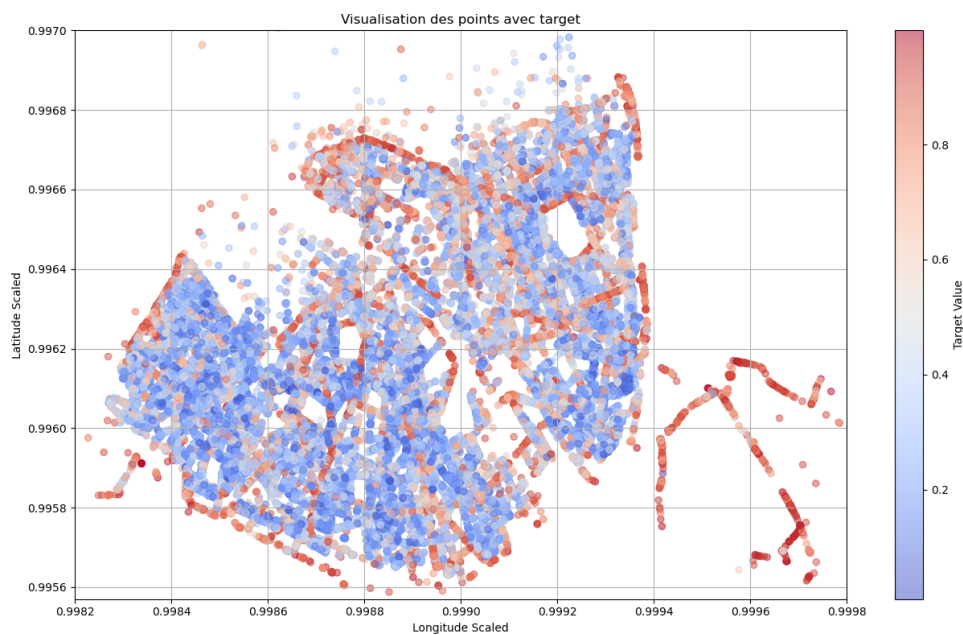


FIGURE 8 – Visualisation des valeurs de la variable cible dans Paris

Avant de détailler nos heuristiques, rappelons comment nous avons d'abord isolé les points en dehors de la zone $[0.9, 1] \times [0.9, 1]$. Nous avons baptisé ces points `isolated_data`. Au total, `isolated_data` représente environ 0.66% des 6 millions de points, et présente une moyenne de la cible (*target*) proche de 0.998127 ; c'est à la fois surprenant (valeur de *target* très élevée) et peu impactant sur le volume global. Nous avons alors conservé le reste sous le nom de `clean_data`. Bien que quelques points de `clean_data` se situent en dehors du petit carré $(0.9982, 0.9998) \times (0.9957, 0.9970)$, l'écrasante majorité réside dans ce carré principal « type Paris ».

