



Temps de Réponse de la Brigade des Pompiers de Londres

Rendu final : rapport d'exploration, de data visualisation,
de pre-processing des données et de modélisation

Date du rapport : 25 Jul 2024

Rédigé par :

Suzanne LIN
Boubs NDIAYE
Keyvan FRANCESCHI

Mentor de projet : Eliott DOUIEB



Table des matières

I. Introduction au projet	3
A. Contexte	3
B. Objectifs	4
II. Compréhension et manipulation des données	5
A. Cadre	5
B. Pertinence	6
C. Visualisations et Statistiques	7
D. Pre-processing et feature engineering	22
III. Modélisation	25
A. Classification du problème	25
❖ Description de la première approche : régression	25
❖ Description de la seconde approche : classification	26
B. Choix du modèle et optimisation	27
❖ Modèles de régression	27
❖ Modèles de classification	32
C. Interprétation des résultats	38
IV. Conclusion	41
A. Difficultés rencontrées	41
B. Bilan	42
C. Suite(s) du projet	43
D. Bibliographie	43
E. Annexe	44



I. Introduction au projet

A. Contexte

L'objectif de ce projet est d'analyser et/ou d'estimer les temps de réponse et de mobilisation de la Brigade des Pompiers de Londres. La brigade des pompiers de Londres est le service d'incendie et de sauvetage le plus actif du Royaume-Uni et l'une des plus grandes organisations de lutte contre l'incendie et de sauvetage au monde.

- Le premier jeu de données fourni contient les détails de chaque incident traité depuis janvier 2009. Des informations sont fournies sur la date et le lieu de l'incident ainsi que sur le type d'incident traité.
- Le second jeu de données contient les détails de chaque camion de pompiers envoyé sur les lieux d'un incident depuis janvier 2009. Des informations sont fournies sur l'appareil mobilisé, son lieu de déploiement et les heures d'arrivée sur les lieux de l'incident.

Ce projet peut s'inscrire dans plusieurs dynamiques :

- Analyse et dynamique des différentes localités de Londres vis-à-vis de l'efficacité de la couverture par la Fire Brigade of London (FBG).
- Optimisation des temps de réponse et d'intervention en vue d'une meilleure réponse.
- Identification et compréhension des facteurs pouvant induire un temps de réponse élevé.
- Optimisation des coûts pour une meilleure performance des services de pompiers et une meilleure anticipation budgétaire associé au secours de personnes.



B. Objectifs

- Objectif principal :
 - Prédiction du temps de mobilisation et de trajet suite à appel du numéro d'urgence (999) et mobilisation d'une brigade.
- Objectifs secondaires :
 - Prédiction du temps d'intervention une fois sur place,
 - Identification des principales données d'entrée influent sur les durées d'intervention dans une optique d'amélioration du service : temps de réponse.
 - La réduction du temps d'intervention (souvent liée au temps de réponse) et par conséquent du coût d'une intervention :
 - Coût humain (vies sauvées)
 - Coût matériel (dégâts de l'incendie, ...)
 - Coût financier (assurances, coût de l'intervention)
- Niveau d'expertise des membres du groupe autour de la problématique adressée :
 - Suzanne : pas d'expertise préalable particulière sur la problématique.
 - Boubs : pas d'expertise préalable particulière sur la problématique.
 - Keyvan : pas d'expertise préalable particulière sur la problématique.



II. Compréhension et manipulation des données

A. Cadre

Description des jeux de données :

Jeux de données “Incidents” 2 data sets couvrant la période globale du 1er janvier 2009 à 2024	1,701,647 entrées 39 variables
Jeux de données “Mobilisation” 3 data sets couvrant la période globale du 1er janvier 2009 à 2024	2,373,348 entrées 22 variables

Ces données* sont-elles disponibles librement aux liens suivant sur le site du gouvernement du Royaume-Uni :

- [London Fire Brigade Incident Records - London Datastore](#)
- <https://data.london.gov.uk/dataset/london-fire-brigade-mobilisation-record>

*Les datasets ont été téléchargés le 31/05/2024.

Introduction de datasets complémentaires :

- Datasets avec des données géographiques permettant des regroupement de données par code postal ou code postal de district pour la datavisualisation.



B. Pertinence

Afin de se lancer dans l'exploration des données, nous avons dans un premier temps identifié les variables que nous avons trouvé les plus pertinentes dans leur signification :

- **Date et heure de l'incident** (datetime de l'appel/mobilisation)
 - Comprend l'année, le mois, le jour et l'heure de la mobilisation.
- **Le nombre de véhicules et leur caserne de déploiement :**
 - Station de proximité ou autre station
 - Localisation de la station
 - Nombres de véhicules déployés depuis cette même station
- **La date et l'heure d'arrivée sur les lieux** : de cette variable nous pouvons calculer le temps nécessaire pour l'arrivée du véhicule.
- **Le lieu de l'incident** : il est essentiel que d'une manière où d'une autre le lieu de l'incident entre en compte dans les variables de notre modèle.
- **La typologie du lieu d'incident** : une maison, un immeuble, un bateau, un avion... Les lieux peuvent nécessiter du matériel ou des expertise différentes susceptibles d'influer sur le temps de préparation et d'arrivée des secours.
- **La typologie d'incident** : intervention de secours classique, incendie, fausse alerte...

Nous souhaitons étudier la valeur du temps de réponse, soit la différence entre la valeur **datetime** de l'appel et la valeur **datetime** d'arrivée sur les lieux.

- Limites :
 - Un certain nombre d'entrées (environ 5%) n'ont pu trouver de cohérence entre les deux datasets (incidents et mobilisations) et ont donc été supprimés dans la fusion des datasets
 - Les variables "date" et "time" provenant du dataset mobilisations ne sont pas cohérentes avec le reste des valeurs "date" et "time" du dataset "incidents" et n'ont donc pas pu être exploitées.
 - Certaines variables montrent un nombre élevé de NaN difficile à traiter/ remplacer et deviennent donc inutilisables.



C. Visualisations et Statistiques

Dans le but de réaliser des visualisations de qualité, nous avons analysé les datasets, créé 3 nouvelles variables et fait un premier tri parmis les variables :

Nouvelles variables :

- 'ResponseDuration' représente la durée en secondes entre la mobilisation et l'arrivée du véhicule sur les lieux de l'incident.
- 'DateAndTimeCalled' rassemble les informations de 4 variables de temps originaires du dataset mobilisation. Cette nouvelle variable permet la comparaison avec les autres variables DateTime proposées par le dataset Incidents.
- 'IncidentType' - nouvelle variable détaillant le contenu de False Alarm, Fire & Special Service en combinant les variables 'IncidentGroup', 'StopCodeDescription' et 'SpecialServiceType'.

Nettoyage pré-data visualisation :

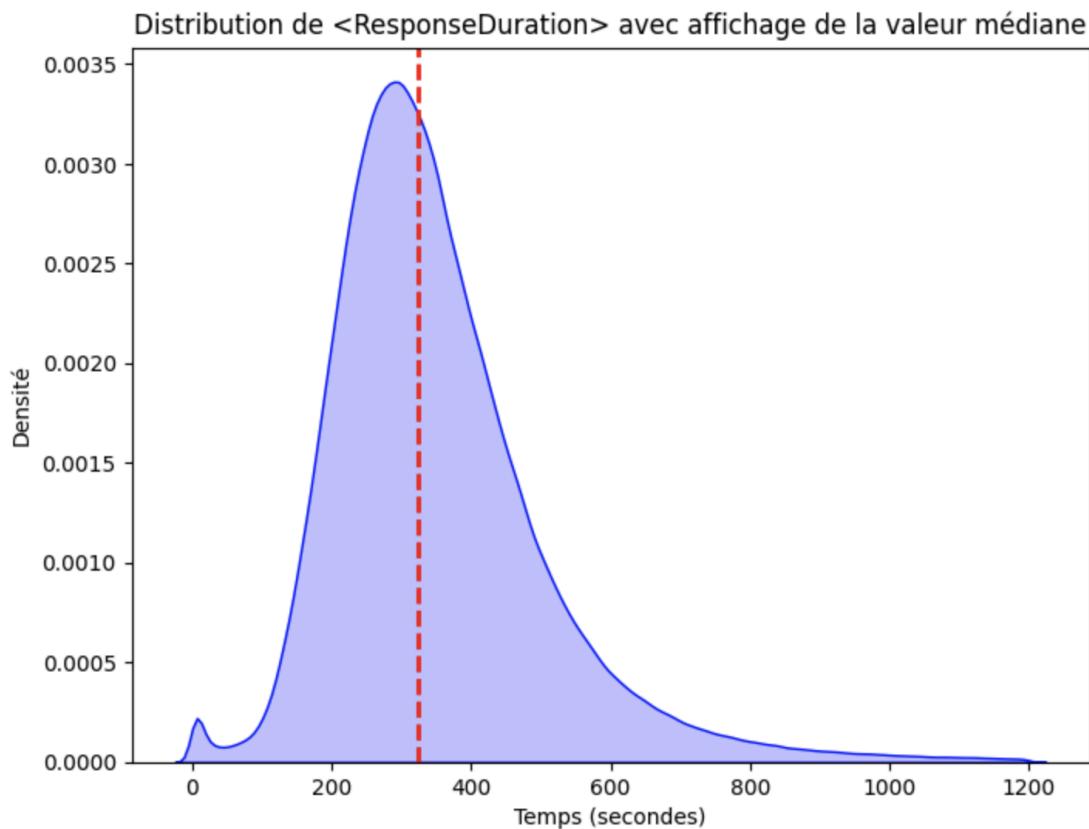
- Les **entrées sans correspondance** entre les deux datasets par la clé de liaison 'IncidentNumber' ont été supprimées : il était en effet impossible de faire la liaison entre ces données et les conserver aurait induit leur suppression par la suite faute de pouvoir traiter les NaNs de variables essentielles.
- Suppression **des variables administratives** :
 - 'IncidentNumber'
 - 'ResourceMobilisationId'
- Suppression d'une variable à **valeur unique** :
 - 'FRS'
- Nous avons supprimé des **variables à information redondante** :
 - 'DeployedFromStation_Code' - code pour identifier la station de déploiement,
 - 'PlusCode_Code', - code pour identifier le type d'incident,
 - 'DelayCodeId' - code pour identifier une cause de retard,
 - 'ProperCase' - mise en forme textuelle d'une variable identique.
 - 'IncGeo_WardCode' - code pour identifier le quartier



- Suppression des **variables “colinéaires”** pouvant être recalculés à partir d'autres variables :
 - 'PerformanceReporting' - calculable à partir de PumpOrder
 - 'DateOfCall' - calculable à partir de 'DateTimeCall',
 - , 'TimeOfCall' - calculable à partir de 'DateTimeCall',
 - 'CalYear' - calculable à partir de 'DateTimeCall',
 - 'HourOfCall' - calculable à partir de 'DateTimeCall',
 - 'TurnoutTimeSeconds' - calculable à partir des valeurs DateTime du dataset incident
 - 'TravelTimeSeconds', - calculable à partir des valeurs DateTime du dataset incident
- Suppression des **variables avec trop de NaN** :
 - 'DelayCode_Description' - 75% de NaN - malgré le fort intérêt que pourrait porter cette valeur, il est difficile d'interpréter avec certitude la cause des NaN et donc de les renseigner,
 - 'DateAndTimeReturned' - 59% de NaNs - cette variable n'a en plus à priori que peu d'intérêt car située temporellement après les événements qui nous intéressent,
 - Informations géographiques de l'incident basée sur le 'Postcode_full'
 - 54% de NaNs - elle peut être retrouvée par d'autres variables.
 - 'Postcode_full',
 - 'Easting_m',
 - 'Northing_m',
 - 'Latitude',
 - 'Longitude'



Distribution de la “target” représentant la durée entre la mobilisation des pompiers et leur arrivée sur les lieux :



Observations :

On constate une distribution des valeurs asymétriques autour d'une durée d'environ 300 secondes (5 minutes).

