# Description bref de la solution IA

## Introduction

La solution IA chez Tedies est conçue dans le but d’identifier destinataire dans la base des données à partir d’une adresse EDI reçue (adresse de livraison) qui contient le nom et adresse postale du destinataire.

La manière directe pour le faire est de comparer le nom et l’adresse postale de la livraison avec tous les noms et adresses postales des destinataires préenregistrés dans la base de données et choisir le destinataire qui correspond exactement avec l’adresse de livraison. Bien sur cette méthode est celle qui est toujours utilisée aujourd’hui et afin d’avoir une qualité élevé de correspondance, Tedies a fait en sorte de garder pour chaque destinataire toutes les versions d’adresses déjà reçues et qui sont appelées les alias de destinataires. Cette manière de faire permet d’augmenter la chance d’identification d’un destinataire par un de ces alias en prenant en compte que des destinataires préenregistrées ont tendance à avoir les mêmes alias d’adresses qui se répètent à travers le temps.

La solution IA vient complémenter cette méthode pour identifier les adresses que cette dernière n’a pas pu identifier. En proposant l’encodage des adresses en vecteurs sémantiques, les alias d’adresses d’un destinataire donnée sont supposés avoir des vecteurs similaires en termes de distance vectorielle. En conséquent, si une nouvelle adresse de livraison est reçue et son destinataire existe dans la base, le vecteur de cette nouvelle adresse est supposé être plus proche aux vecteurs d’alias du destinataire en question par rapport à d’autres vecteurs d’alias de destinataires différents.

## Fonctionnement de la solution IA

La solution IA est une méthode ensembliste qui utilisent plusieurs modèles de récupération de documents et agrègent leurs résultats en une seule liste finale de candidats d’adresses similaires. L’algorithme utilisé pour récupérer les listes est un kNN avec comme mesure de calcul des distances le cosinus et une valeur de k pour fixer le nombre de candidats récupérés.

### La solution IA se compose de 2 parties essentielles

* **La partie encodage des adresses :** Plusieurs modèles de Transformers spécialisés sont entrainés à classifier des paires d’adresses labélisées (adresse livraison, adresse destinataire) en « match » ou « nom match ». Cet entrainement par la suite permet d’utiliser les Transformers pour encoder toutes les adresses des destinataires en vecteurs sémantiques et ainsi construire des datastores de référence respectives. Chaque Transformers représente un type d’encodage différent.
* **La partie récupération des candidats :** 
  + Une fois nos datastores conçus, nous pouvons encoder une adresse de livraison nouvelle en vecteurs d’encodages respectifs.
  + Nous récupérons ensuite via le kNN pour chaque vecteur de livraison dans un datastore donné, les meilleurs vecteurs les plus proches à ce dernier. En principe, ces vecteurs récupérés pointent sur le destinataire de la livraison reçue s’il existe déjà.
  + Cette opération nous permet d’avoir une liste de candidats pour chaque type d’encodage. Nous appliquerons par la suite une technique d’agrégation des différentes listes pour construire une liste finale de candidats triés selon les adresses les plus similaires.

### Le fonctionnement de la technique d’agrégation

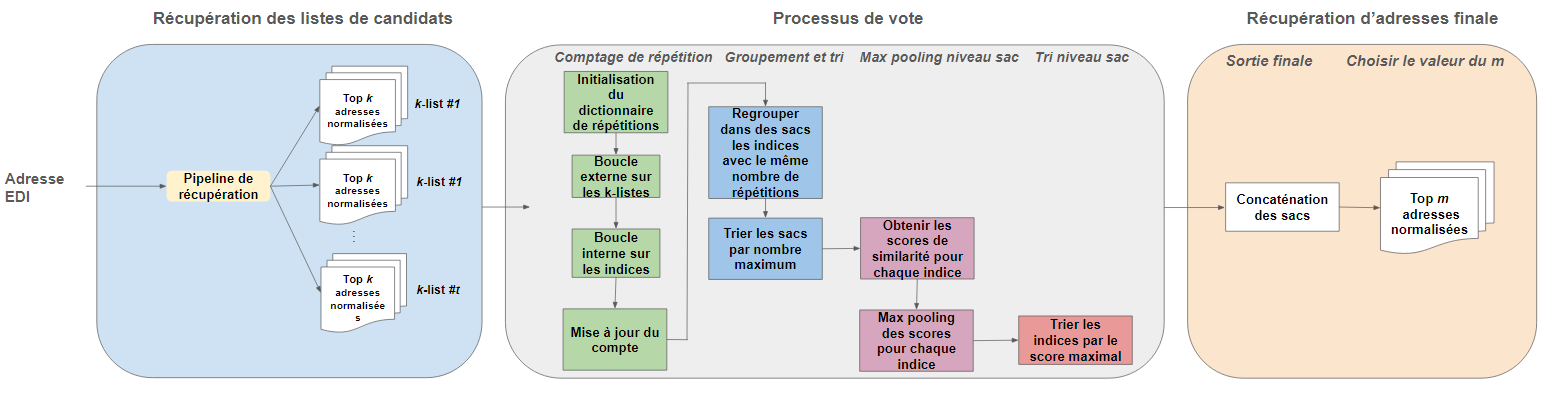
Cette technique se base sur deux principes :