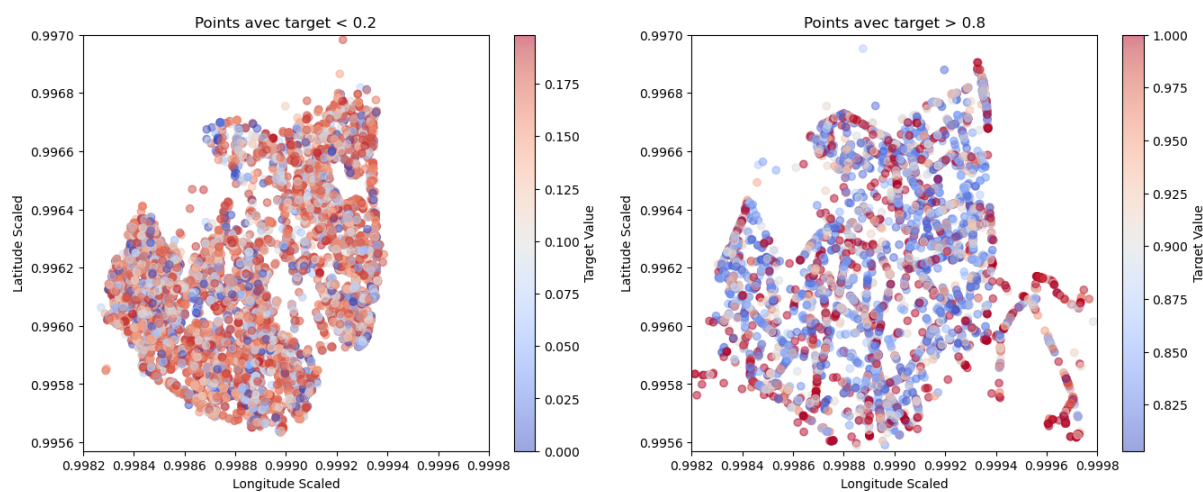


FIGURE 9 – Comparaison des valeurs de la variable cible dans Paris

En étudiant la *target* sur ce carré, nous constatons plusieurs faits (voir figure 8 et figure 9) :

- Lorsque la *target* est faible (disons, < 0.2), les points semblent assez diffus et ne présentent pas de motifs clairement apparents.
- Au contraire, quand la *target* est élevée (> 0.8), on observe de nombreux « segments » de points qui ressortent nettement. Ces segments correspondent probablement à des « rues ».
- De plus, la « couronne » périphérique de Paris, c'est-à-dire la bordure du bloc de points, est elle-même très rouge (*target* élevée). L'intérieur paraît plus bleu (cible plus faible), mis à part quelques lignes (rues) où la *target* reste forte.
- Enfin, la zone au sud-est (le bois de Vincennes) présente également une moyenne de *target* supérieure à 0.85, ce qui la distingue du reste.

Ces observations sont importantes pour définir des stratégies de **clustering**¹ adaptées. Nous proposons ci-après trois heuristiques, chacune travaillant sur un *échantillon* à 10% de `clean_data` pour réduire les temps de calcul, avant de propager le clustering au reste des données (et de façon symétrique à `x_test`).

3.1 Heuristique 1 : K-means simple

La première approche consiste simplement à :

1. Extraire un **échantillon** à 10% de `clean_data`.
2. Appliquer un **K-means** (nombre de clusters à fixer) sur cet échantillon.
3. Entraîner un `nn-model` (1 voisin) sur ces mêmes points pour *propager* les étiquettes de cluster à la totalité de `clean_data`.
4. Effectuer le même traitement sur `x_test`, en attribuant à chaque point le cluster de son plus proche voisin (1-NN) parmi les points échantillonnés.

Comme l'illustre la figure 10, on obtient alors un partitionnement pur « K-means » de la zone, sans aucune prise en compte des autres informations que nous avons pu avoir lors de l'analyse.

En prenant $n_{\text{clusters}} \approx 15$, on remarque tout de même qu'on retrouve plus ou moins la structure des différents arrondissements de Paris. Cette information nous a été très utile pour déterminer la valeur de n_{clusters} pour les autres heuristiques testées.

3.2 Heuristique 2 : Délimitation de bordure et K-means

Pour mieux tenir compte des particularités géographiques (la « bordure de Paris » très rouge, certaines zones comme Vincennes au sud-est, etc.), nous avons élaboré une heuristique plus complexe. Elle s'effectue en plusieurs étapes (toujours sur un échantillon à 10%) :