Hypothèses :

On constate également des **valeurs potentiellement aberrantes** sur la valeur de 0 OU très proches de 0, dont les causes peuvent être diverses, par exemple :

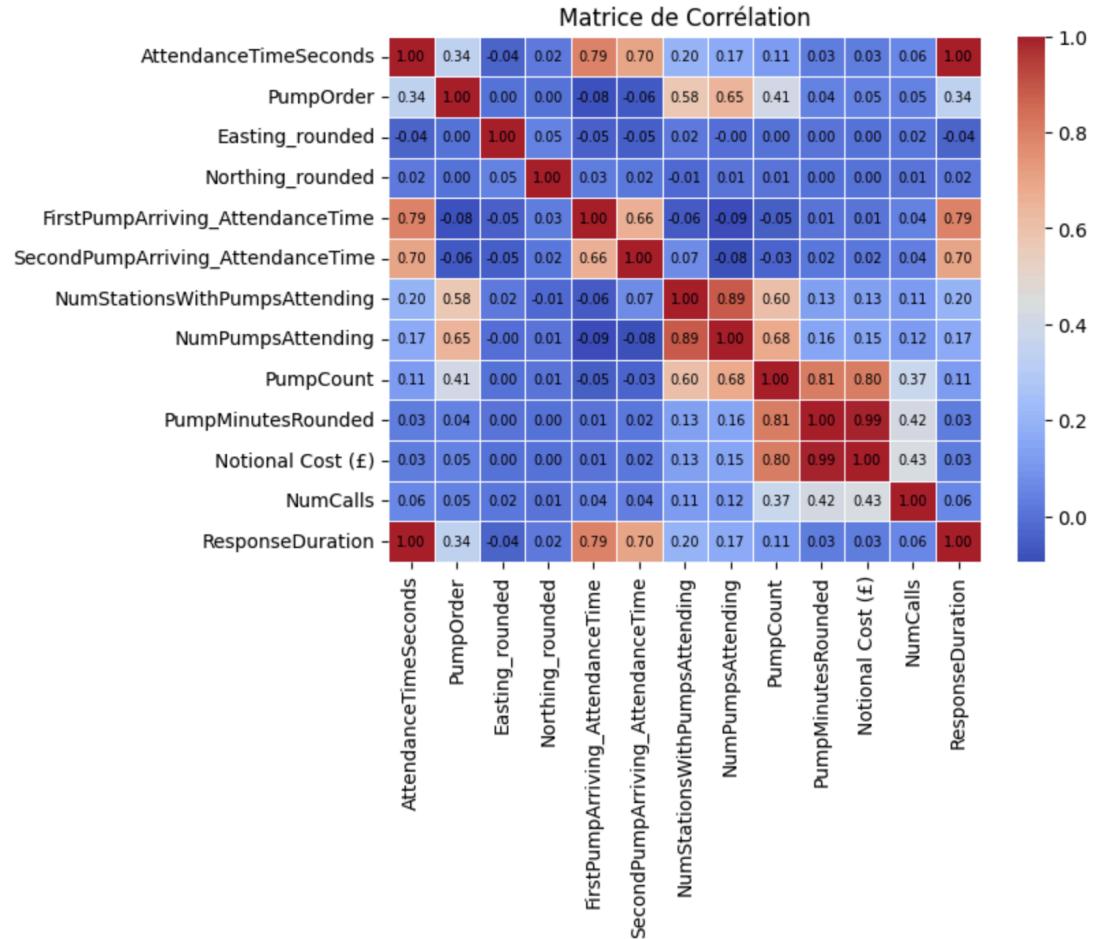
- Mauvais renseignement de la valeur.
- Faux départ et annulation juste après déclenchement.
- Valeur par défaut pour une intervention annulée avant le départ du véhicule de secours.

Pré-traitement :

Une transformation logarithmique ou un scaling sont à envisager pour aider à la prédiction de cette variable dont la distribution est asymétrique.



- **Identification des corrélations entre la target et les features potentielles**



Observations :

- On constate que la valeur AttendanceTimeSeconds est identique (corrélation à 1) avec notre variable cible.
- FirstPumpArriving_AttendanceTime a une corrélation de 0.79 avec la ResponseDuration. En faisant un focus sur cette variable on comprend qu'elle est égale au temps d'arrivée du véhicule arrivant en premier sur les lieux.
- SecondPumpArriving_AttendanceTime a une corrélation de 0.70 avec la ResponseDuration. En faisant un focus sur cette variable on comprend qu'elle est égale au temps d'arrivée du véhicule arrivant en second sur les lieux.

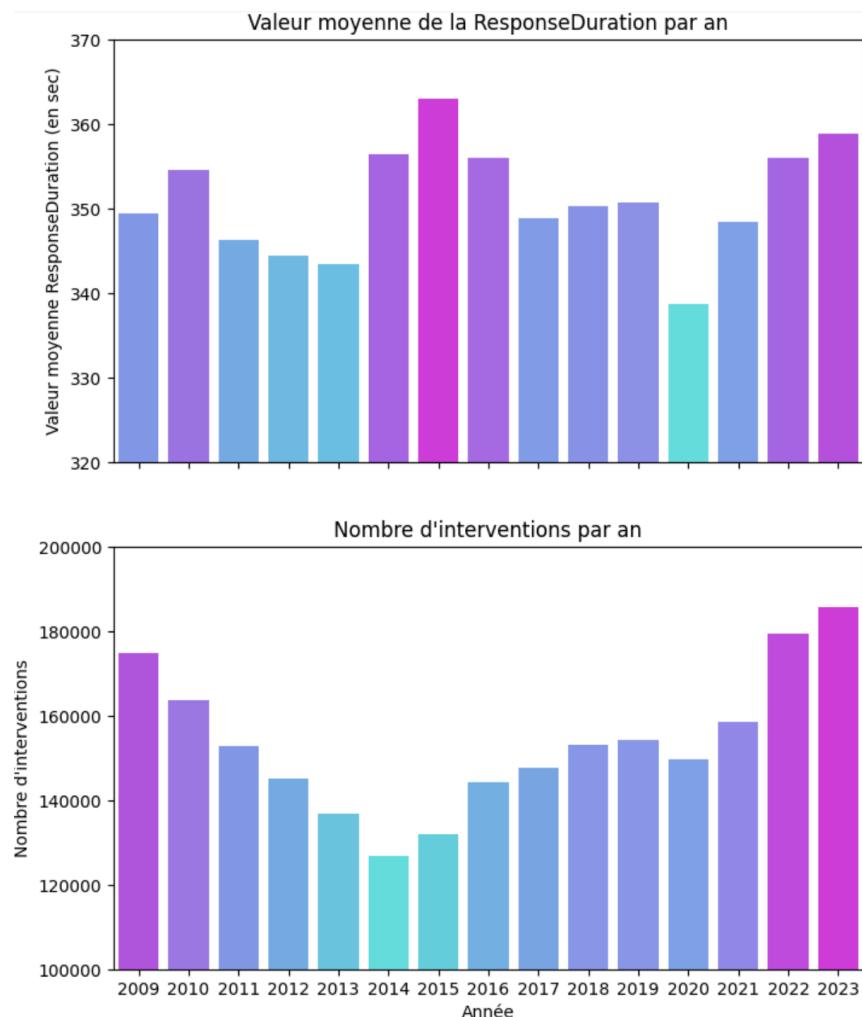
Pré-traitement :

- Les variables AttendanceTimeSeconds, FirstPumpArriving_AttendanceTime et SecondPumpArriving_AttendanceTime ne sont en fait pas des variables indépendantes de la ResponseDuration et nous ne **conserverons pas ces variables.**



Analyse temporelle

- Comparaison de la durée moyenne du “ResponseDuration” au nombre d'interventions comptabilisées **chaque année** (on ne retient pas 2024 sur ce graphe qui n'est pas une année complète).



Observations :

On peut observer des performances record sur le temps d'intervention en 2020 (l'année COVID).

On observe également de mauvaises performances en 2014, 2015 & 2016, années où le nombre d'interventions semble pourtant au plus bas.

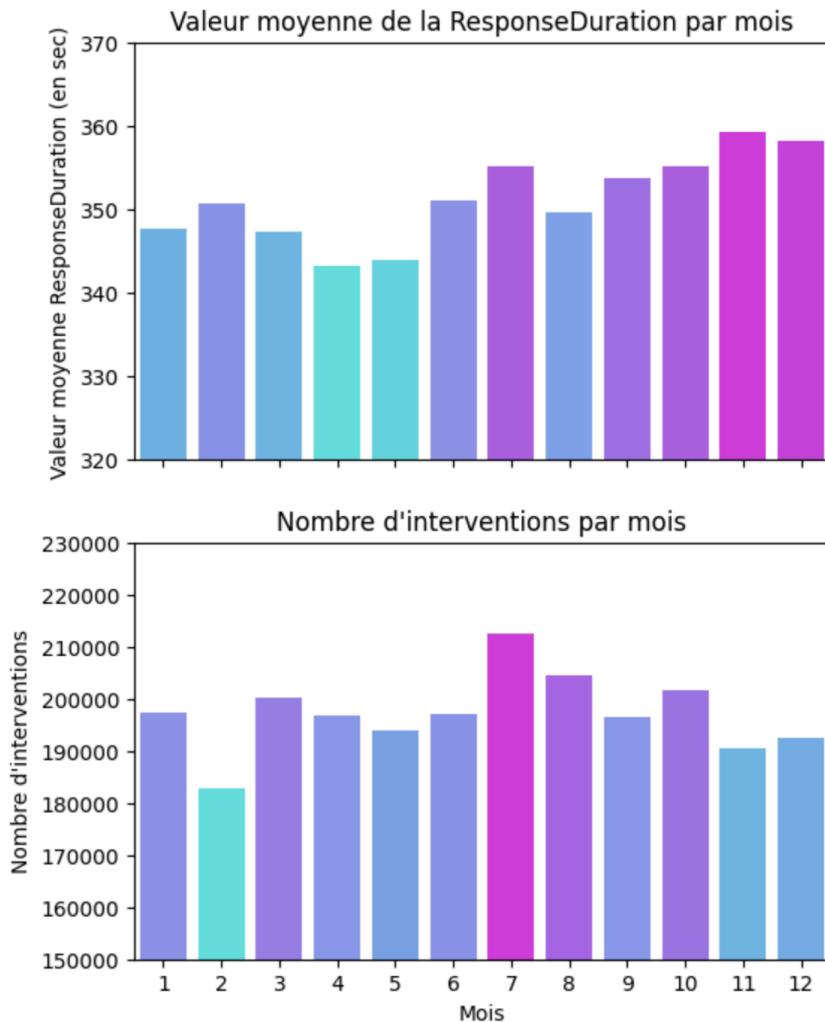
Hypothèse(s) :

Le confinement a diminué le trafic routier sur une partie importante de l'année et l'activité industrielle a diminué entraînant avec elle les causes d'incidents.