* Le nombre de répétition d’une adresse de destinataire ou un de ces alias à travers les différentes listes récupérées.
* Les scores de similarités cosinus associés à chaque adresse destinataire ou un de ces alias à travers les différentes listes récupérées.

#### Stratégies d’agrégation

##### Stratégie 1

## Celle du papier qui se base sur le max pooling des scores de similarité pour le tri des adresses.



###### 1. Récupération des listes d'adresses candidates

Le système commence par récupérer les 𝑘-listes d'adresses candidates en utilisant le pipeline de récupération ensembliste 𝑘NN. Chaque modèle de l'ensemble fournit une liste d'indices d'adresses du datastore pour une requête donnée (adresse de livraison reçue).

###### 2. Processus de vote

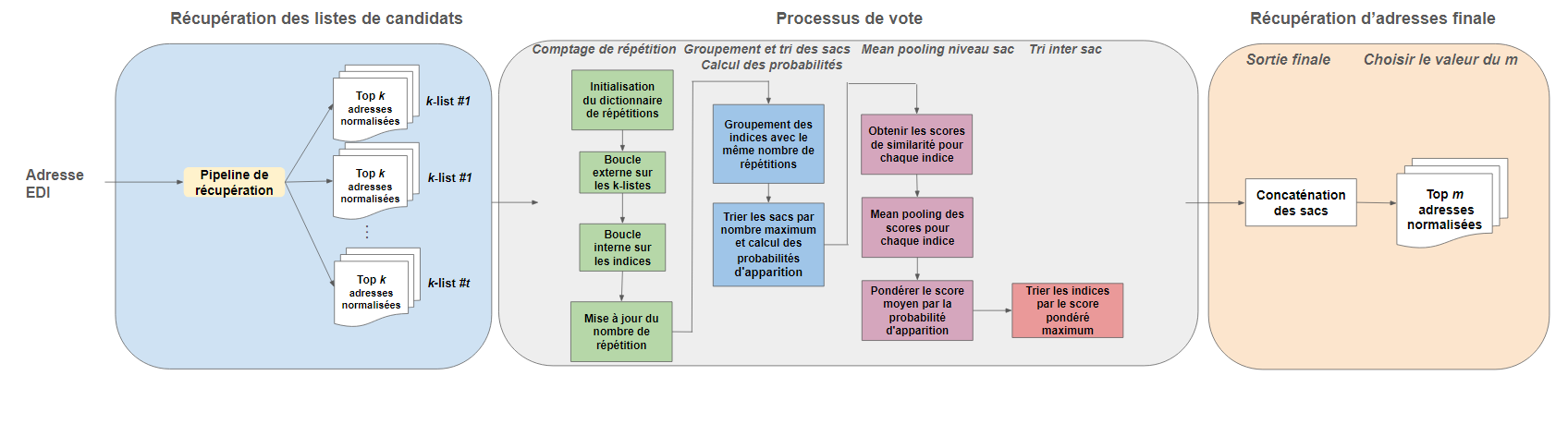
* **Comptage des répétitions :** La première étape du processus de vote consiste à compter le nombre de répétitions de chaque adresse candidate 𝑖 à travers les différentes 𝑘-listes.
* **Regroupement et tri :** Les indices des candidats sont ensuite regroupés en fonction de leurs comptages de répétitions. Cela crée des "sacs" d'indices, où chaque sac contient un ou plusieurs indices pointant vers des adresses associées. Ensuite, les sacs sont triés par le nombre maximum de répétitions associé.
* **Max pooling des scores de similarité dans les sacs :** Au sein de chaque sac, le système collecte les scores de similarité pour chaque adresse provenant des différents modèles 𝑘NN dans lesquels ils sont apparus. Le max pooling est ensuite appliqué à ces scores pour déterminer le score de similarité maximum pour chaque adresse dans le sac.
* **Classement dans les sacs :** Les adresses sont ensuite triées dans chaque sac en fonction de leurs scores de similarité maximum.

