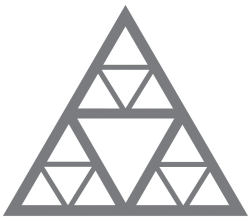

École des Ponts
ParisTech

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

ANALYSE DES DONNÉES DE CREUSEMENT DE TUNNELS

Par

Martin BIBONNE, Bertille BONIN, Cyril POSSARD, Mathis WAUQUIEZ

Ce projet a été effectué sous la supervision de

Tatiana RICHA, Jean-Michel PEREIRA et Lina GUAYACAN

Janvier – Mai 2023

Abstract

The objective of this work is to analyze the surface settlements observed following tunnel excavation. First, the data is organized in a database structure and then processed. Then, a statistical study is done in order to better understand the data. Finally, simple machine learning algorithms such as Random Forests are used for the prediction of settlements. The obtained results are compared with those measured in the field in order to validate the quality of the predictions.

Keywords

— Machine Learning
— Random Forest
— Tunnel boring machine
— Database
— Settlements
— Feature Engineering

1. Introduction

Dans le cadre du projet du Grand Paris Express, plusieurs lignes de métro sont actuellement en construction (Figure 1). Au total plus de 200 km de lignes seront tracées pour un budget total d'un peu moins de 40 milliards Euros.

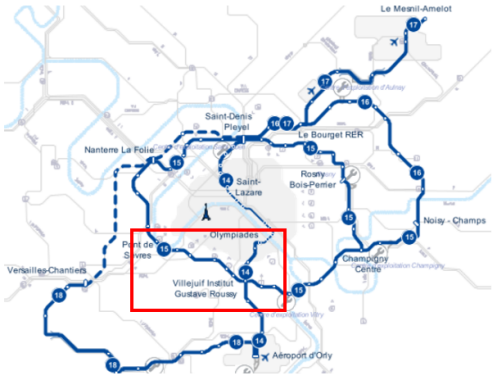

FIGURE 1 – Plan des lignes du Grand Paris Express (SOCIÉTÉGRANDPARIS, 2023)

Si certaines des lignes sont aériennes, la plupart sont souterraines. A cette date, six tunneliers sont en fonctionnement pour creuser les futures lignes souterraines 16, 17 et 18 (SOCIÉTÉGRANDPARIS, 2023). Or, après l'excavation de tunnel, apparaissent souvent en surface des fissures et fractures des terrains. Lorsque des tunnels sont creusés en-dessous de sites urbanisés, comme pour le projet du Grand Paris Express, il est primordial de limiter au maximum le tassement des terrains en surface afin d'éviter des dégâts sur les structures avoisinantes.

Dans le cadre de ce travail, nous avons accès aux données des lignes 14 Sud et 15 Sud-Ouest du Grand Paris Express.

1.1. Fonctionnement d'un tunnelier

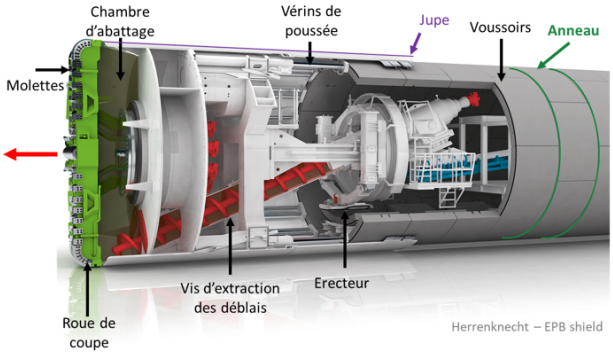

FIGURE 2 – Tunnelier à pression de terre (adapté de HERRENKNECHT, 2022)

Les lignes 14 et 15 du Grand Paris Express sont excavées en souterrain à l'aide de trois tunneliers à pression de terre (Figure 2). Ces derniers sont des machines de grande taille — 100 m de long pour un diamètre de 10 m — qui permettent d'excaver le sol. L'excavation se fait grâce à une roue de coupe possédant de nombreuses molettes qui creusent les terrains. Les tunneliers ont également pour fonction d'assurer la stabilité des sols au-dessus du front de l'excavation et la mise en place du soutènement (AFTES, 2019).

Un des principaux risques liés au creusement des tunnels en profondeur est la fracturation des sols en surface. Après l'excavation du tunnel, les couches de sol situées au-dessus vont progressivement se tasser. Si le tassement est trop important ou trop brutal, il peut y avoir des dommages en surface et des dégâts sur les structures.

1.2. Objectifs

Le long des 10 km des lignes 14 Sud et 15 Sud-Ouest étudiés, plus de 8500 capteurs ont été positionnés en surface et en profondeur. Les données fournies par les mesures de ces capteurs, mises en relation avec des informations diverses sur les terrains et le creusement du tunnel, permettent d'étudier les phénomènes de tassement en surface.

L'objectif est donc de prévoir les tassements en surface en fonction des données de creusement des tunnels et des données sur la nature des terrains. Cela permettra d'éviter de tels dommages lors de futures excavations de tunnels ou, à minima, d'effectuer les modifications nécessaires pour éviter les dégâts.

Cet article présente d'abord la mise en place des données dans une base de données, ensuite des études statistiques pour une meilleure compréhension des features et enfin une application d'algorithmes de *Machine Learning* pour la prévision des tassements.

2. Méthodologie

2.1. Création d'une base de données

Dans un objectif d'établir une méthode de traitement de données de creusement de tunnel qui puisse s'étendre à de longs tronçons, les données vont d'abord être organisées dans une base de données puis être traitées. En effet, les avantages d'une base de données sont nombreux : rapidité de lecture des données optimisée, utilisation partagée en ligne de la base de données, manipulation des données facilitée.