Pas de corrélation évidente visible à première vue.



- Comparaison de la durée moyenne du “ResponseDuration” au nombre d'interventions comptabilisées **chaque mois**



Observations :

On observe un pic du nombre d'incidents durant l'été (juillet & août).

On observe également une dégradation progressive le long de l'année civile du temps de réponse.

Pas de corrélation visible entre la valeur moyenne et le nombre d'interventions sur ce découpage.

Hypothèse(s) :

=> impact météorologique sur le nombre d'incidents (et particulièrement les incendies).

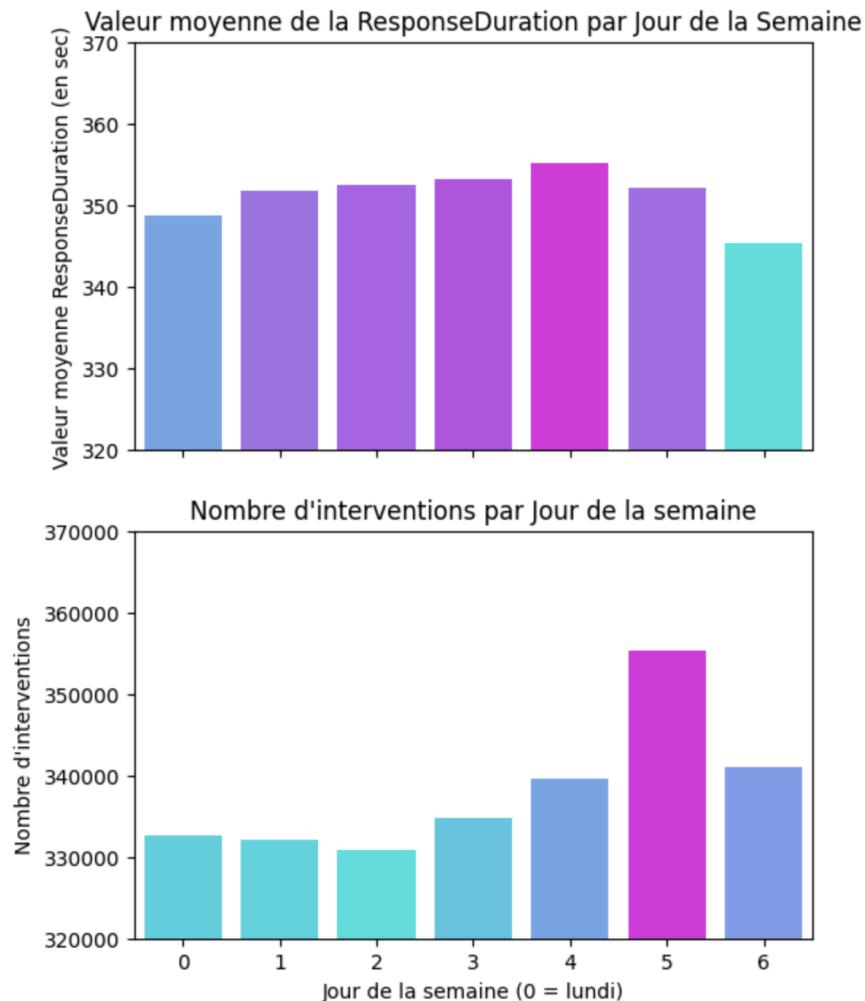
=> diminution des moyens financiers le long de l'année pouvant justifier une baisse de moyens pour répondre aux incidents.

Pré-traitement :

Un encodage cyclique de la variable “month” semble approprié.



- Comparaison de la durée moyenne du “ResponseDuration” au nombre d'interventions comptabilisées chaque **jour de la semaine**



Observations :

On note un temps de réponse meilleur les dimanches .

Pas de corrélation visible entre la valeur moyenne et le nombre d'interventions sur ce découpage.

Hypothèse(s) :

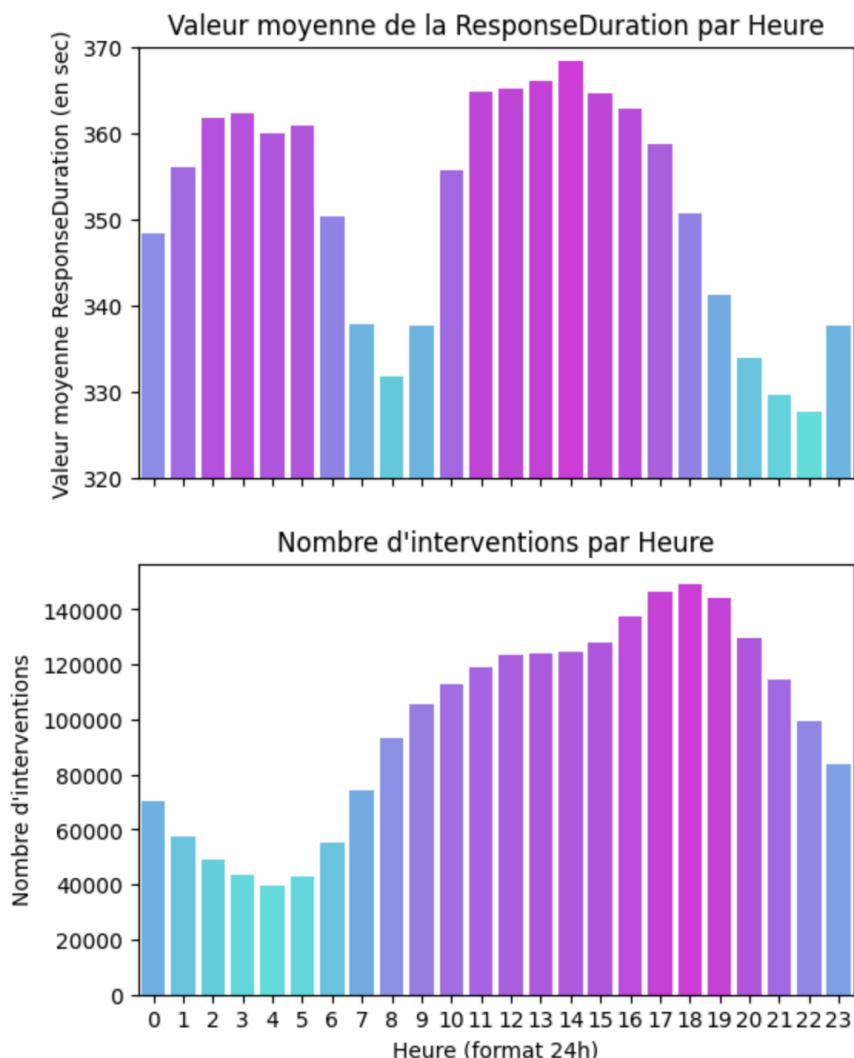
=> le trafic routier londonien est moins fort le dimanche.

Pré-traitement :

Un encodage cyclique de la variable “weekday” semble approprié.



- Comparaison de la durée moyenne du “ResponseDuration” au nombre d'interventions comptabilisées chaque **heure de la journée**



Observations :

On observe un mauvais temps de réponse pendant la nuit que l'on peut corrélérer à une baisse d'effectif disponible.

On observe également un excellent temps de réponse en début de matinée mais également durant la soirée (où le nombre d'incidents est pourtant très élevé).

Pas de corrélation visible entre la valeur moyenne et le nombre d'interventions sur ce découpage.

Pré-traitement :

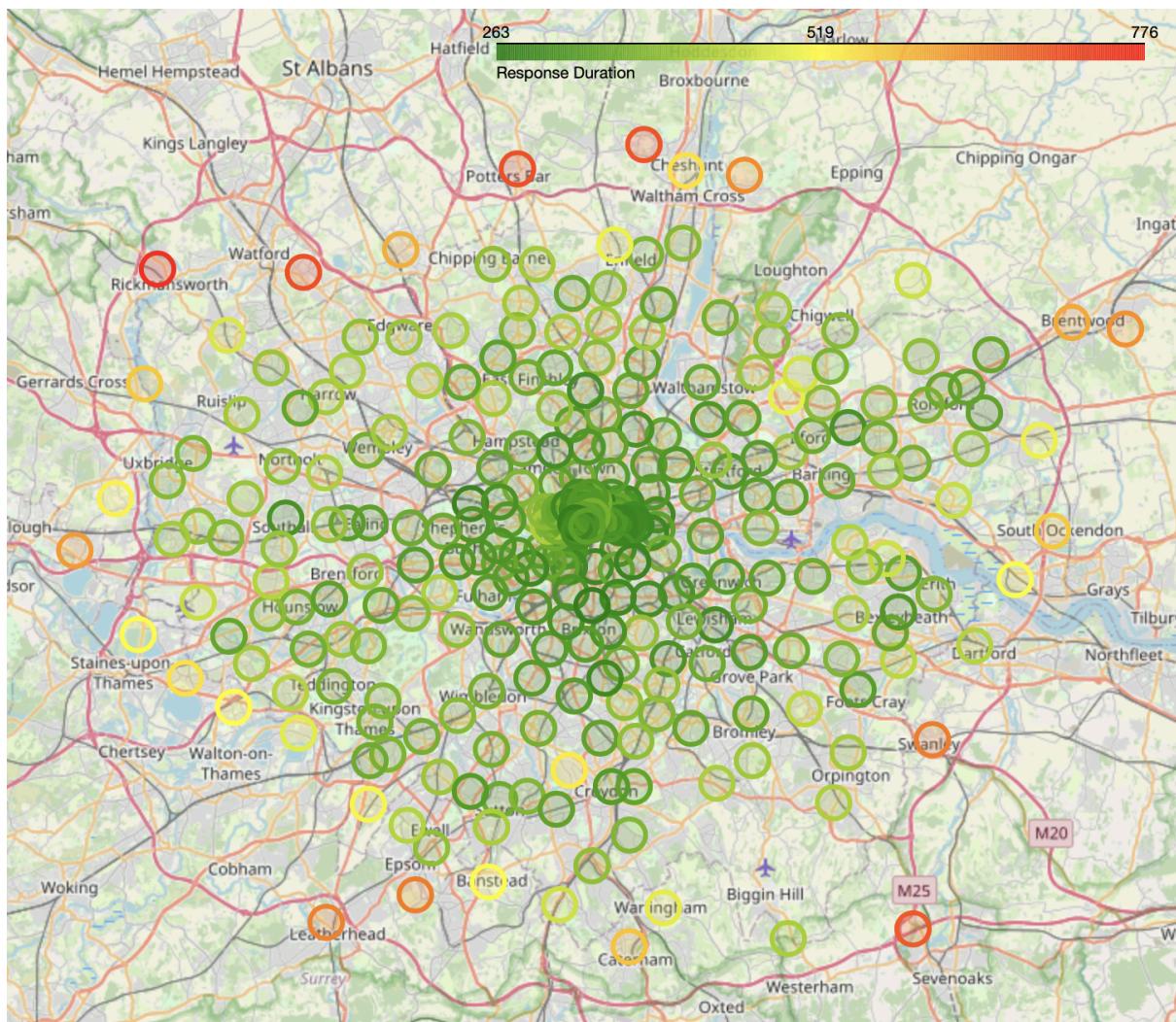
Un encodage cyclique de la variable “weekday” semble approprié.