### 3. Récupération finale de la liste d'adresses

* **Sortie finale :** Tous les sacs sont concaténés, ce qui donne une liste triée d'adresses où l'adresse candidate principale a été répétée le plus grand nombre de fois et possède le score de similarité le plus élevé.
* **Choix de la valeur de coupure :** Le système définit la valeur de 𝑚 (le nombre de voisins à retourner). La valeur de 𝑚 n'est pas nécessairement la même que celle de 𝑘, car le processus de vote classe finalement tous les candidats des 𝑘-listes combinées, ce qui produit naturellement des candidats supplémentaires en fonction de l'hétérogénéité des 𝑘-listes.

##### **Stratégie 2**

Cette stratégie est un peu différente car elle se base sur le mean pooling des scores de similarité multiplié par la probabilité de répétition.



### 1. Récupération des listes d'adresses candidates

Le système commence par récupérer de la même manière les 𝑘-listes d'adresses candidates en utilisant le pipeline de récupération ensembliste 𝑘NN. Chaque modèle de l'ensemble fournit une liste d'indices d'adresses pour une requête donnée.

### 2. Processus de vote

* **Comptage des répétitions :** La première étape du processus de vote consiste à compter le nombre de répétitions de chaque adresse candidate 𝑖 à travers les différentes 𝑘-listes.
* **Regroupement et tri:** Les indices des candidats sont ensuite regroupés en fonction de leurs comptages de répétitions. Cela crée des "sacs" d'indices, où chaque sac contient un ou plusieurs indices pointant vers des adresses associées. Ensuite, les sacs sont triés par le nombre maximum de répétitions associé.
* **Calcul des probabilités :** Chaque sac se voit attribué ensuite une probabilité d’apparition ou de répétition qui est calculée comme suite : nombre de répétition associé / le nombre de k-listes.
* **Mean pooling des scores de similarité pondéré dans les sacs :** Au sein de chaque sac, le système collecte les scores de similarité pour chaque adresse provenant des différents modèles 𝑘NN dans lesquels ils sont apparus. Le mean pooling est ensuite appliqué à ces scores pour déterminer le score de similarité moyen pour chaque adresse dans le sac. Ensuite chaque score moyen est pondéré par la probabilité d’apparition associée au sac.
* **Classement inter sac :** toutes les adresses sacs confondus seront triées en fonction des scores de similarités pondérées.

### 3. Récupération finale de la liste d'adresses

* **Sortie finale :** Nous aurons une liste finale triée d'adresses où l'adresse candidate principale aura le plus grand score de similarité pondérée.
* **Choix de la valeur de coupure :** Le système définit la valeur de 𝑚 (le nombre de voisins à retourner). La valeur de 𝑚 n'est pas nécessairement la même que celle de 𝑘, car le processus de vote classe finalement tous les candidats des 𝑘-listes combinées, ce qui produit naturellement des candidats supplémentaires en fonction de l'hétérogénéité des 𝑘-listes.

### Echantillon de test pour la méthode d’agrégation

Nous allons prendre un échantillon d’adresse de livraisons reçues depuis le 05/07/2024 et pour éviter des biais dans les résultats, nous prenons nos datastore jusqu’au 04/07/2024.

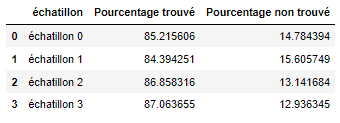
Nous construisons 5 échantillon de 487 adresses chacun à partir de l’échantillon original. Nous allons laisser 1 échantillon pour le test. Nous voulons avoir une idée plus claire sur:

* Comment choisir le meilleur prétendant parmi la liste finale des candidats.
* Quand ne pas choisir un prétendant et assumer que le destinataire de l’adresse reçue n’existe pas et il faudrait le créer.

#### Résultats Stratégie 2

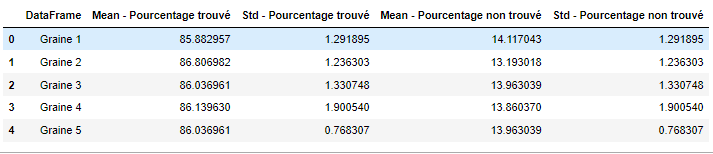
##### Premier rang :

Si nous nous infligeons une condition stricte de ne choisir que le premier candidat dans la liste finale, nous aurons les résultats globaux suivants :



Nous remarquons qu’en général, on a un pourcentage d’existence d’adresses correctes d’à peu près 85% des destinataires recherchés. Si nous générons plusieurs graines des 4 échantillons, nous aurons les résultats globaux suivants :