1. On parlera tantôt de « clustering », tantôt de « regroupement » des données.

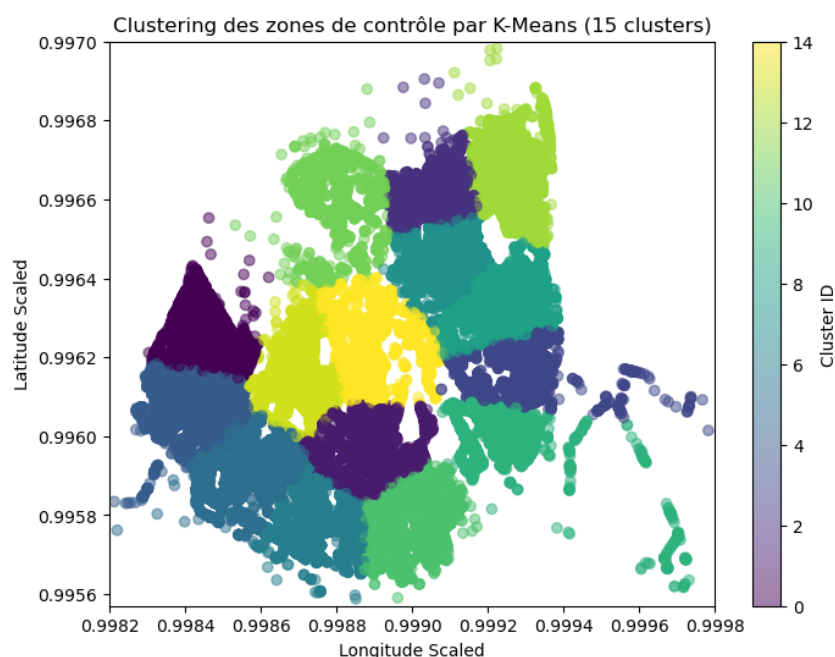


FIGURE 10 – Recherche du clustering par la première heuristique

1. **Déterminer le bloc principal** : on cherche la plus grosse composante de points (`main_block`) dans la zone, via la fonction `find_main_block_grid`. Celle-ci découpe l'espace en une grille de taille `cell_size` donnée (cf. figure 11), forme un graphe de cellules adjacentes non vides, et isole la plus grande composante connexe. Les points `others` (hors bloc) sont assez peu nombreux.

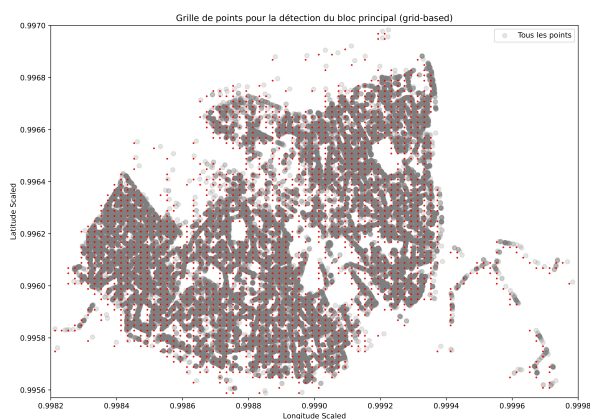


FIGURE 11 – Approche *grid-based* pour déterminer le `main_block`

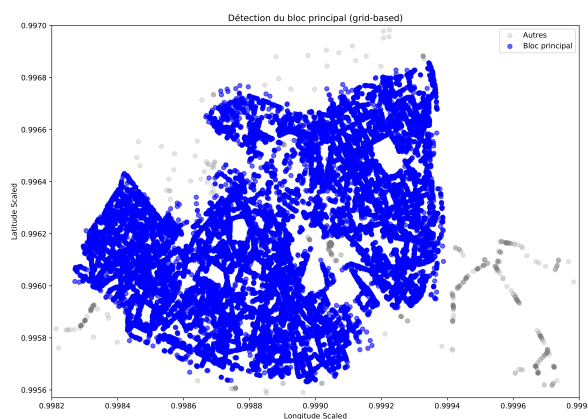


FIGURE 12 – Visualisation du `main_block`

Le résultat de la détection du "bloc principal", c'est-à-dire la ville de Paris intra muros, est donné dans la figure 12.

2. **Extraire la bordure** : Ensuite, on calcule l'enveloppe convexe (`ConvexHull` de `scipy.spatial`) du bloc principal, puis on considère comme « `bordure` » tous les

points suffisamment proches de cette *barrière convexhull*. L'idée est de capturer la « couronne de Paris », très rouge en *target*.