Ce graphique nous montre que l'heure de la journée à laquelle l'incident a lieu a un **fort impact** sur notre valeur target.



Analyse spatiale

- Affichage de carte avec visualisation du temps moyen de réponse regroupé par "District Code"



Groupement géographique des temps moyen de réponse

Observations :

On observe que le temps moyen d'intervention augmente en périphérie de la ville.

Cela s'explique simplement par le maillage des stations plus dense dans le centre ville de Londres qu'en périphérie.

L'utilisation des variables Easting and Northing devraient permettre une prise en compte de ces disparités par le modèle de machine learning.



Rendu final : Rapport d'exploration, de data visualisation, de pre-processing des données et de modélisation



Observations :

Les valeurs sont centrées autour de la moyenne, avec une répartition symétrique des valeurs autour de cette moyenne. Cela illustre la concentration du nombre d'incidents au centre de Londres.

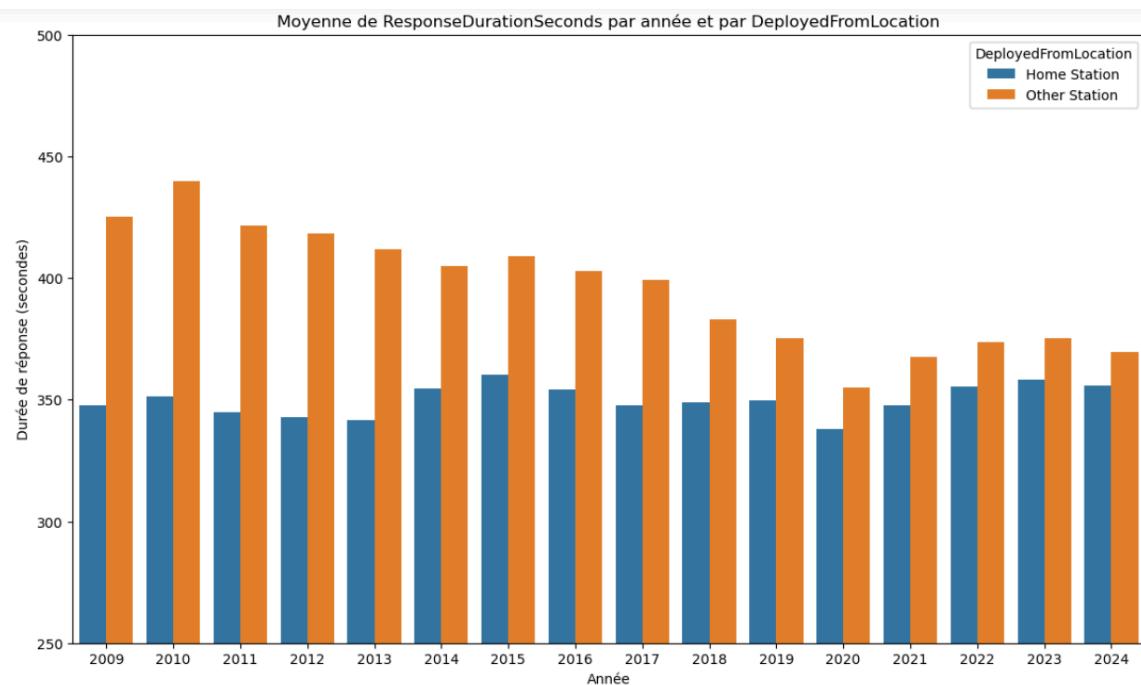
Pré-traitement :

Une normalisation des valeurs de Northing_rounded et Easting_rounded.



Analyse temporelle par station de déploiement

- Comparaison de la durée moyenne du “ResponseDuration” selon un déploiement en mission depuis la station la plus proche du lieu d’incident ou autres stations



Observations :

On observe que les déploiements en mission depuis la station la plus proche du lieu d’incident est plus efficace qu’un déploiement depuis une autre station que celle-ci.

On observe cependant une nette amélioration au fil des années de la gestion des mobilisation depuis d’autres stations que la station locale.

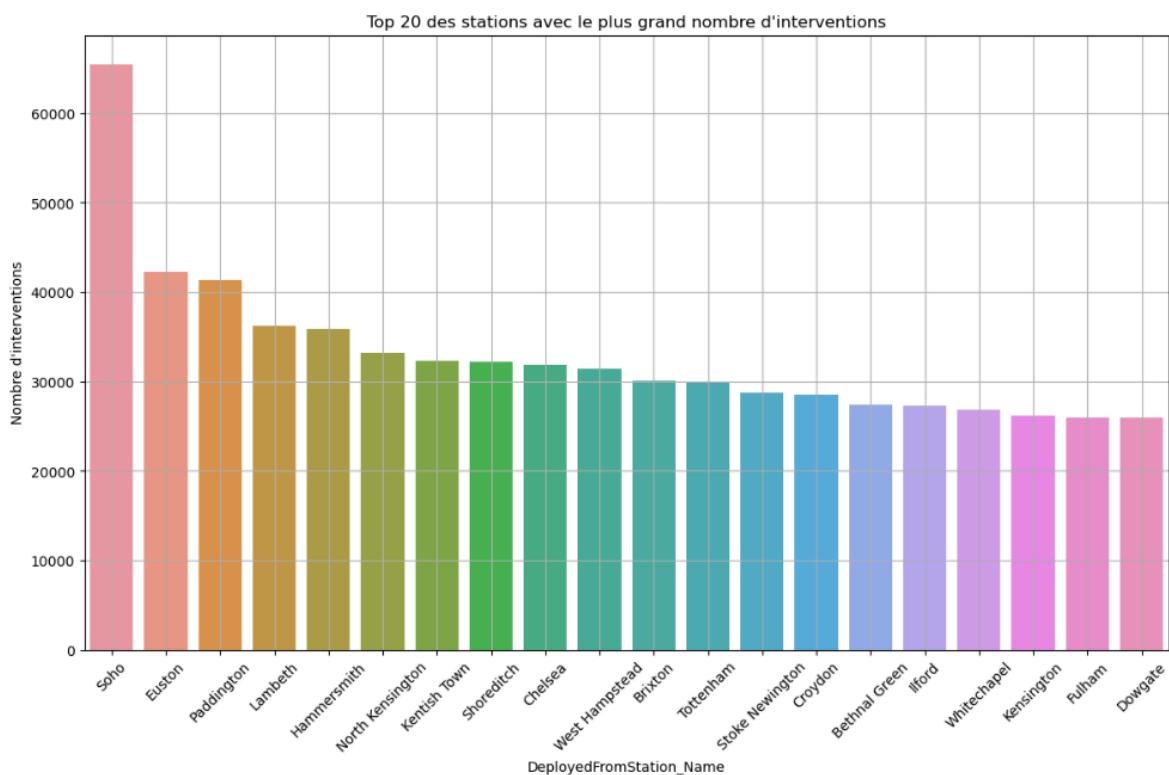
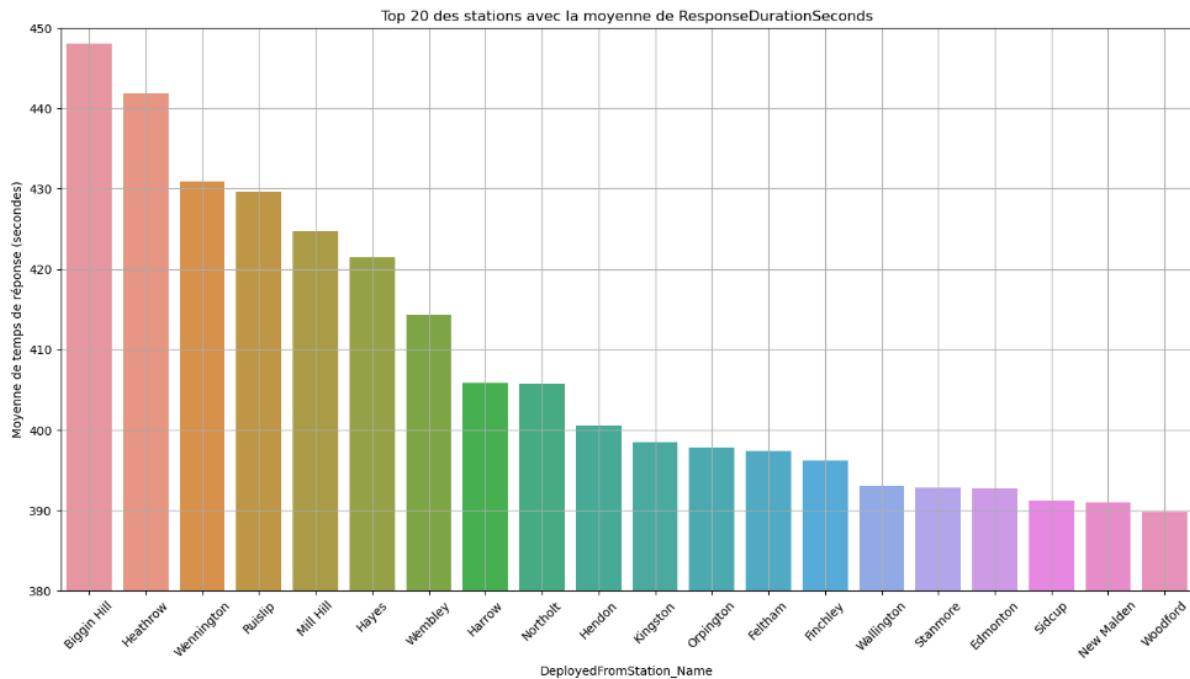
Pré-traitement :

Cette variable sera encodée de manière binaire.