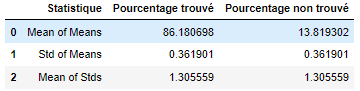
Statistiques par graine :



Analyse :

* Les moyennes des échantillons par graine des adresses trouvées varient entre 85.88% et 86.31%, ce qui indique en général, un taux de récupération correcte assez élevé. Cela suggère que le système reste assez consistant à travers les variations.
* Les écarts type des adresses trouvées sont assez bas (de 0.77 à 1.90), ce qui signifie qu’il n’y a pas trop de fluctuation dans les taux de récupération à travers les graines de test. Nous remarquons quand même un écart type relativement élevé pour la graine 4 mais reste tout de même modéré par rapport à sa moyenne.

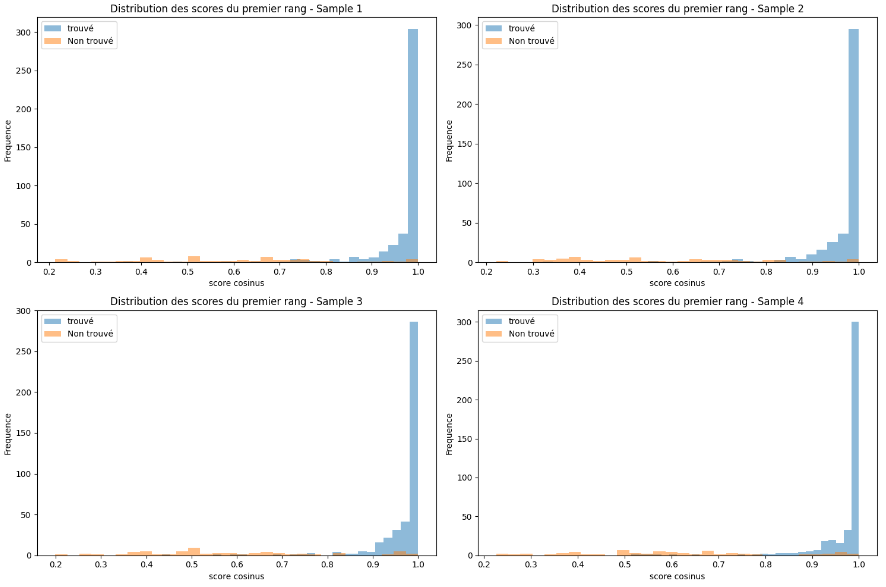
Statistiques globale :



Analyse :

* La moyenne des graines et de 86.18 pour les adresses trouvées, ce qui est un assez bon résultat à travers les différentes graines.
* L’écart type de cette moyenne est de 0.36, ce qui renforce la consistance du système à travers les graines. Nous remarquons également que la valeur de l’écart type global est assez petite ce qui indique que la performance ne varie pas trop d’une graine à l’autre.
* La moyenne des écarts type des graines reste assez basse et indique la stabilité relative des graines de tests individuels.

Nous allons maintenant visualiser les scores de similarité cosinus associés aux adresses récupérées au premier rang. Bien sûr, nous avons déjà les identifiants des adresses de livraisons qui nous permettent de savoir si nous avons trouvé le bon destinataire ou non :



Comme nous pouvons l’observer, nous remarquons que la majorité des destinataires trouvés en bleu ont un score de similarité pondéré supérieur à 0.90. Cela n’exclut pas que certaines adresses de destinataires trouvées ont un score de similarité nettement inférieurs à 0.90 jusqu’à arriver à 0.51 pour certaines.

Quelques exemples de certaines adresses trouvées avec un score pondéré en dessous de 0.90 :