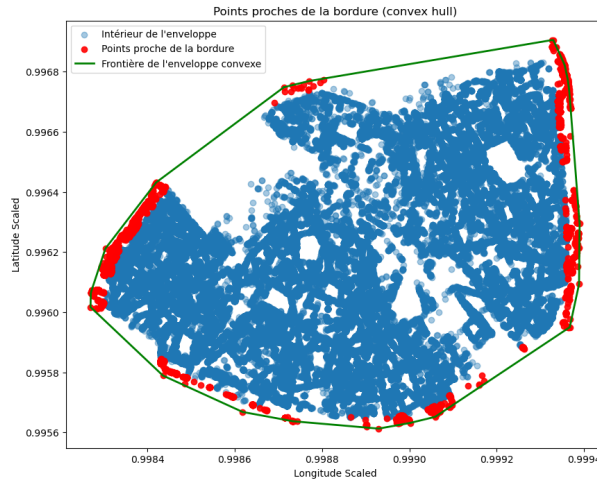


FIGURE 13 – Recherche des points proche de la bordure du bloc principal

3. **K-means sur le bloc intérieur** : une fois la bordure retirée, on applique un K-means (*aussi* sur l'échantillon) au cœur de la ville (`bloc_principal - bordure`). On choisit un nombre de clusters adapté.
4. **Séparation Nord/Sud pour les points restants** : pour les points plus dispersés au nord ou au sud, nous introduisons une « frontière » arbitraire en `latitude_scaled = 0.99625`. Les points sous cette ligne sont regroupés (zone sud, ex. Vincennes), ceux au-dessus sont zone nord. Le résultat est donné dans la figure 14.

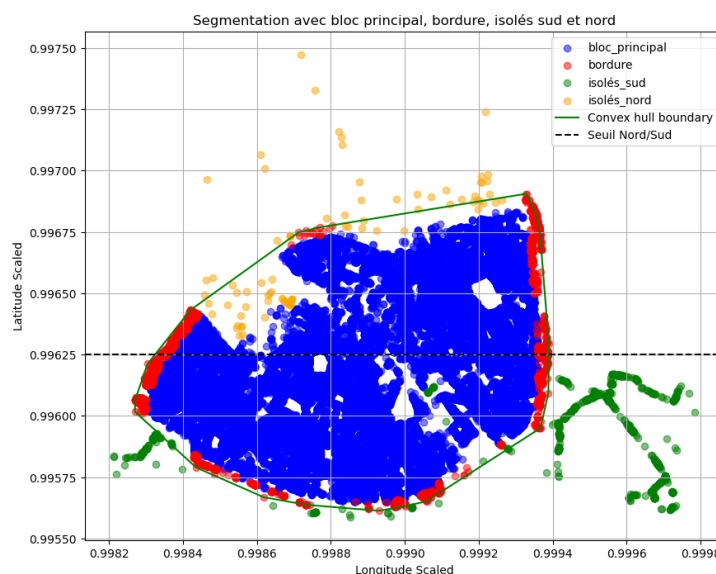


FIGURE 14 – Finalisation du clustering avec le partitionnement Nord/Sud

5. **Propagation à toutes les données** : comme dans l'heuristique 1, on utilise un

`nn-model` (1 voisin) pour étiqueter *toute* `clean_data` puis `x_test`.

Enfin, compte tenu du très faible nombre de points dans `isolated_data` et de leur *target* moyenne très élevée, nous les avons placés dans le cluster « `zone_sud` », car c'est celui qui affichait la moyenne la plus forte. On peut ainsi afficher la moyenne *target* de chaque cluster (voir figure 15) : on obtient clairement plus de 0.8 dans ce cluster `zone_sud` + `isolated_data` (le cluster 17).