Analyse de la performance par station

- Analyse du “ResponseDuration” par station de déploiement



Observations :

Dans le premier graphique, seul les Top 20 stations qui ont pris le plus de temps pour se déployer en mission sont affichées. Les 3 stations Hertfordshire, Staines et Esher étaient en service uniquement la première année. Nous avons donc filtré au

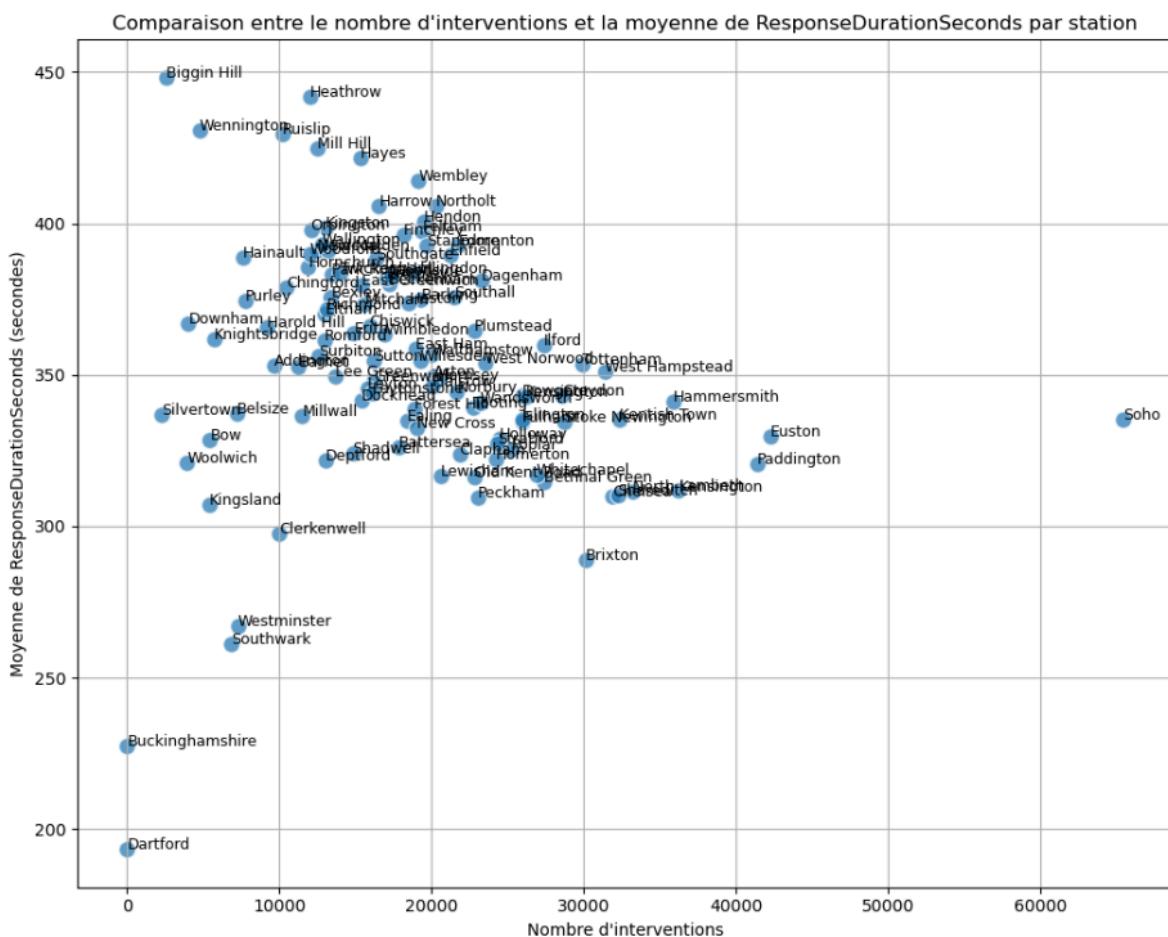


préalable ces trois valeurs extrêmes. Si nous les prenons en compte, ces trois stations ont un temps de réponse jusqu'à 400 secondes, soit 6,6 minutes de plus que la station Biggin Hills. Elles étaient par ailleurs situées à l'extérieur de l'agglomération Londonienne.

Il est donc clair que les mobilisations avec un temps de réponse aussi long sont des renforts envoyés par ces stations vers le centre ville.

Le deuxième graphique montre le Top 20 des stations avec le plus d'interventions depuis 2009. A noter que ces stations affichées ont toutes au moins 15 années de service.

- Analyse de la corrélation entre notre target et le nombre d'intervention



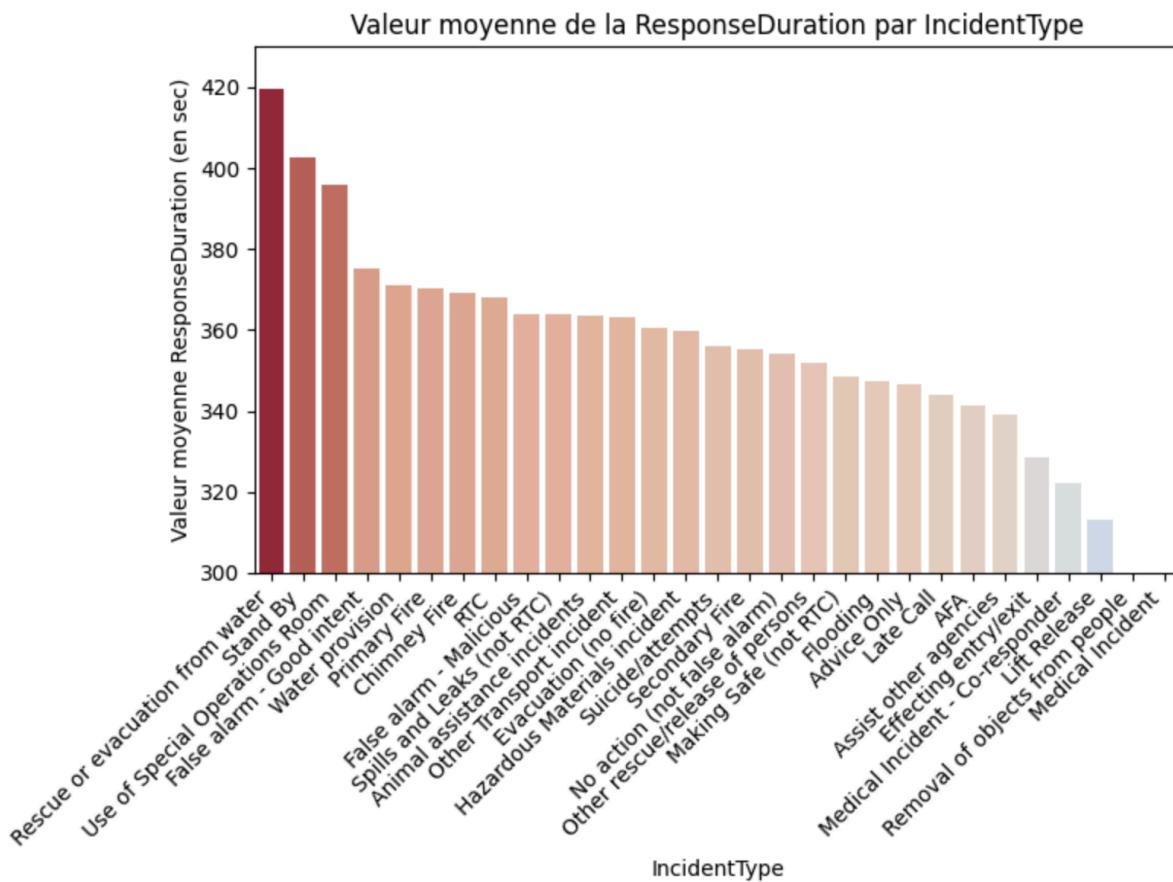
Observations :

On observe que les stations qui ont un temps de réponses relativement plus long que les autres ne sont pas celles qui ont le plus d'interventions, comme nous pouvons le voir dans les nuages de point ci-dessus. Ces nuages de points comparent le nombre d'interventions et la moyenne du temps de réponse des stations. L'objectif est de savoir s'il y a une dépendance entre notre target et le nombre d'interventions. Visuellement, la dépendance n'est pas évidente.



Analyse des type d'incidents

- Affichage des durées moyennes d'intervention en fonction du type d'Incident



Observations :

On constate une différence conséquente de la moyenne du temps de réponse entre les différentes catégories d'incidents.

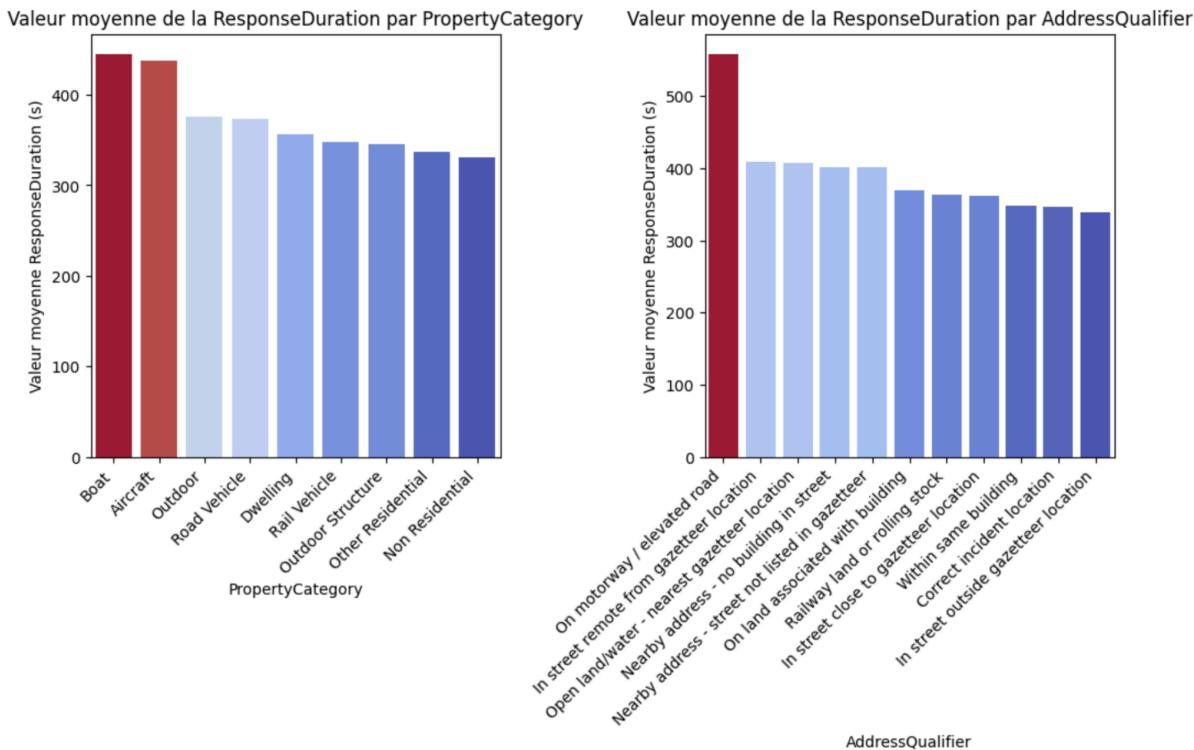
Pré-traitement :

Un encodage catégoriel de cette variable sera nécessaire pour son utilisation.