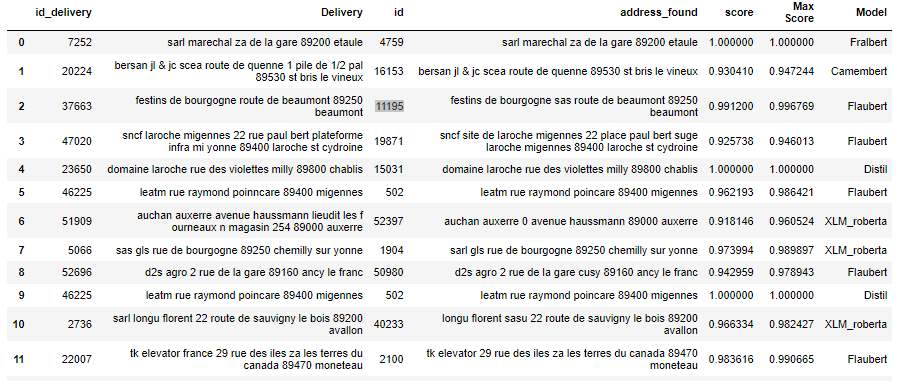


Dans le tableau, à droite nous avons le score de similarité pondéré et le score de similarité maximum entre les différents types d’encodage. Le problème ici est que si nous prenons un seuil d’acceptation sur le score pondéré disant « > 0.90 », nous ne pouvons pas prendre les adresses correctes avec un score en dessous. Si nous observons le score maximum, nous remarquons qu’il est plus élevé mais ce score est un score provenant d’un seul type d’encodage. Nous pourrons le considérer comme un score suffisant avec un seuil choisi mais dans ce cas-là, il faut un seuil d’acception pour chaque type d’encodage.

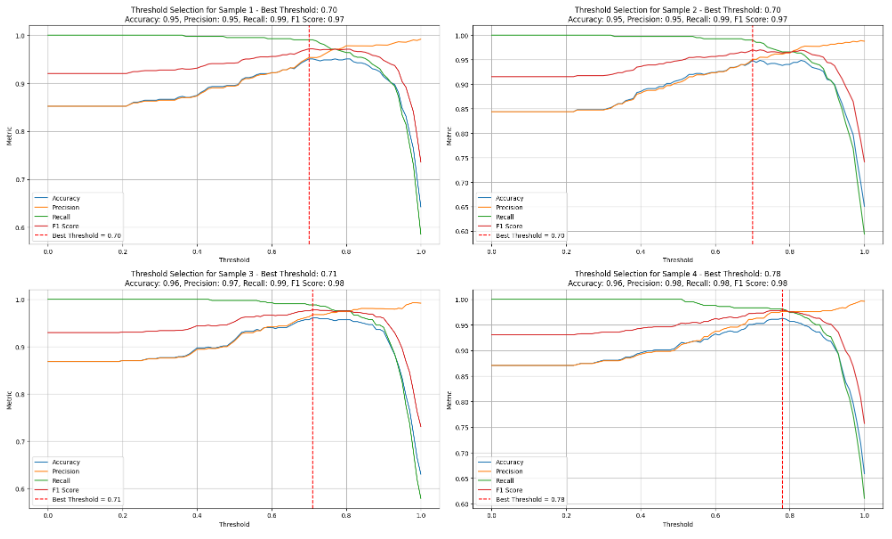
*« La question est de trouver une méthode qui nous permettrait de prendre ou pas la décision d’accepter une des adresse de destinataire dans la liste comme « match » avec l’adresse de livraison. »*

Inversement dans les graphes, nous remarquons aussi que certaines adresses non trouvées en orange ont un score pondéré très élevé (> 0.90 aussi). Après investigation, nous nous apercevons que ces adresses sont des doublons dans la base des données et qu’en réalité, ce sont des bonnes réponses. Ces doublons biaiserons toujours les résultats car pour l’instant, nous n’avons pas réussi à tous les fusionner.

Quelques exemples de certaines adresses non trouvées avec un score au-dessus de 0.90 sur le score pondéré :

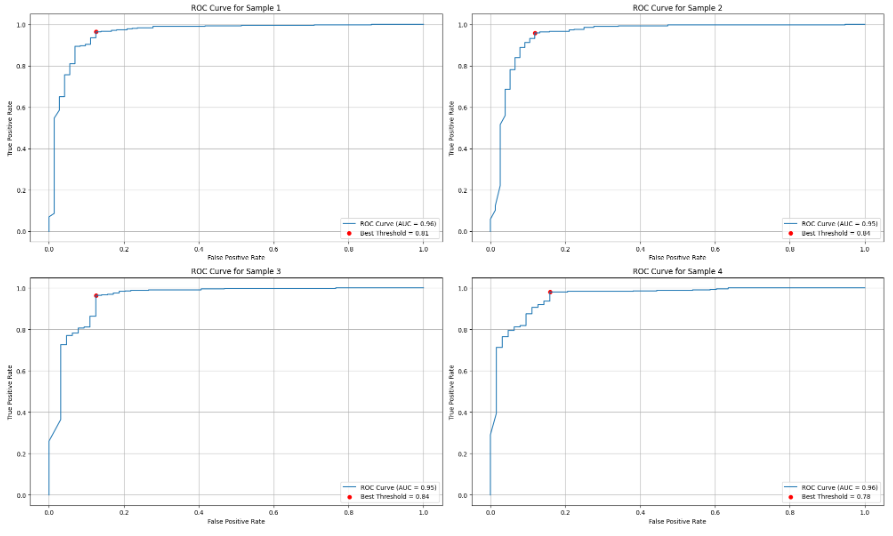


Nous avons effectué une estimation de seuil d’acceptation pour le score pondéré sur les scores des premiers rang. Nous calculons les metrics suivantes : accuracy, précision, rappel et F1 score et nous choissons le seuil qui produit le meilleur F1 score. Les graphes suivant montrent les résultats pour nos 4 échantillons :