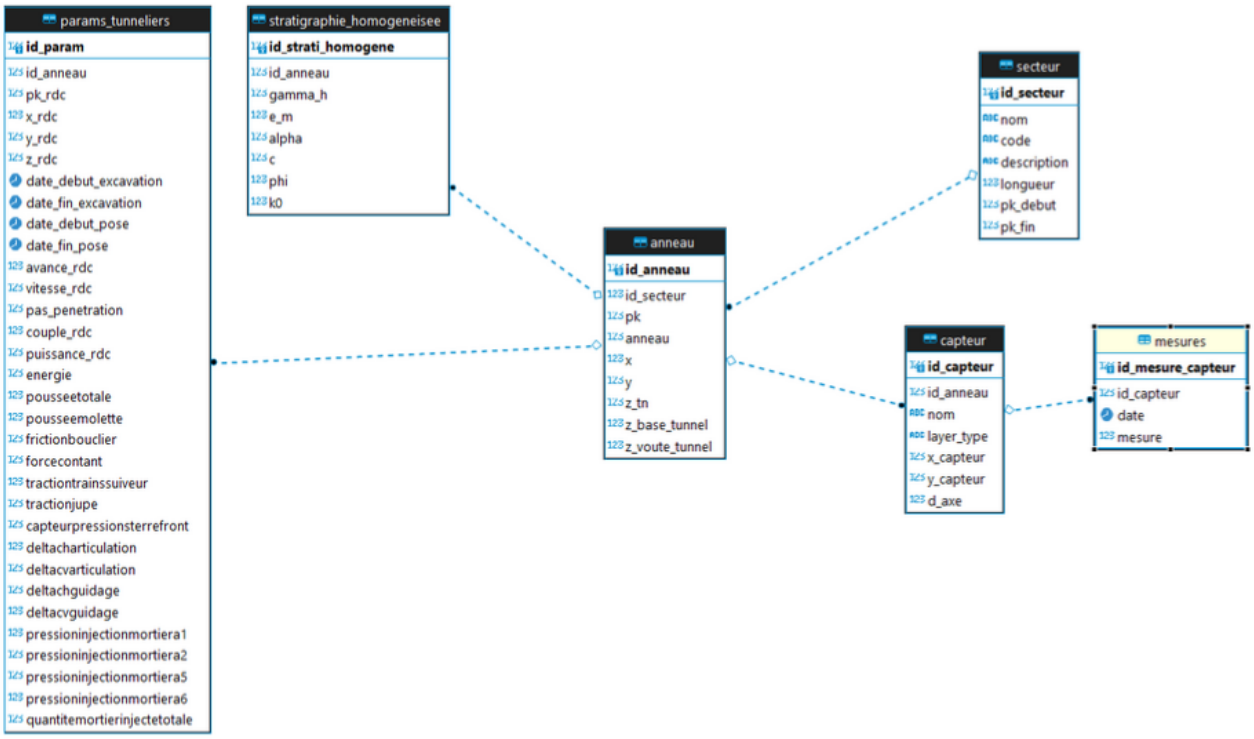

FIGURE 3 – Diagramme Entité-Relation de la base de données

La base de données choisie est organisée selon trois catégories : les tables des capteurs et leurs mesures, la table des paramètres de tunnelier et les tables des données des paramètres de sol. A cela s'ajoute la table anneau qui donne la position à chaque mètre du tracé. Le diagramme de la figure 3 montre l'organisation de la base de données.

D'abord, une base de donnée "vide" est créée pour définir les différentes tables et leurs propriétés. Ensuite, les données fournies dans des fichiers type csv sont injectées dans cette base. Pour cela, une connexion est d'abord établie entre la base de donnée et les instructions en ligne de commande sur Python (annexe A). Ensuite, les données de chaque table sont injectées dans la base de données (annexe C présente le cas de la table "secteur").

Pour que la méthode d'injection soit la plus générale possible, une fonction d'injection compatible avec n'importe quelle table est créée (annexe B).

Les données sont alors facilement accessibles, à travers des requêtes simples et rapides.

2.2. Filtrage des mesures des capteurs

Les mesures des capteurs collectées contiennent des incohérences qu'il faut traiter. Par exemple, certains capteurs ne renvoient pas des valeurs correctes. Une analyse statistique rapide des mesures des capteurs est réalisée afin d'identifier les erreurs.

nombre de valeurs	27 000 000
moyenne (en mm)	0,1096
écart type (en mm)	160,1
25% (en mm)	-1,100
50% (en mm)	-0,1125
75% (en mm)	0,5000
min (en mm)	$-5,738 \times 10^5$
max (en mm)	$1,325 \times 10^4$

TABLE 1 – Informations sur les mesures de tassement pour l'ensemble des capteurs

Les valeurs mesurées par les capteurs donnent un écart type $\sigma = 160$ mm et un troisième quartile valant 0.50 mm (Table 1). Ainsi certaines mesures sont très éloignées des tassements usuels. Ces valeurs sont considérées comme aberrantes et sont donc supprimées. Une observation des quantiles permet de mieux voir ces écarts caractéristiques de valeur de tassement. La Figure 4 représente les valeurs des mesures de tassement en fonction des différents quantiles.

Les valeurs sont à peu près cohérentes jusque des mesures de plus de 30 mm, ce qui correspond au dernier centile. Toutes les lignes dont la mesure est supérieure à 30 mm sont supprimées après le filtrage (code en annexe E). Ainsi, plus de 99% des données seront conservées.

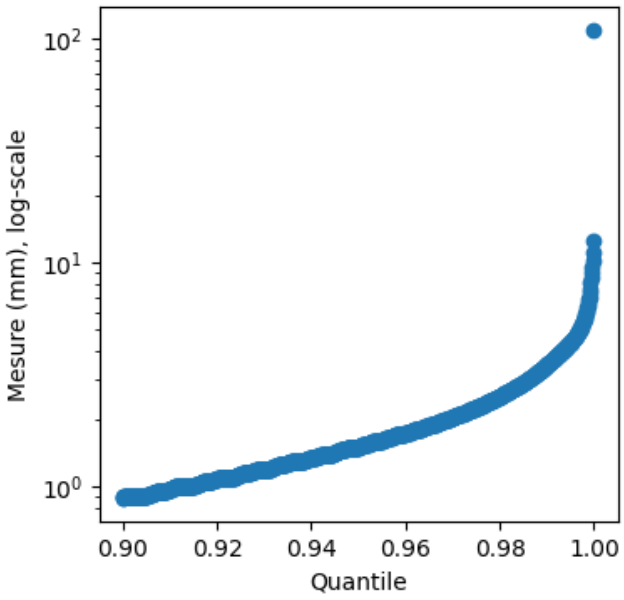

FIGURE 4 – Observation des valeurs des mesures des capteurs en fonction des quantiles

2.3. Machine Learning et Random Forest

Le *machine learning*, également connu sous le nom d'apprentissage automatique, est une branche de l'intelligence artificielle qui se concentre sur le développement de techniques permettant aux ordinateurs d'apprendre et de prendre des décisions sans être explicitement programmés. Plutôt que de suivre des instructions précises, les machines apprennent à partir de données et d'expériences passées pour améliorer leurs performances sur une tâche spécifique.