Analyse des catégories de propriétés & des qualificatifs d'adresse

Affichage des durées moyennes des temps de réponses



Observations :

On distingue une nette différence entre les délais de réponse pour intervenir sur des moyens de transport difficilement accessibles :

- Bateau : sur un fleuve (Tamise)
- Avion : dans un aéroport - accès restreints
- Autoroute ou route aérienne : mise en sécurité pour intervention pouvant être difficile + blocage / embouteillage de la route en cas d'accident.

Pré-traitement :

Un encodage catégoriel de cette variable sera nécessaire pour son utilisation.



D. Pre-processing et feature engineering

Afin de faire du machine learning sur notre dataset, nous allons devoir réduire notre dataset pour améliorer la performance et la durée d'entraînement de nos modèles tout en conservant un maximum d'informations.

Les choix réalisés dans le tableau sont justifiés et illustrés par les graphes présentés dans le paragraphe [Visualisations et Statistiques](#).

On retiendra le code couleur suivant : **Variable retenue / variable non-retenue**

Variable	Description	Justification du choix des variables non-retenues	Traitement des NaNs/ Normalisation/ Encodage
Variables en lien avec le déploiement des véhicules			
IncidentStationGround	Nom de la station locale	Pour synthétiser la station d'origine du véhicule, nous choisissons les variables DeployedFromStation_Name et DeployedFromLocation qui associée à des variables géographiques conserve l'information.	-
DeployedFromStation_Name	Nom de la station ayant déployé le véhicule		Suppression des 23 entrées NaN. Encodage de la variable catégorielle (One-Hot)
DeployedFromLocation	Le véhicule provient-il de la station locale ou d'une autre station		Encodage binaire
PlusCode_Description	La mobilisation du véhicule est une mobilisation initiale ou une demande de renfort.	-	Encodage de la variable catégorielle (One-Hot)
NumStationsWithPumpsAttending	Nombre de stations ayant des véhicules présent sur l'incident en question	-	Normalisation
NumPumpsAttending	Nombre de véhicules présents sur l'incident	-	Normalisation
PumpOrder	Numéro d'arrivée sur les lieux de l'incident du véhicule.	-	Suppression des NaNs Normalisation
FirstPumpArriving_DeployedFromStation	Nom de la station qui a déployé le premier véhicule arrivé sur les lieux	Variables trouvables par logique entre la variable de pumpOrder et le nom de la station de déploiement DeployedFromStation_Name	-
SecondPumpArriving_DeployedFromStation	Nom de la station qui a déployé le second véhicule arrivé sur les lieux		-



Variables de temps			
DateAndTimeCalled	DateTime (supposée) de l'appel	Données incohérentes avec le reste des données temporelles.	-
DateAndTimeMobilised	DateTime de la mobilisation	Choix réalisé de la variable DateAndTimeMobilised pour en extraire les variables descriptives du temps.	Transformation en 4 valeurs : year, month, weekday, hour. => Normalisation de la variable year. => Encodage cyclique des valeurs de month, weekday, hour.
DateAndTimeMobile	DateTime de la mise en mouvement du véhicule		
DateAndTimeArrived	DateTime de l'arrivée sur les lieux du véhicule		
DateAndTimeLeft	DateTime de départ du véhicule du lieu de l'incident	-	Target
ResponseDuration	Durée entre la mobilisation et l'arrivée sur les lieux de l'incident		
AttendanceTimeSeconds	Durée entre la mobilisation et l'arrivée sur les lieux de l'incident	Identique à la valeur "target" - découvert dans la matrice de corrélation	-
FirstPumpArriving_AttendanceTime	Durée entre la mobilisation et l'arrivée sur les lieux de l'incident du PREMIER véhicule	Identique à la valeur target pour les véhicules arrivant en premier	-
SecondPumpArriving_AttendanceTime	Durée entre la mobilisation et l'arrivée sur les lieux de l'incident du SECOND véhicule	Identique à la valeur target pour les véhicules arrivant en second	-
Variables de géo-localisation de l'incident			
IncGeo_BoroughName	Nom de la localité	Pour faire apparaître l'information de la localisation précise de l'incident nous choisissons les variables Easting et Northing qui sont des variables quantitatives ne présentant pas de NaNs et restant très précises.	-
Postcode_district	Code postal du quartier		-
IncGeo_WardName	Nom de quartier		-
IncGeo_WardNameNew	Nom de quartier avec nouvelle appellation		-
UPRN	Numéro unique de propriété		-
USRN	Numéro unique de rue		-
Easting_rounded	Valeur permettant une localisation sur l'axe		Normalisation



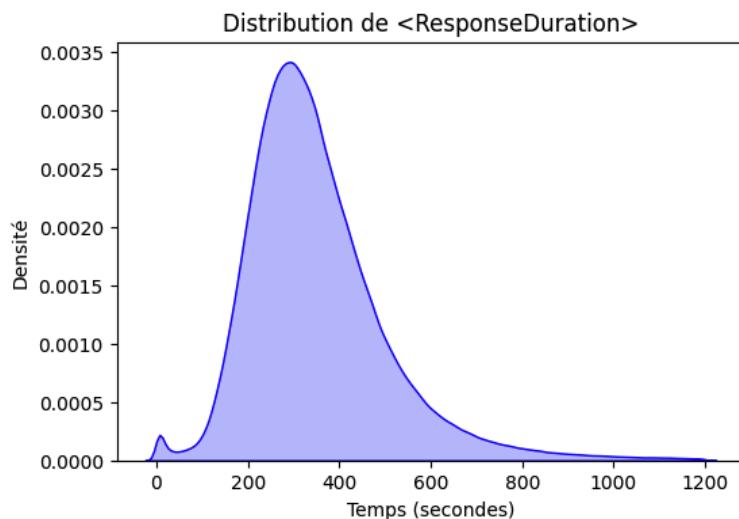
	est-ouest avec une précision de 50 m		
Northing_rounded	Valeur permettant une localisation sur l'axe nord-sud avec une précision de 50 m		Normalisation
Variables descriptives de l'incident			
IncidentType	Variable détaillant le contenu de False Alarm, Fire & Special Service	IncidentType est une variable créée par nos soins pour rassembler (sans perdre) l'information des variables IncidentGroup, StopCodeDescription et SpecialServiceType en une seule variable.	Encodage de la variable catégorielle (One-Hot)
IncidentGroup	False Alarm / Fire / Special Service		-
StopCodeDescription	Variable détaillant le contenu de False Alarm et de Fire		-
SpecialServiceType	Variable détaillant le contenu de SpecialService		-
Variables descriptives du lieu de l'incident			
PropertyCategory	Catégorie de propriété où a lieu l'intervention	Les variables PropertyCategory et AddressQualifier classifient au bon niveau les niveaux les différentes typologies (.PropertyType en compte plus de 300 différentes)	Encodage de la variable catégorielle (One-Hot)
.PropertyType	Type de propriété où a lieu l'intervention		-
AddressQualifier	Description qualifiant le lieu de l'incident réel correspondant à la catégorie ci-dessus		Encodage de la variable catégorielle (One-Hot)
Variables diverses			
PumpCount	Pas de compréhension claire de la variable	Variable non comprise et donc non retenue.	-
PumpMinutesRounded	Temps passé par le véhicule en question sur l'incident (arrondi à l'heure si inférieur à 60 minutes)	Variable successive dans le temps de la variable target et sans impact sur celle-ci.	-
Notional Cost (£)	Coût approximatif associé à la durée de mobilisation et de présence du véhicule	Variable successive dans le temps de la variable target et sans impact sur celle-ci.	-
NumCalls	Nombre d'appels reçus pour l'incident	-	Suppression des NaN Normalisation



III. Modélisation

A. Description de la première approche : modèle de régression

Dans un premier temps, l'approche retenue a été le **modèle de régression**. En effet, la variable cible (le temps nécessaire à la mobilisation des pompiers) est une variable continue ayant une valeur comprise entre 0 à 1250 secondes.



Les métriques observées dans ce cas sont les suivantes :

- **R²** : le coefficient de détermination qui mesure la proportion de la variance de la variable dépendante target qui est expliquée par les variables indépendantes du modèle. Il est compris entre 0 et 1. L'objectif principal est d'obtenir une valeur de R² la plus proche possible de 1.
- **RMSE** : l'erreur quadratique moyenne qui est la racine carrée de la moyenne des carrés des erreurs. Elle mesure l'écart type des résidus (erreurs de prédiction) et donne une indication de la taille moyenne de ces erreurs en unités des variables prédites. L'objectif est d'obtenir une RMSE proche de 0.
- **MAE** : l'erreur absolue moyenne est la moyenne des valeurs absolues des erreurs de prédiction. Elle mesure la taille moyenne des erreurs en unités des variables prédites, sans tenir compte de la direction des erreurs (positives ou négatives). L'objectif est également d'avoir une valeur de la MAE proche de 0.

R² est la métrique que à utiliser pour comparer nos modèles, la RMSE et la MAE permettent d'affiner l'analyse au besoin.