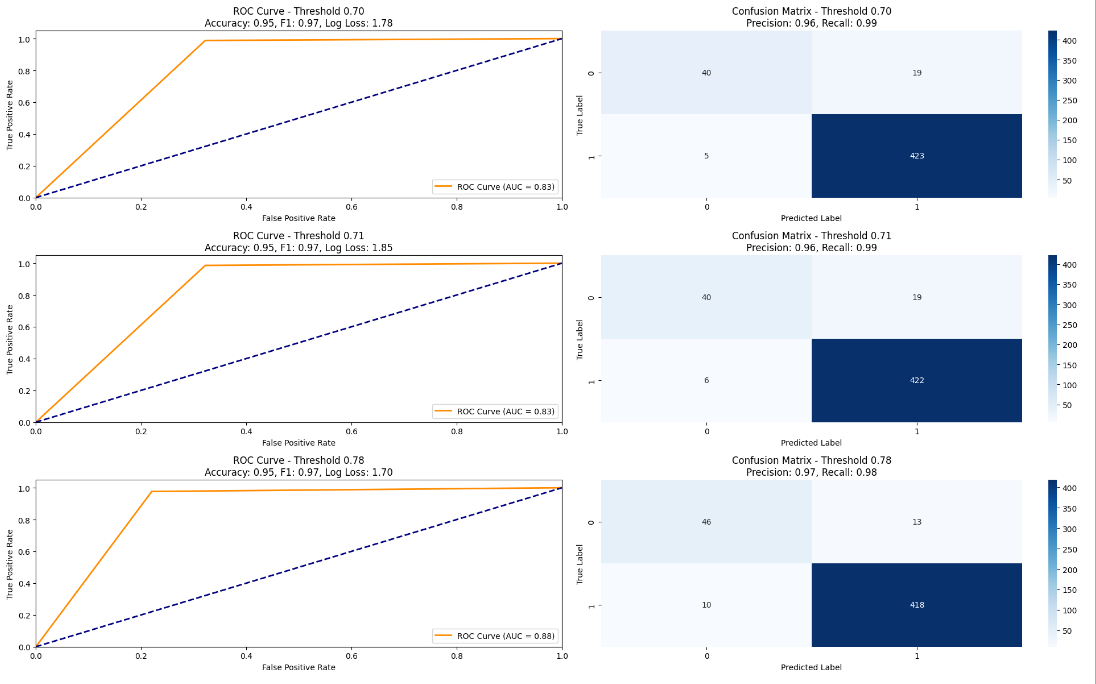
Ici le seuil varie entre 0.70, 0.71 et 0.78 ce qui n’est pas très optimal. Nous remarquons également, le F1 score très élevé pour ces seuils optimaux en haut des graphes, ce qui nous encourage à croire qu’un seuil de validation bien choisi, nous permettrait d’identifier la majorité des adresses de livraison.