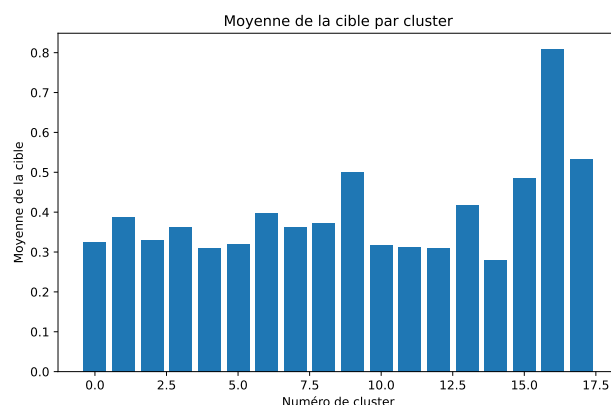


FIGURE 15 – Histogramme des valeurs cibles des données groupées par cluster

Ainsi, comme l'illustre la figure 16, le résultat final est plutôt convainquant et permet de mettre en lumière quelques propriétés du jeu de données que nous avons eu lors de l'analyse (comme la présence de valeurs cibles très élevées sur la bordure de Paris).

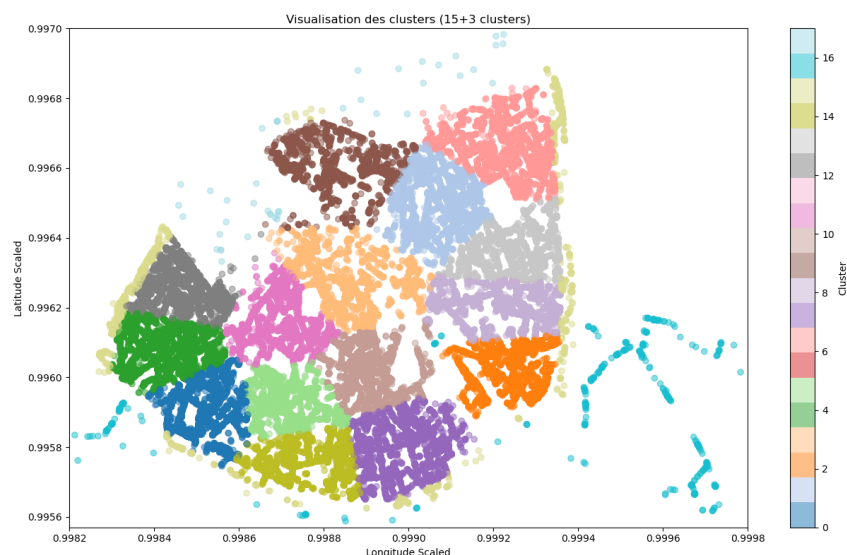


FIGURE 16 – Résultat final du clustering par la deuxième heuristique

3.3 Heuristique 3 : Variante de l'heuristique 2

Dans cette dernière heuristique, nous reprenons exactement la même logique :

- trouver le `main_block`, la `bordure`, séparer le nord et le sud, etc.
- appliquer un K-means sur l'intérieur (avec un nombre de clusters plus grand, typiquement 50).

Puis, juste avant la fin, nous *fusionnons* les nombreux clusters issus du K-means, en nous basant sur leur *target* moyenne. Autrement dit, nous les ordonnons par valeur moyenne de *target* et les regroupons (par exemple en 10 bins), afin d'obtenir un nombre final de groupes plus restreint (10 au lieu de 50), tout en maintenant l'idée de séparation forte entre « clusters à *target* faible » et « clusters à *target* élevée ». Nous appliquons ensuite la méthode 1-NN pour projeter ces clusters unifiés sur l'intégralité de `clean_data` puis sur `x_test`.