Pour synthétiser les différents entraînements réalisés l'ensemble des résultats peuvent être trouvés dans le tableau ci-dessous :

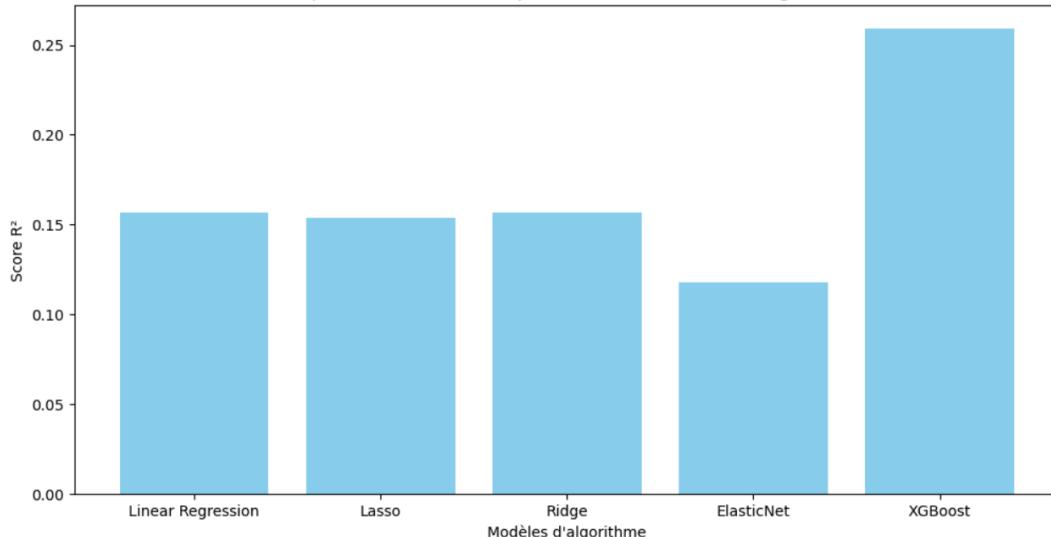
Modèle	Hyper-paramètres	Divers	Standardisation	Features	Réduction	R ²	RMSE	MAE
Linéaire (RL)	sans	Avec variables météo Sans DeployedFromStation_Name	Standard MinMax	72 PCA (95%) : 22	PCA (95%) : 22	0.147	141.6	102.4
Ridge	sans					0.147	141.6	102.4
Lasso	sans					0.145	141.7	102.4
ElasticNet	alpha=1.0 l1_ratio=0.5 random_state=42					0.116	144.1	104.5
XGBoost	random_state=42					0.254	132.4	94.1
Linéaire (RL)	sans	Avec variables météo Sans DeployedFromStation_Name	MinMax	72 PCA (95%) : 22	PCA (95%) : 22	0.026	151	111
Ridge	sans					0.026	151	111
Lasso	sans					0.019	152	112
ElasticNet	alpha=1.0 l1_ratio=0.5 random_state=42					0.009	153	112
XGBoost	random_state=42					0.114	144	105
Linéaire (RL)	sans	Avec variables météo Sans DeployedFromStation_Name	Standard	72 sans	embedded method estimator : lasso threshold : 1e=10 => 24	0.176	139	100
Ridge	sans					0.026	151	111
Lasso	sans					0.019	151	111
ElasticNet	alpha=1.0 l1_ratio=0.5 random_state=42					0.009	152	112
XGBoost	random_state=42					0.114	144	104
Linéaire (RL)	sans	Avec variables météo Sans DeployedFromStation_Name	Standard	72 embedded method estimator : lasso threshold : 1e=10 => 24	embedded method estimator : lasso threshold : 1e=10 => 24	0.163	140	101
Ridge	sans					0.025	151	111
Lasso	sans					0.019	152	112
ElasticNet	alpha=1.0 l1_ratio=0.5 random_state=42					0.119	143.8	104
XGBoost	random_state=42					0.33	125	86



Modèle	Hyper-paramètres	Divers	Standardisation	Features	Réduction	R ²	RMSE	MAE
Linéaire (RL)	sans	Sans DeployedFromStation_Name Sans data météo	Standard	66	PCA (95%) : 19	0.157	140.72	101.77
Ridge	sans					0.157	140.72	101.77
Lasso	sans					0.154	140.93	101.82
ElasticNet	alpha=1.0 l1_ratio=0.5 random_state=42					0.118	143.89	104.26
XGBoost	random_state=42					0.259	131.87	93.66
Linéaire (RL)	sans	Sans DeployedFromStation_Name Sans data météo	MinMax	66	PCA (95%) : 21	0.025	151.28	111.13
Ridge	sans					0.025	151.28	111.13
Lasso	sans					0.019	151.76	111.53
ElasticNet	alpha=1.0, l1_ratio=0.5, random_state=42					0.009	152.55	112.17
XGBoost	random_state=42					0.144	141.77	102.51
XGBoost	random_state=42	Sans variables météo Sans DeployedFromStation_Name	Standard	66	embedded method estimator : lasso threshold : 1e=10 => 22	0.332	125.19	86.54

Comparaison des modèles

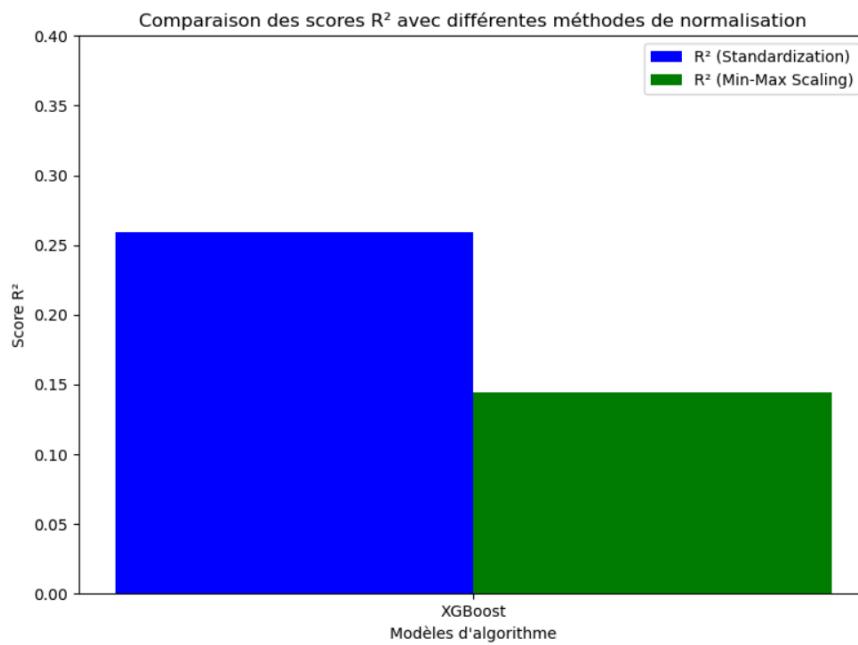
Comparaison des scores R² pour différents modèles de régressions





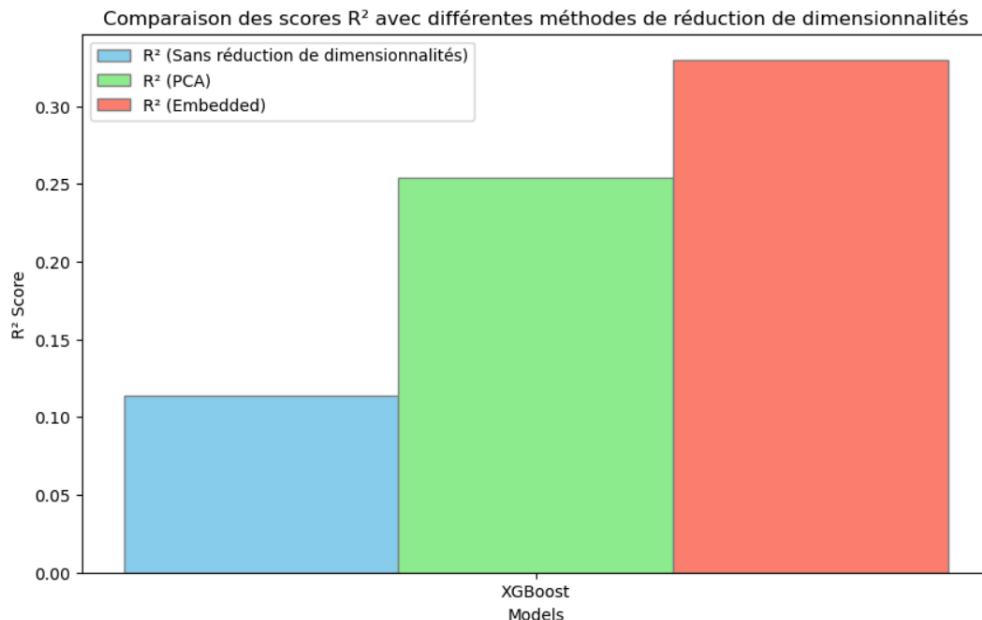
Suite à plusieurs entraînements (cf.figure ci-dessous), le modèle **XGBoost** retourne généralement de meilleurs scores R^2 que les autres modèles, comme montré dans le graphique ci-dessous, les scores R^2 sont tirés d'un entraînement sans variation d'autres paramètres.

Comparaison de la méthode de standardisation



Le graphique ci-dessus montre que la méthode de normalisation **standard** est plus performante que celle du Min-Max à tout autre paramètre égal.

Comparaison de la méthode de réduction de dimensions





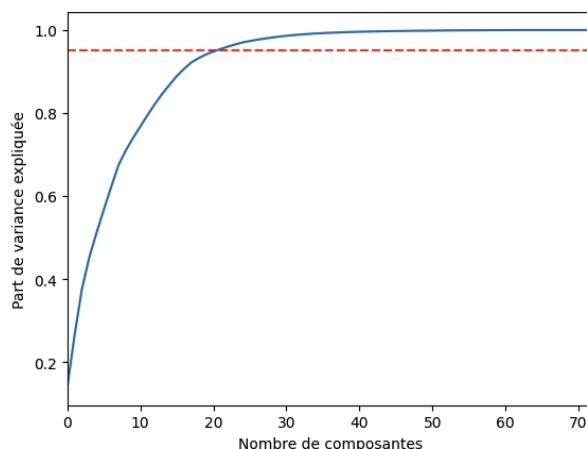
Le graphique ci-dessus montre qu'utiliser la **embedded method** rend ici des scores R² plus pertinents par rapport au PCA ou sans aucune méthode de réduction de dimensionnalités.

Le modèle XGBoost, combiné à une normalisation standard et à une méthode de réduction de dimensions intégrée, atteint le score le plus élevé possible en termes de régression même si les scores R2 restent insatisfaisants.

Optimisations

Ci-dessous quelques-unes des pistes d'optimisation développées plus haut pour améliorer les performances des différents modèles testés et/ou de réduire le temps de calcul pour leur entraînement.

- Non-utilisation de la variable “DeployedFromStation_Name” : cette variable est constituée des noms des 120 stations de pompiers présentes dans Londres et génère 120 variables supplémentaires lors de l'encodage sans en améliorer les résultats.
- Ajout de données météorologiques à l'heure, pour enrichir le dataframe des variables d'entrée, les features ajoutées sont les suivantes :
 - Température (à 2m du sol)
 - Humidité relative (à 2m du sol)
 - Pluie
 - Code météo : correspond au “type de météo” plus la valeur est élevée moins la météo est bonne.
 - Vitesse moyenne du vent (à 10 m du sol)
 - Vitesse des rafales de vent (à 10 m du sol)
- Utilisation de la PCA pour diminuer la durée d'entraînement sans diminuer la performance des modèles : projection de l'information comprise dans les 74 variables présentes après encodage sur 22 nouvelles variables en conservant 95% de la variance.