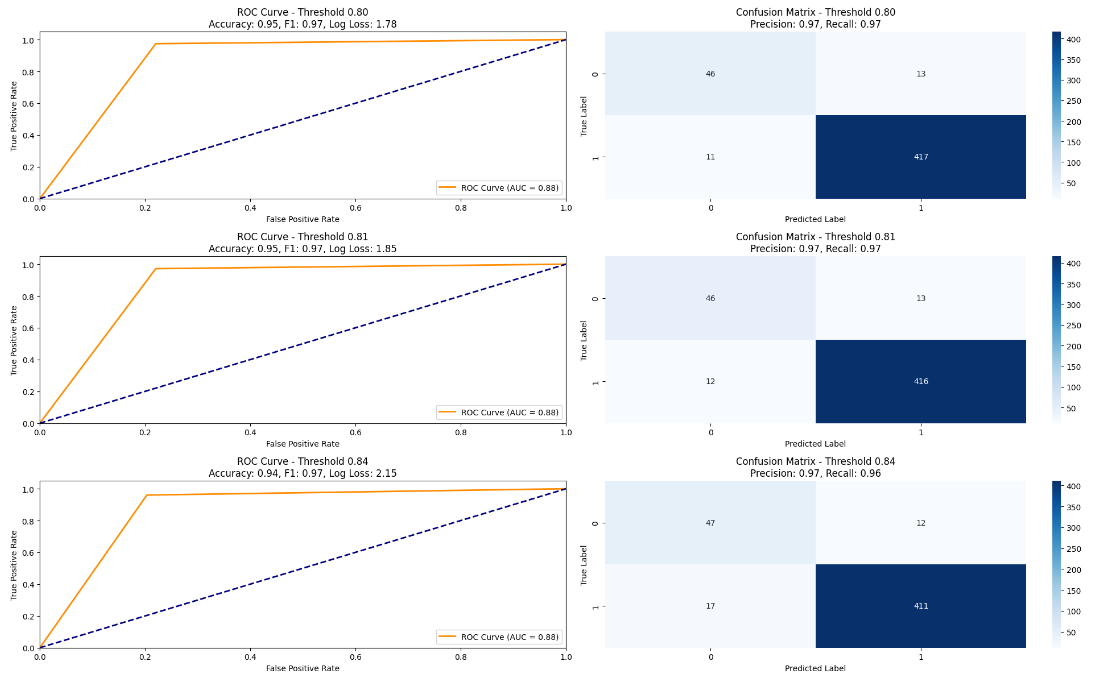
Nous avons également estimé le seuil optimal avec la statistique de *Youden’s J* qui nous a permis d’afficher les courbes ROC suivantes :



Les seuils estimés (0.81, 0.84, 0.78) sont nettement élevés que ceux qui sont estimés sur le F1 score. Nous validons tous ses seuils sur le 5ème échantillon de test et comparons les résultats :

Courbe Roc et matrice de confusions :

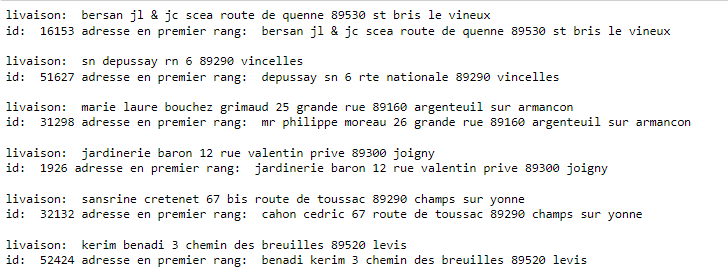




Les résultats sont assez bon contenu les seuils choisis. Nous remarquons tout de même une diminution dans les faux positives en haut à droite de chaque matrice de confusion quand le seuil augmente. L’augmentation du seuil augmente aussi la surface AUC.

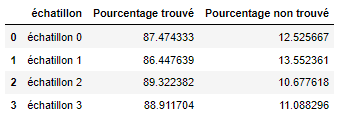
En fin de compte ce qui inacceptable quand nous choisissons une adresse sur la base d’un seuil et de ne pas se tromper par rapport au faux positifs. Nous avons analysé ces adresses apparues en faux positifs avec le seuil 0.70 et nous avons trouvé que bon nombre d’entre eux sont des doublons d’adresses et en fin de compte c’est des bonnes réponses.

Quelques exemples de faux positifs :



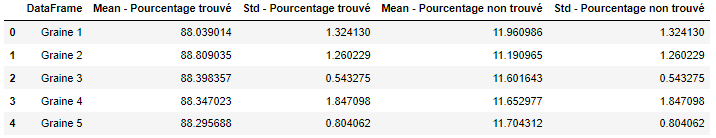
##### Tous les rangs :

Nous estimons qu’une adresse recherchée si elle existe ne se trouve pas forcément dans le premier rang en raison des divers écritures pour une adresse recherchée (les différents alias) et par conséquent, une adresse de livraison si elle est écrite différemment, son adresse de destinataire candidate peut se trouvée dans le 2 ou 3ème rang voir les rangs peu probables comme le 9 ou 10ème rang.