Les modèles de *machine learning* sont généralement catégorisés selon le mode d'apprentissage. Les plus connus sont les algorithmes d'apprentissage supervisés et non supervisés. Les modèles supervisés sont utilisés lorsque le jeu de donnée est constitué de paramètres étiquetés : par exemple, des images d'animaux étiquetées des noms de ces animaux, ou encore des mesures météorologiques provenant de ballons-sondes aux divers paramètres. Le premier exemple correspond alors à un problème de classification, là où le deuxième correspond à un problème de régression : il s'agit de déterminer une valeur continue en fonction de paramètres donnés.
Dans cette étude, a été utilisé un modèle d'apprentissage supervisé spécifique appelé *Random Forest*. Cet algorithme est un assemblage de plusieurs arbres de décision (ou *Decision Trees*, DT). Il faut donc commencer par expliquer les DT.

Les *Decision Trees* sont utilisés pour la régression et la classification. Ils peuvent être représentés visuellement sous la forme d'arbres. la racine puis se sépare en plusieurs branches selon des tests très basiques sur les variables, jusqu'à atteindre les feuilles où le résultat est donné. Un exemple de *Decision Tree* simple est donné sur la figure 5.

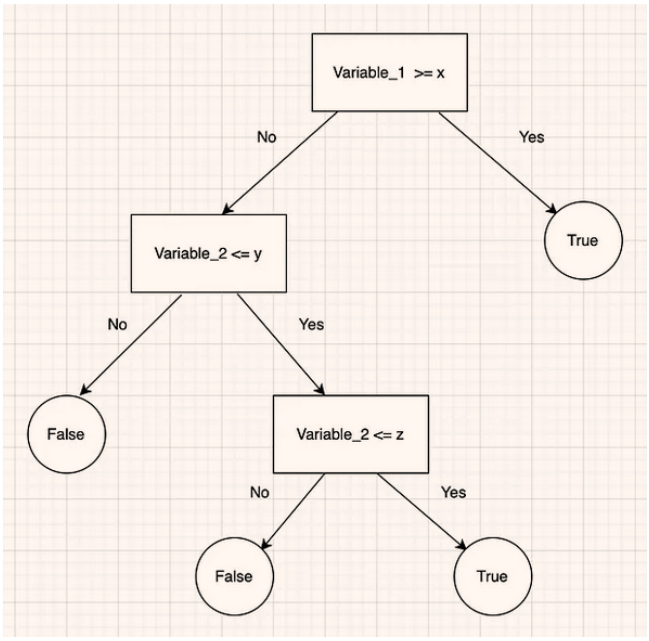

FIGURE 5 – Exemple de *Decision Tree* (BEHESHTI, 2022)

La méthode de *Bootstrapping* consiste à sélectionner aléatoirement plusieurs sous-ensemble du jeu de données et de réaliser un certain nombre d'itérations de *Random Forest* avec un certain nombre de variables (BEHESHTI, 2022). Les résultats sont ensuite moyennés pour avoir une réponse finale plus pertinente et plus fiable.

En résumé, l'algorithme *Random Forest* utilise des *decision trees* pour que le résultat moyenné soit une prédiction plus robuste et donc moins dépendant du jeu des données.

3. Résultats

Les courbes et statistiques obtenues dans les résultats ont été obtenues grâce à l'utilisation du langage Python.

3.1. Informations générales sur le tracé de la ligne

Le tunnel traverse des zones avec différentes couches de sol. La stratigraphie est représentée dans la figure 6 où chaque type de sol est indiqué par une couleur

différente. Cela reflète la diversité de la stratigraphie et la complexité de sa représentation. En effet, plus d'une dizaine de sols différents sont rencontrés par le tunnelier lors de l'excavation du tunnel.

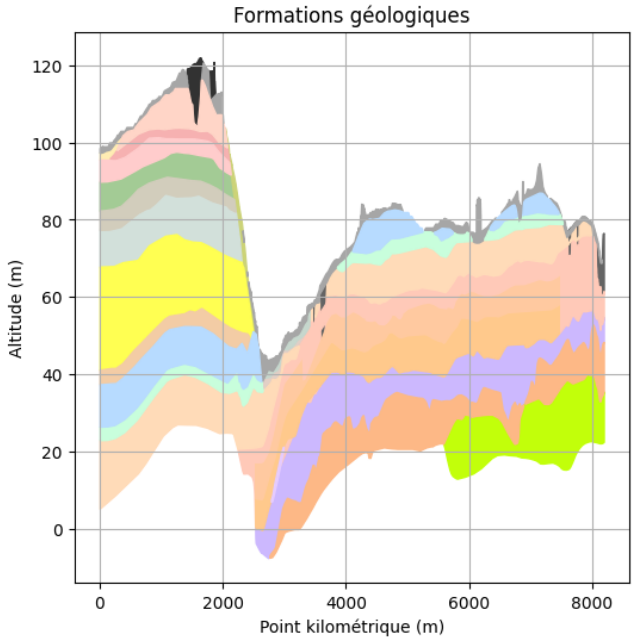

FIGURE 6 – Stratigraphie traversée par le tunnel en fonction du point kilométrique

Le tronçon de 10 km est découpé en dix secteurs selon la définition des caractéristiques mécaniques des couches de sol. Le découpage en secteurs du tracé de la ligne 15 Sud-Ouest est présenté dans la figure 7 (code en annexe D).