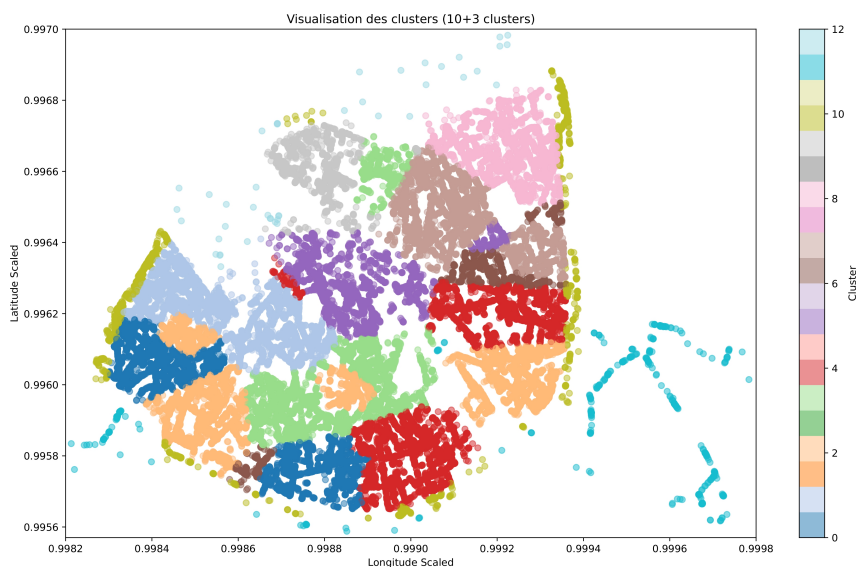


FIGURE 17 – Résultat final du clustering par la troisième heuristique

Le résultat de ce clustering est donnée dans la figure 17. Et ainsi, l'heuristique 3 se veut un raffinement de l'heuristique 2, tirant parti de *nombreux* clusters initiaux (pour capturer la complexité spatiale), tout en résumant cette granularité via **une unification** par déciles de la *target*.

3.4 Résumé de nos 3 heuristiques

Nous obtenons donc, au final, trois manières de **partitionner** les données :

- **Heuristique 1 (K-means simple)** : plus rapide, très standard.
- **Heuristique 2 (bordure + K-means + séparation nord/sud)** : exploite mieux la géographie (bordure rouge, bas-sud est).
- **Heuristique 3 (fusion finale par déciles de *target*)** : affine la répartition en créant moins de clusters *finalement*, tout en conservant l'idée d'une segmentation spatiale.

Dans la suite du document, nous comparerons les performances de ces heuristiques sur différents indicateurs de prédiction et d'interprétabilité.

3.5 Corrélation finale

Après l'étape de clustering, nous pouvons analyser la corrélation finale des différentes **features** sélectionnées.

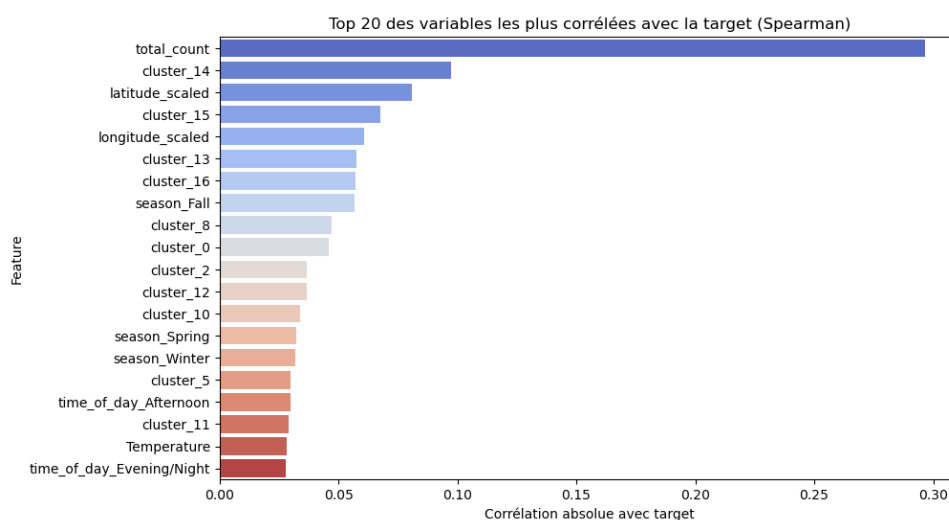


FIGURE 18 – Top 20 des variables les plus corrélées avec la target (Spearman)

Comme le montre la figure 18, en dehors de la variable `total_count`, ce sont principalement les **données issues du clustering** qui présentent une corrélation significative avec la cible. Cela est plutôt rassurant sur notre travail pour effectuer un clustering efficace.

On observe également la présence de certaines variables créées précédemment, notamment les indicateurs saisonniers comme `season_Automne` et `season_Printemps`.