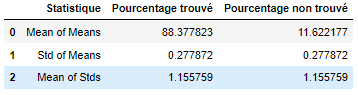


Nous remarquons une nette augmentation des pourcentages de d’identification des adresses recherchées. Nous effectuons les mêmes tests à travers les graines de génération d’échantillons :

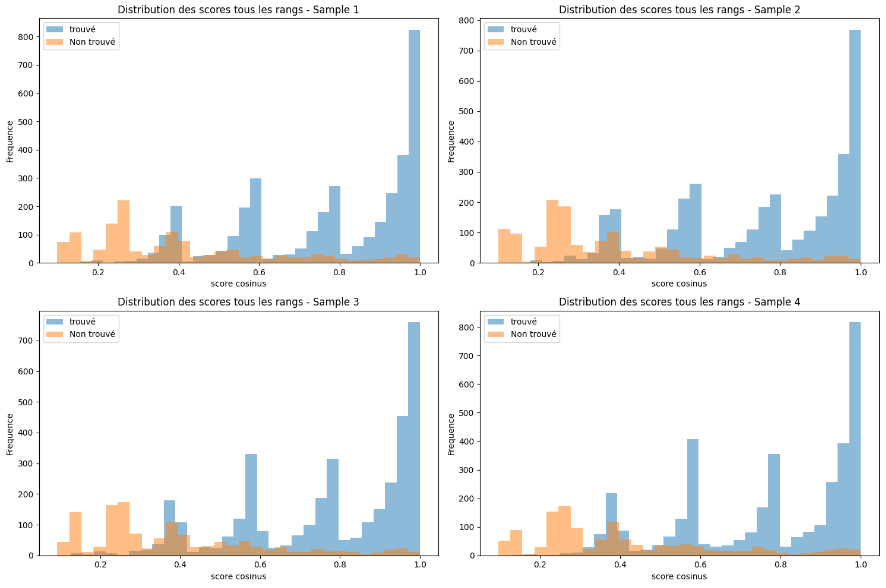
Statistiques par graine :



Statistiques globales :



Nous remarquons que les augmentations observées pour une graine restent consistants et au même niveau pour le reste des graines. Les analyses sont similaires quand nous prenons en compte tous les rangs.

Nous visualisons les scores de similarité également. 

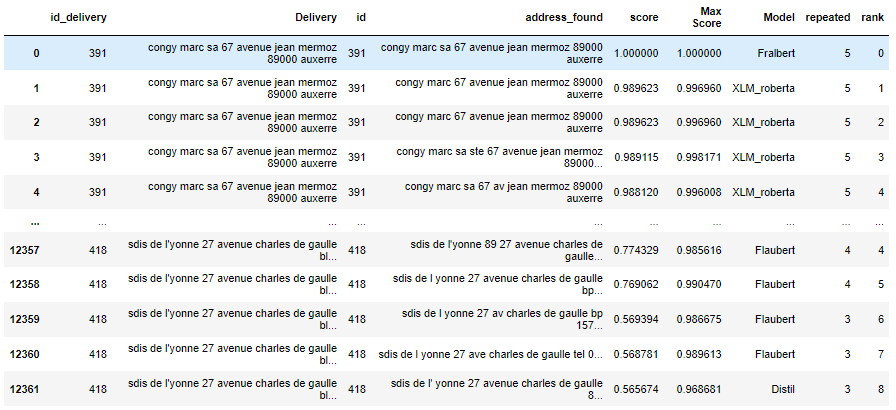
Nous remarquons ici que bon nombre d’adresses trouvées ont des scores très bas. Il faut noter ici que ces scores sont ramassés à travers tous les candidats de toutes les requêtes d’adresses de livraisons reçues, ce qui veut dire que plusieurs groupes de scores en bleu sont pour une même requête. C’est le principe des alias d’adresses. Nous pouvons donc avoir pour une requête, plusieurs bonnes réponses à partir du moment où le destinataire recherché a plusieurs alias, ce qui explique les scores assez bas car la récupération des adresses recherchées se fait avec le tri des scores.

Quelques exemples de certaines adresses trouvées avec un score vraiment bas :



Nous remarquons ici que le nombre de répétition influence énormément le score pondéré car comme nous pouvons l’observer les scores maximaux sont très élevés. Ces adresses sont des alias récupéré assez bas dans la liste car leurs nombre de répétition parmi les modèles d’encodage est assez bas (2 ou 3 répétitions).

Les autres alias associés ont été trouvés plus haut dans la liste avec des scores pondérés plus élevés car le nombre de répétition élevé n’a pas influencé leurs scores :



La critique que nous pouvons avoir ici est comment faire si l’adresse correcte recherchée est répétée 2 ou 3 fois mais que ces scores sont assez élevés et inversement les adresses fausses non recherchées se trouvent être répétées un maximum de fois avec des scores moins bons. Cela pourrait avantager ces adresses et les proposer au top de la liste.

Conclusion :

* Fixer un seuil approprié pourrait avoir un impact sur la qualité de l’identification des destinataires.
* Trouver d’autres critères de validation d’adresses correctes. Peut-être un score au-dessus d’un seuil plus le score maximum qui serait vraiment élevé pour décider du choix.