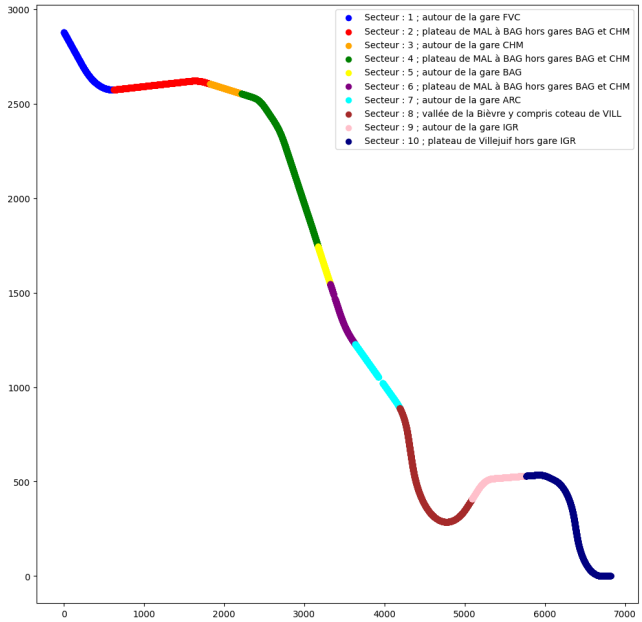

FIGURE 7 – Vue aérienne de la ligne découpée en secteurs

Ce graphique peut aussi être superposé à une carte du terrain, afin de vérifier que les coordonnées des points ont bien été prises en charge par le code python comme le montre la figure 8.

Il existe 14 types de capteurs différents qui couvrent tout le tronçon qui sont représentés ici en différentes couleurs. Ceux-ci permettront par la suite de déterminer les tassements le long de la ligne.

FIGURE 8 – Carte de la position des capteurs et des anneaux

Une vue de près sur la carte montre la présence de zone sans anneaux. Cette zone correspond à l'emplacement du puit creusé pour mettre en place le tunnelier (figure 9). Il est donc normal qu'il n'y ait ni capteurs, ni anneaux à cet endroit.

FIGURE 9 – Zoom sur la carte

La figure 10 donne l'emplacement du terrain naturel (en orange), du haut de la voûte (en vert) et du bas

de la voûte (en bleu) le long du trajet de la ligne de métro.

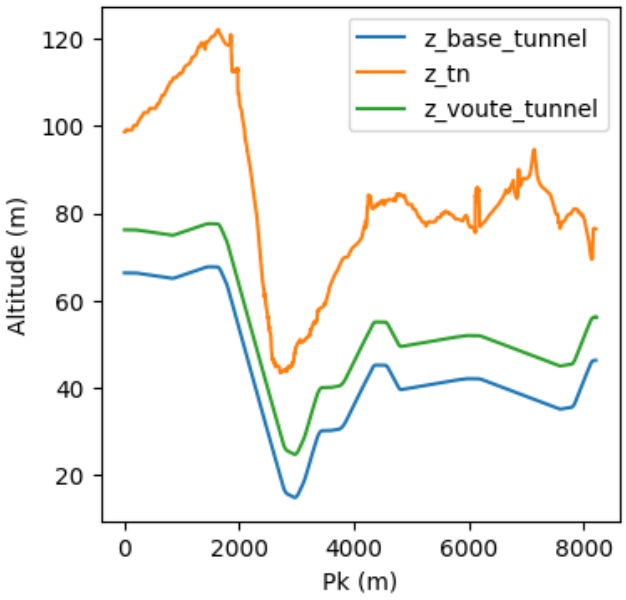

FIGURE 10 – Profondeur du tunnel et du terrain en fonction du point kilométrique

3.2. Détermination des tassements

Les capteurs placés le long des lignes enregistrent leur altitude au cours du temps. Il est donc possible d'observer le tassement sous forme d'une série temporelle. Dans cette étude, on s'intéresse au tassement maximal observé à long terme. Pour obtenir sa valeur, une régression est faite sur les mesures de chaque capteur. Pour cela sont déterminés les paramètres d'une fonction g définie par :

$$g(x) = s_m * F(\frac{x - x_0}{d_c}) + s_0$$

avec

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \, \mathrm{d}u$$

où

d_c la distance caractéristique de tassement

s_m le tassement maximal

s_0 constante de calibrage

x_0 paramètre permettant de translater horizontalement la courbe

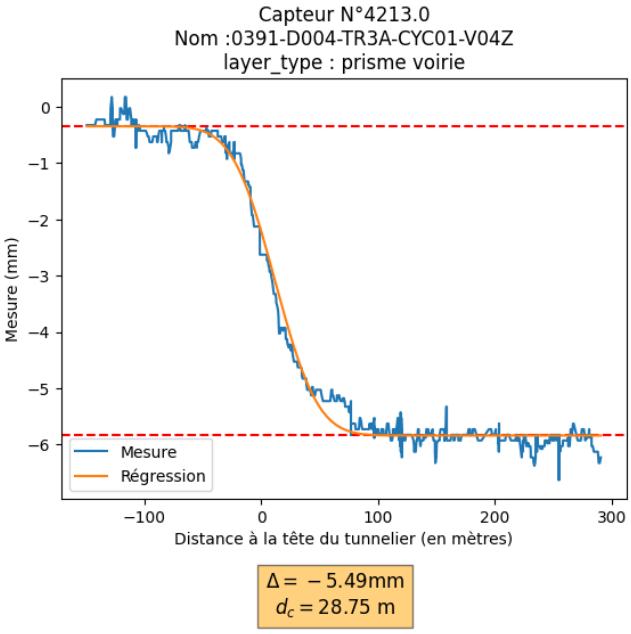

FIGURE 11 – Tassements en fonction de la date pour un des capteurs

La figure 11 représente les valeurs des tassements (en bleu) mesurés par le capteur n°3707 en fonction de la date. En orange est superposée la régression qui a été obtenue après calcul. Ensuite, le tassement maximal peut être calculé en soustrayant la limite en $-\infty$ à celle en $+\infty$ de la fonction de régression obtenue.