4 Modèles et évaluation des performances

4.1 Métrique d'évaluation

Pour évaluer nos modèles, on utilise **la corrélation de Spearman**.

Définition :

La corrélation de Spearman, notée ρ , est une mesure non paramétrique de la relation entre deux variables. Elle est définie comme le coefficient de corrélation de Pearson appliqué aux **rangs** des valeurs, plutôt qu'aux valeurs elles-mêmes. Le rang étant la position relative d'une donnée dans le dataset lorsqu'on trie les valeurs par ordre croissant.

Sa formule est donnée par :

$$r_s = \frac{\text{cov}(\text{rg}_X, \text{rg}_Y)}{\sigma_{\text{rg}_X} \sigma_{\text{rg}_Y}}$$

où rg_X et rg_Y représentent les rangs des variables X et Y , cov est la covariance, et σ désigne l'écart-type. La valeur de r_s varie entre -1 (relation strictement décroissante) et +1 (relation strictement croissante), avec 0 indiquant l'absence de relation monotone.

Cette mesure est particulièrement robuste aux valeurs aberrantes et est souvent utilisée lorsque les données ne suivent pas une distribution normale. C'est clairement le cas ici, et l'objectif est donc de **faire tendre cette corrélation vers 1** pour les données tests présentes dans X_{test} .

4.2 Modèles utilisés

Nous avons testé plusieurs modèles de Machine Learning afin de prédire la variable cible. Chaque modèle a été optimisé à l'aide de la validation croisée et d'une recherche d'hyperparamètres.

Il faut noter que c'est **l'heuristique 2 qui a obtenu les meilleurs résultats**. Assez surprenant puisque nous pensions que la troisième heuristique de clustering pourrait arriver à de meilleurs résultats, mais elle laissait finalement trop de place au surapprentissage.

4.2.1 Régressions linéaires (Ridge et Lasso)

Les modèles de régression linéaire, incluant **Ridge** et **Lasso**, ont été appliqués avec un réglage optimal de leur coefficient de régularisation λ . Malgré ces optimisations, ils atteignent un **score de Spearman autour de 0.35** sur l'ensemble de validation, montrant leurs limites face aux relations non linéaires des données.

4.2.2 Support Vector Machine (SVM)

Un modèle **SVM** avec un noyau RBF a été testé. Après une optimisation des paramètres γ et C , les meilleurs résultats obtenus étaient de l'ordre de **0.40 à 0.42** en validation, montrant un léger gain par rapport aux modèles linéaires, mais restant inférieur aux méthodes basées sur les arbres.

On aurait pu essayer de s'intéresser davantage à ce modèle, mais les modèles basés sur les arbres ou sur du *boosting* nous a semblé plus pertinent.

4.2.3 Random Forest

L'algorithme **Random Forest**, basé sur un ensemble d'arbres de décision, s'est révélé plus performant. Avec une recherche d'hyperparamètres, nous avons obtenu **une corrélation de 0.50 à 0.52** en validation avec les paramètres suivants :

- Nombre d'arbres : 500

- **Profondeur maximale** : 20
- **Nombre minimal d'échantillons par feuille** : 5
- **Fraction de caractéristiques sélectionnées** : 0.8

4.2.4 XGBoost

XGBoost s'est imposé comme le modèle le plus performant. Après optimisation, nous avons atteint **une corrélation de 0.56 à 0.58** en validation avec les hyperparamètres suivants :

- **subsample** : 1.0
- **n_estimators** : 300
- **max_depth** : 12
- **learning_rate** : 0.05
- **lambda** : 1, **gamma** : 0
- **colsample_bytree** : 1.0, **alpha** : 0

4.2.5 Moyenne des modèles

Une approche combinant **Random Forest et XGBoost** a été testée en effectuant la moyenne des prédictions des deux modèles. Cette méthode a permis une légère amélioration, avec une augmentation de la corrélation de Spearman d'environ **0.01 point**.