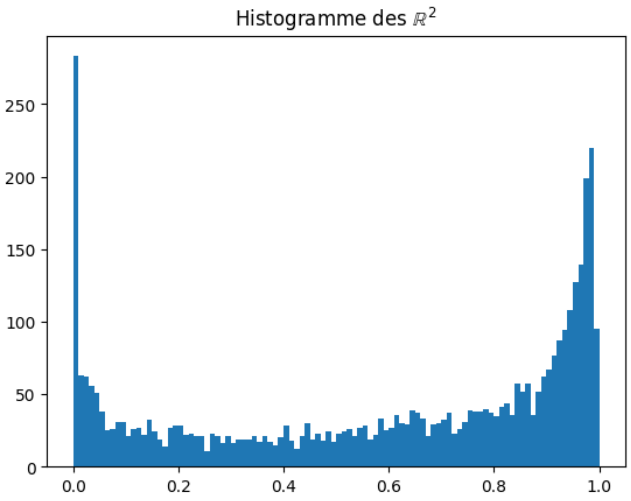

FIGURE 12 – *Feature* : distance au tunnelier

Pour savoir si la régression est pertinente, il est possible de calculer le coefficient de détermination R^2 de la régression – c'est-à-dire le pourcentage de variance expliquée par le modèle – qui compare les mesures réelles à la courbe calée. A savoir, un R^2 proche de 1 indique que le calage est bien réussi. La figure 12 représente un histogramme des poids des R^2 en fonction de leur valeur. De nombreux capteurs donnent

des R^2 compris entre 0.7 et 1, mais d'autres donnent de faibles valeurs de R^2. Ces derniers indiquent que les régressions ne sont pas optimales puisque seul un faible pourcentage de variance est expliqué par la régression. Pour la suite, seules les régressions avec un R^2 supérieur ou égal à 0.7 sont conservées.

3.3. Résultats du modèle de Random Forest

D'abord, le modèle est entraîné sur un jeu d'entraînement. La taille de cet ensemble doit être choisie de façon à ce qu'il y ait suffisamment de données pour que le modèle puisse généraliser les relations qu'il a pu apprendre. De plus, il faut garder suffisamment de données pour tester de manière statistiquement rigoureuse la performance du modèle. Le jeu de test sera utilisé pour valider le modèle obtenu après l'entraînement.

Dans cette étude, nous avons pris 80% des données pour l'entraînement, soit 828 observations, et 20% pour le test, soit 207 observations.

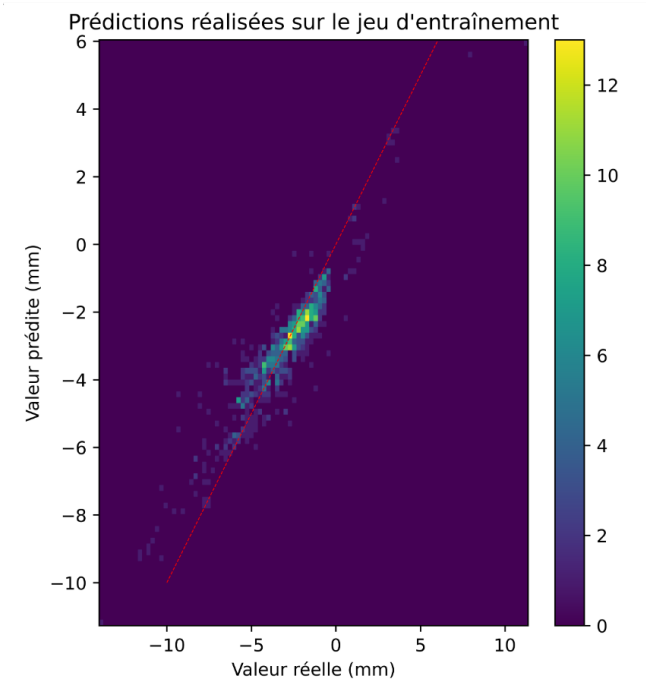

FIGURE 13

La figure 13 présente les résultats de l'entraînement : les valeurs prédites sont tracées en fonction des valeurs réelles. La droite en rouge représente la droite d'équation $y = x$, c'est-à-dire les points dont les prédictions sont égales aux valeurs réelles. Ainsi si les valeurs des prédictions sont situées sur cette droite, le modèle fonctionne correctement et prédit les bonnes valeurs. Ici les valeurs prédites correspondent aux valeurs mesurées par les capteurs et le modèle fonctionne pour le set d'entraînement. L'échelle de couleur représente le nombre de valeurs obtenues pour un même tassement.

Ensuite le modèle des *Random Forests* maintenant entraîné est utilisé pour réaliser des prédictions sur le set de test. De la même manière, la figure 14 représente les prédictions réalisées sur le jeu de test.

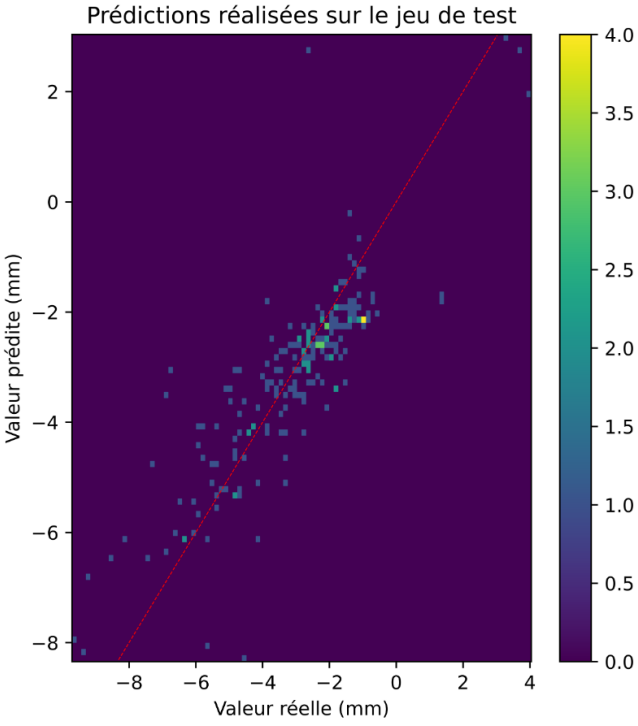

FIGURE 14

Les résultats sur le jeu de test sont en général moins bons que ceux sur le jeu d'entraînement (valeurs prédites plus éloignées des valeurs réelles). Cela fait sens car la performance du modèle ne peut pas être aussi bonne sur le jeu de test que d'entraînement, à moins d'avoir un grand nombre de données devant le nombre de paramètres appris par le modèle.

Pour évaluer la performance du modèle, sont calculés son MAE (Mean Absolute Error) et son RMSE (Root Mean Squared Error) (KARUNASINGHA, 2022) sur les différents jeux de données. Leurs valeurs sont consignées dans la table 2. Plus les valeurs de RMSE et de MAE sont basses, plus les résultats sont fiables.

Set	Entraînement	Test
RMSE	0.92	1.1
MAE	0.66	0.75

TABLE 2 – Évaluation de la performance du modèle

Les valeurs des RMSE sont significativement plus

grandes que celles des MAE, ce qui signifie que la distribution des erreurs est à haute variance. Les métriques d'erreur sont meilleures (plus faibles) sur le jeu d'entraînement que de test, ce qui est un résultat normal. Néanmoins, il serait sans doute possible d'améliorer la performance du modèle en utilisant un set plus important pour les données d'entraînement afin d'éviter les problèmes d'*overfitting*.

La MAE sur le set de test vaut $0.75\ mm$, ce qui signifie que la valeur des tassements prédite est déterminée pour chaque capteur avec une erreur de 0.75mm en moyenne.

Afin d'améliorer la performance globale du modèle et d'éviter au maximum l'*overfitting*, il faut réduire le nombre de paramètres du modèle de *Random Forest*. Pour l'instant le modèle tournait en utilisant 28 *features*. Grâce à une étude ce nombre de *features* a été réduit à 7, ce qui a permit d'améliorer la généralisation du modèle.

4. Analyses

4.1. Choix des caractéristiques pertinentes pour les prédictions des tassements

La figure 12 indique les valeurs des R^2 pour des régressions faites en choisissant comme *feature* principal la distance au tunnelier. Pourtant ce choix de *feature* pour la prédiction peut paraître arbitraire. Un histogramme des R^2 obtenus en faisant les régressions en fonction du temps est en figure 15.

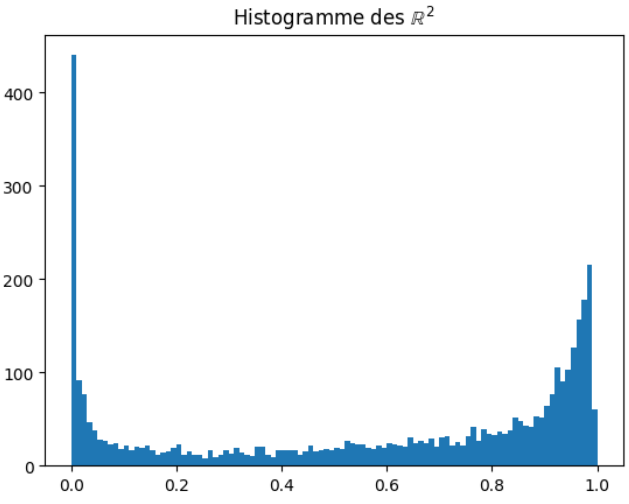

FIGURE 15 – *Feature* : temps

La comparaison des figures 12 et 15 permet de voir que choisir la distance au tunnelier comme *feature* au lieu du temps est beaucoup plus pertinent que de choisir le temps. En effet, pour plus de capteurs la valeur de R^2 proche obtenue est proche de 1 lorsque la *feature* principale de la régression est la distance au tunnelier.

4.2. Statistiques sur les tassements

Les tassements obtenus donnent des informations sur l'impact du passage du tunnelier sur les terrains. En effet, la figure 16 est une figure donne la probabilité conditionnelle des tassements obtenus pour chaque distance à l'axe.

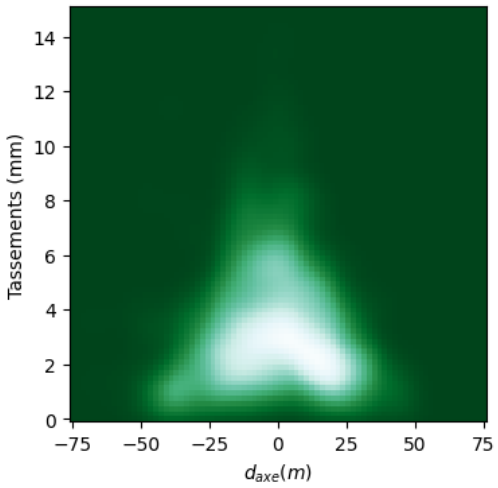

FIGURE 16 – Histogramme des tassement en fonction de la distance à l'axe

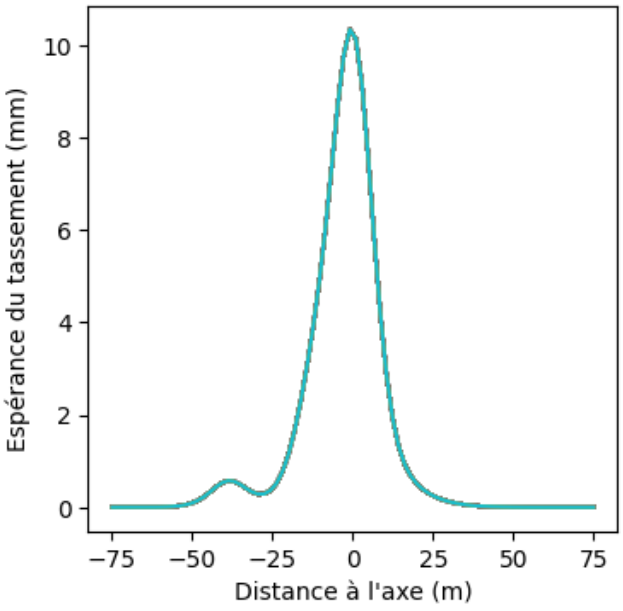

FIGURE 17 – Espérance conditionnelle des tassements sachant la distance à l'axe

Le graphique de la figure 16 montre que les tassements sont plus en général plus importants lorsque la distance à l'axe est faible. Ainsi, les terrains proches du tunnel creusés se tassent plus que ceux situés plus loin. Ce résultat est logique et pertinent. Il est par ailleurs intéressant d'afficher l'espérance du tassement en fonction de la distance à l'axe, ce qui est illustré par la figure 17.