5 Résultats

5.1 Influence du filtrage des données

Les performances des modèles sont fortement influencées par le choix du seuil appliqué à `total_count` (voir section 2.6). Les résultats présentés ici sont obtenus **sans filtrage** pour être en cohérence avec l'ensemble `X_test`. Cependant, en appliquant un seuil à 45, **XGBoost peut atteindre des performances de l'ordre de 0.80** sur l'ensemble de validation.

5.2 Résultats sur le dataset public

Avec **XGBoost optimisé**, nous avons atteint une **corrélation de 0.492** sur le dataset public, avec les hyperparamètres mentionnés précédemment.

5.3 Résultats sur le dataset privé

Le résultat du dataset privé a été entraîné avec une variante légèrement différente des hyperparamètres de XGBoost :

- **subsample** : 0.8, **n_estimators** : 800, **max_depth** : 15
- **learning_rate** : 0.03, **lambda** : 1, **gamma** : 0
- **colsample_bytree** : 0.8, **alpha** : 0

Avec ce modèle, on a obtenu une **corrélation de 0.432 sur l'ensemble privé**, contre 0,4329 sur l'ensemble publique, confirmant l'absence de sur-apprentissage.

5.4 Classement final

Dans le classement du challenge :

- Nous avons terminé **3ème** dans le classement académique public.
- Nous sommes **2ème** dans le classement privé.
- Nous avons obtenu la **7ème place** au classement général.

6 Idées d'améliorations

6.1 Utilisation de RANSAC

Dans une tentative d'amélioration des résultats, nous avons cherché à exploiter la structuration spatiale des hautes valeurs de la cible. Comme l'illustre la figure 19, ces valeurs forment clairement des segments distincts dans Paris. En les isolant, nous avons appliqué un algorithme de type RANSAC afin de les détecter automatiquement.

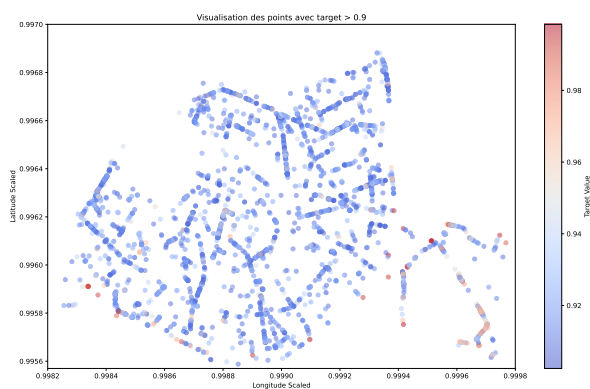


FIGURE 19 – Visualisation des points à très haute valeur cible (> 0.9)

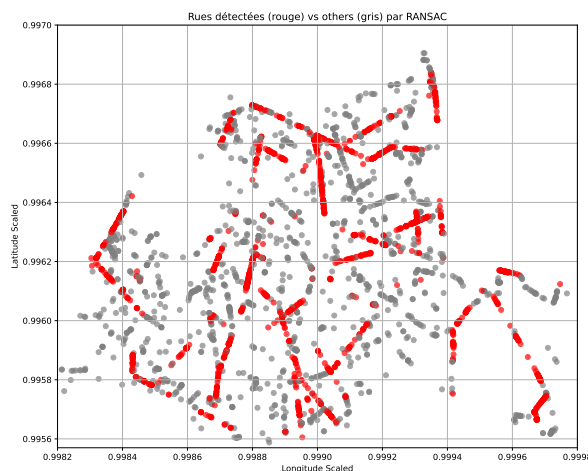


FIGURE 20 – Résultats de l'algorithme RANSAC

La figure 20 montre que cette approche permet effectivement d'identifier ces segments à fort taux d'infraction. Toutefois, lorsque l'ensemble des données est considéré, la diffusion des points à faible valeur de cible empêche une détection robuste de ces structures.

Bien que largement perfectible, cette approche pourrait être développée pour améliorer la prédiction des zones à fort taux d'infraction. Notre modèle final a obtenu un **score de Spearman de 0.49**, tandis que les meilleurs participants du challenge atteignaient **0.58**. Il est possible qu'ils aient réussi à mieux isoler ces segments, augmentant ainsi la corrélation.