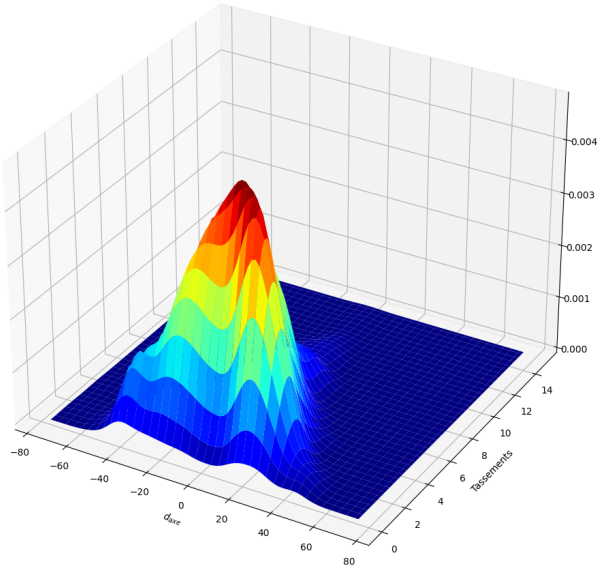

FIGURE 18 – Histogramme des tassement en fonction de la distance à l'axe

Une observation en 3D peut aussi être faite. Le graphique figure 18 représente une densité en fonction de la distance à l'axe et du tassement. Ainsi plus la valeur de l'ordonnée est élevée et plus il y a de valeurs indiquant un certain tassement pour une certaine distance à l'axe. De même une forme de voûte est observée proche de l'axe du tunnel indiquant que les tassements sont plus importants proche de l'axe comme l'indique la figure 18.

4.3. Choix des *features* pour les *Random Rorests*

Ainsi, l'affichage de ces figures pour chaque paramètre a permis, dans un premier temps, de voir si le paramètre était fortement déterminant ou non dans la régression. Cela a permis, par la suite de réduire le nombre de features du modèle.

Il est donc d'une importance cruciale de prendre en compte la corrélation des caractéristiques. Pour ce faire, il est intéressant de regarder la table de corrélation des caractéristiques, qui affiche, pour chaque combinaison, leur corrélation, comme le montre les figures 19 et 20.

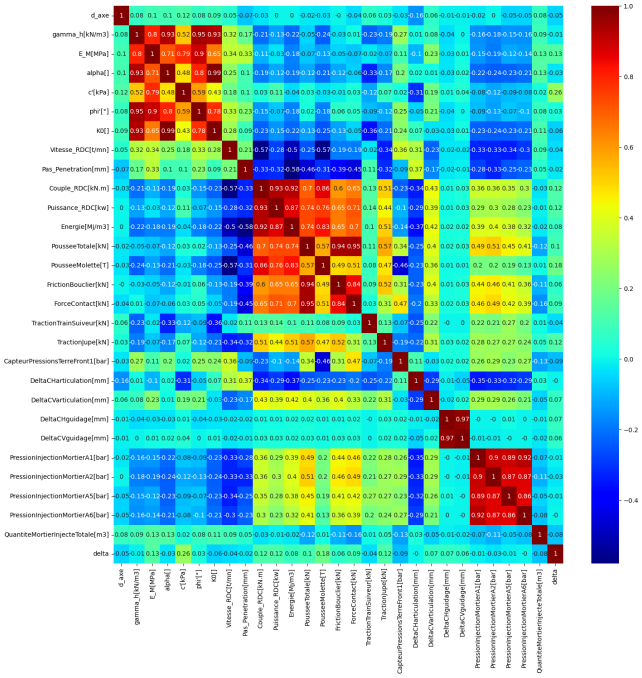

FIGURE 19 – Carte des corrélations entre paramètres

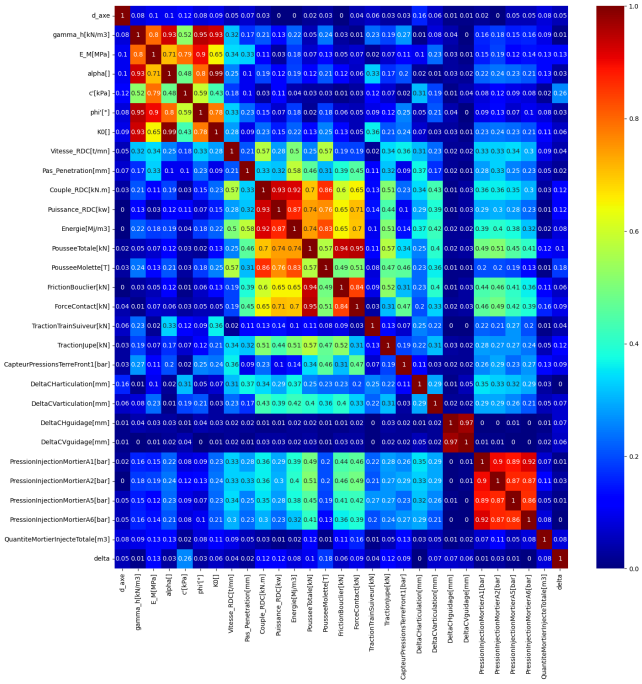

FIGURE 20 – Carte des valeurs absolues des corrélations entre paramètres

Certaines caractéristiques sont fortement corrélées (en couleur rouge–orange sur la figure 19) et il n'est pas pertinent de toutes les utiliser pour la régression. Par exemple, les pressions d'injection de mortier ou encore certaines données mécaniques du creusement par le tunnelier.

Un coefficient de corrélation linéaire proche de 1 indique que lorsqu'une caractéristique est élevée, l'autre a tendance à l'être aussi. Au contraire, un coefficient proche de -1 indique que lorsqu'une caractéristique est élevée, l'autre est faible, et enfin une corrélation de 0 indique une indépendance linéaire des variables, qui peuvent néanmoins être corrélées non-linéairement.

La valeur absolue des corrélations sur la figure 20 est affichée alors un coefficient qui correspond au pourcentage de variance expliquée par une relation linéaire entre les deux variables dont le coefficient est calculé.

Cette analyse de corrélations permet de diminuer encore le nombre de *features* étudiées, tombant alors 13.

Ce nombre de *features* étant faible, il est possible d'explorer les $2^{13} = 8192$ combinaisons de *features* afin de déterminer quelle combinaison est la plus pertinente, ainsi que, pour chaque nombre de *features*, lesquelles mènent aux meilleurs résultats, afin de déterminer leur ordre d'importance dans la détermination des tassements.

Le nombre optimal de caractéristiques vaut alors 7, et les caractéristiques optimales trouvées sont les suivantes, classées par ordre d'importance :

— la cohésion du sol
— la distance à l'axe
— le module de Young
— la poussée totale du tunnelier
— l'énergie du creusement
— la pression d'injection du mortier
— le pas de pénétration de la roue de coupe

Après entraînement d'un modèle de *Random Forest* en utilisant les caractéristiques optimales, la MAE obtenue est bien plus faible sur le jeu de validation, qui vaut alors $0.57\ mm$, pour un RMSE de $0.83\ mm$, ce qui correspond à une nette amélioration des résultats.

5. Conclusion

Lors de cette étude ont été utilisées des données des lignes 14 et 15 concernant les sols, le tunnelier et les capteurs. Une base de données a été créée et les données ont été nettoyées.

Le tassement maximal pour chaque capteur a été calculé grâce à une régression dont les paramètres ont été déterminés. Une étude faite sur les valeurs des R^2 des régressions a permis de valider la qualité du calage.

Le modèle de *Machine Learning* utilisé – les *Decision Tree* – a permis de prédire les tassements. Le fonctionnement et la pertinence du modèle a ensuite été validé en comparant valeurs mesurées et valeurs prédites. L'étude s'est prolongée par une amélioration des modèles de *Random forest* grâce à du *Feature Engineering*.

Références

AFTES (2019). *État de l'art concernant les évolutions des tunneliers et de leurs capacités de 2000 à 2019 (Groupe de Travail n°4 - GT4R6F1)*. Rapp. tech. (cf. p. 2).

BEHESHTI, N. (2022). *Random Forest Classification*. URL : https://towardsdatascience.com/random-forest-classification-678e551462f5 (cf. p. 4).

HERRENKNECHT (2022). *EPB Shield*. URL : https://www.herrenknecht.com/en/products/productdetail/epb-shield/ (cf. p. 2).

KARUNASINGHA, D. S. K. (2022). "Root mean square error or mean absolute error ? Use their ratio as well". In : *Information Sciences* 585, p. 609-629. DOI : https://doi.org/10.1016/j.ins.2021.11.036. URL : https://www.sciencedirect.com/science/article/pii/S0020025521011567 (cf. p. 7).

SOCIÉTÉGRANDPARIS (2023). *Carte du Grand Paris*. URL : https://www.societedugrandparis.fr/travaux/tunneliers (cf. p. 2).

6. Annexes

A.

```
conn = psycopg2.connect(database="postgres", user="postgres", password="1234")
curs = conn.cursor()
```

B.

```
def inject_into_table(table, dataframe, names, function=None, nb_per_cmd = 5000, drop = False):
"""Injectes les données dans une table"""
    if drop:
        curs.execute('DELETE FROM ' + table)
    if function is None:
        function = [str for i in range(len(names.split(',')))]

    a = 0
    n = len(dataframe)
    gen = dataframe.iterrows()

    while (a < n):
        if(a % 10000 == 0):
            print('{} / {} : {:.2f} % traités'.format(a, n, 100*a/n))
        string = ''
        i = 0
        while(a < n and i < nb_per_cmd):
            i += 1
            a += 1
            row = next(gen)[1]
            try:
                string += ' (' + ", ".join([function[j](row[j]) for j in range(len(row))]) + ') ,'
            except:
                print(row)
                raise
        string = string[:-2]

        cmd = "INSERT INTO " + table + " (" + names + ") VALUES" + string
        curs.execute(cmd)
        conn.commit()
```

LA fonction d'injection ci-dessus prend en argument le nom de la table dans laquelle on va injecter les données, le tableau pandas, les noms des colonnes correspondantes dans la table, et potentiellement les fonctions à utiliser pour traiter les données (pour bien convertir les données dans la requête d'injection), ainsi que le nombre de lignes injectées par commande, et un argument drop que l'on peut passer pour oublier les données préalablement injectées.

C.

```
table = "SECTEUR"
dataframe = data['secteurs']
names = "id_secteur, nom, code, description, longueur, pk_debut, pk_fin"
```

```
function = [stringify, string2string, string2string, string2string, stringify, stringify,
    stringifyNA]
inject_into_table(table, dataframe, names, function)
```

D.

```
    data['anneaux']['x'] -= np.min(data['anneaux']['x'])
    data['anneaux']['y'] -= np.min(data['anneaux']['y'])
    data['anneaux']['pk'] -= np.min(data['anneaux']['pk'])

    fig = plt.figure(figsize=(12, 12))

    ax = fig.add_subplot()

    colors = np.array(['blue', 'red', 'orange', 'green', 'yellow', 'purple', 'cyan', '
    brown', 'pink', 'navy', 'grey', 'black', 'olive', 'limegreen', 'coral'])
    for i, (sect, grp) in enumerate(data['anneaux'].groupby('id_secteur')):
        ax.scatter(grp['x'], grp['y'], c=colors[i], label="Secteur : " + str(sect) + ' ;
    ' + np.array(data['secteurs']['nom'][data['secteurs']['id_secteur'] == sect])[0])
    _=plt.legend()

```

E.

```
    mesures.drop(mesures[mesures['mesure'] > 30].index)

```