- Utilisation de l'embedded method pour diminuer la durée d'entraînement sans diminuer la performance des modèles (estimator = lasso, threshold = 1e-10). Ci-dessous le spectre des features retenues (en noir).



- Utilisation du Grid Search : afin d'améliorer les modèles, le Grid Search permet d'identifier des hyper paramètres intéressants. Ici les hyper-paramètres testés n'ont pas permis d'améliorer les performances des modèles (cf. tableau ci-dessous)

Modèle	Hyper-paramètres	Divers	Standardisation	Features	Réduction	R ²	RMSE	MAE
XGBoost (gridsearch)	Best params : 'colsample_bytree': 1.0 'gamma': 0 'learning_rate': 0.1 'max_depth': 5 'n_estimators': 100 'subsample': 0.8	Avec variables météo Sans DeployedFromStation_Name Simplification de la variable incident_type	Standard	54	PCA (95%) : 21	0.2207	121	96
		- Avec variables météo - Sans DeployedFromStation_Name		72	embeded method estimator : lasso threshold : 1e=10 => 24	0.27	121	92

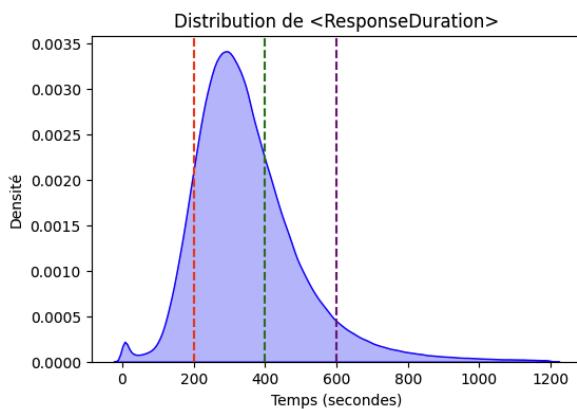
Comme décrit dans le tableau de synthèse présenté dans la partie suivante, les résultats obtenus par cette approche ne sont pas vraiment satisfaisants : aucun modèle n'ayant permis d'obtenir de R² supérieur à 0.35.



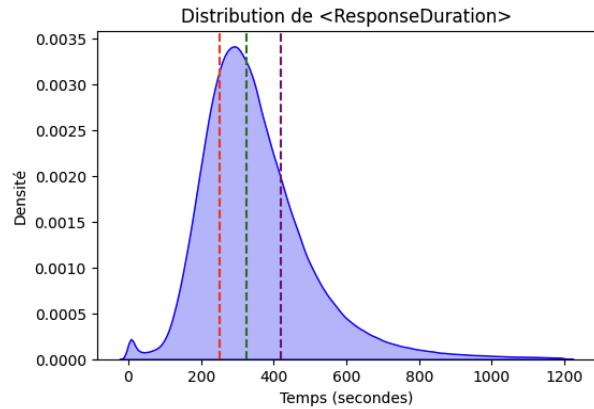
B. Description de la seconde approche : modèle de classification

Face aux résultats décevants que proposent les modèles basés sur la régression, il a été décidé de catégoriser la variable de durée d'intervention en plusieurs catégories de tailles variables. Cette nouvelle approche transforme la méthode de résolution en **modèle de classification multiple**.

Ci-dessous plusieurs possibilités de découpage de notre variable :



Découpage arbitraire [0, 200, 400, 600, 1300]



Découpage par quartile [0, 251, 325, 421, 1300]

L'objectif reste le même que précédemment : prédire le temps nécessaire à la mobilisation des pompiers sur les lieux de l'incidents, mais la réponse sera fournie sous-forme de fourchette de durée catégorisées en 0, 1, 2 & 3 :

- **0 = temps considéré comme “rapide”**
- **1 = temps considéré comme “moyen”**
- **2 = temps considéré comme “lent”**
- **3 = temps considéré comme “très lent”**

La métrique à observer pour comparer les performances des modèles est **l'accuracy** : elle mesure le pourcentage de prédictions correctes parmi toutes les prédictions effectuées. L'objectif est d'obtenir une valeur la plus proche possible de 1.

Une **confusion matrix** permet en complément par la suite de distinguer les classes où le modèle se trompe (faux positifs/ faux négatifs) et ainsi d'analyser les biais du modèle.



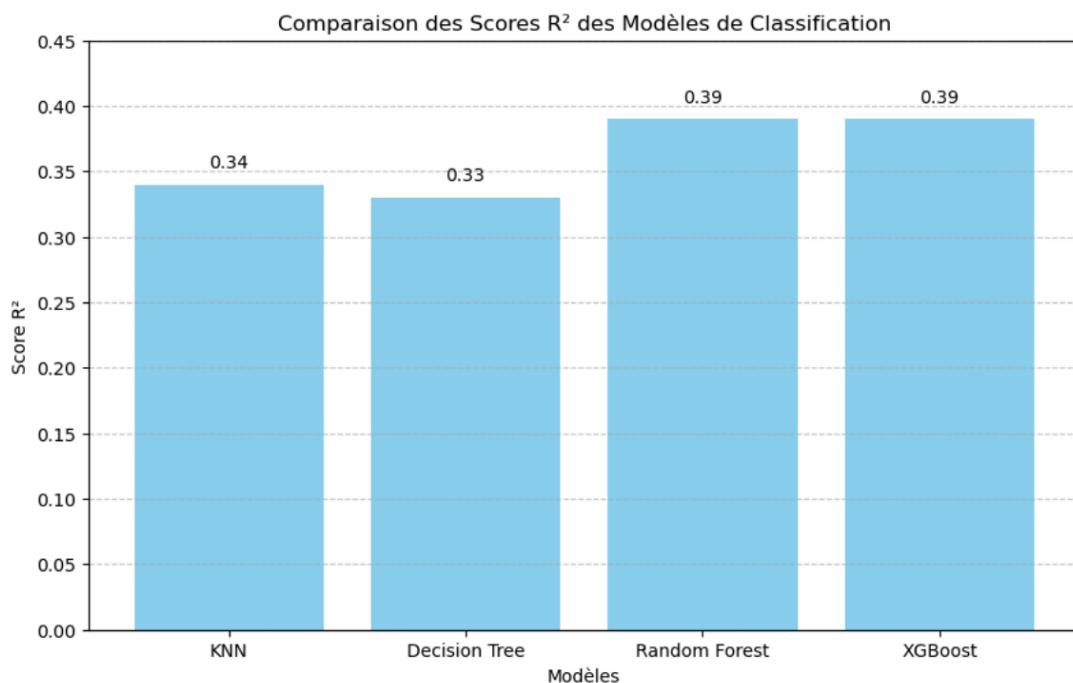
Pour synthétiser les différents entraînements réalisés l'ensemble des résultats peuvent être trouvés dans le tableau ci-dessous :

ML	Classification Target	Hyperparamètres	Sampling (under/over/...)	Features	Réduction	Accuracy
KNN	[0, 251, 325, 421, 1300]	sans	RandomUnderSampling (42) 400000 entrées (100k par classe)	74	PCA (95%) : 22	0.34
DecisionTree		sans				0.33
RandomForest		sans				0.39
XGBOOST		n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42				0.39
RandomForest	[0, 200, 400, 600, 1300]	sans	RandomUnderSampling (42) 400000 entrées (100k par classe)	74	PCA (95%) : 22	0.35
XGBOOST		n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42				0.29
RandomForest	[0, 251, 325, 421, 1300]	sans	RandomUnderSampling (42) 200000 entrées (50k par classe)	74	PCA (95%) : 22	0.39
XGBOOST		n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42				0.39
RandomForest	[0, 251, 325, 421, 1300]	sans	RandomUnderSampling (42) 200000 entrées (50k par classe)	74	sans	0.41
XGBOOST		n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42				0.41
RandomForest		sans				0.34
XGBOOST		n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42				0.37



Pour la sélection du modèle avec les paramètres les plus pertinents, on procède à un raisonnement en entonnoir qui nous permet de commencer par une approche large et inclusive, puis de filtrer progressivement les options pour identifier le modèle et les paramètres les plus pertinents, en utilisant des méthodes et des pratiques similaires à celles du deep learning.

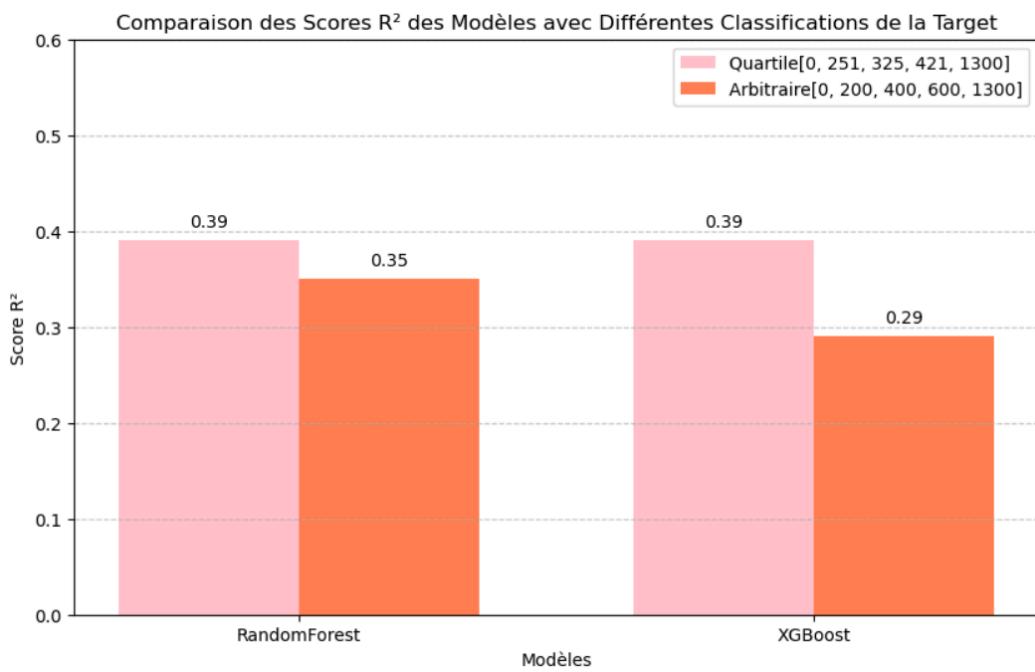
Comparaison des scores R² des modèles de classification



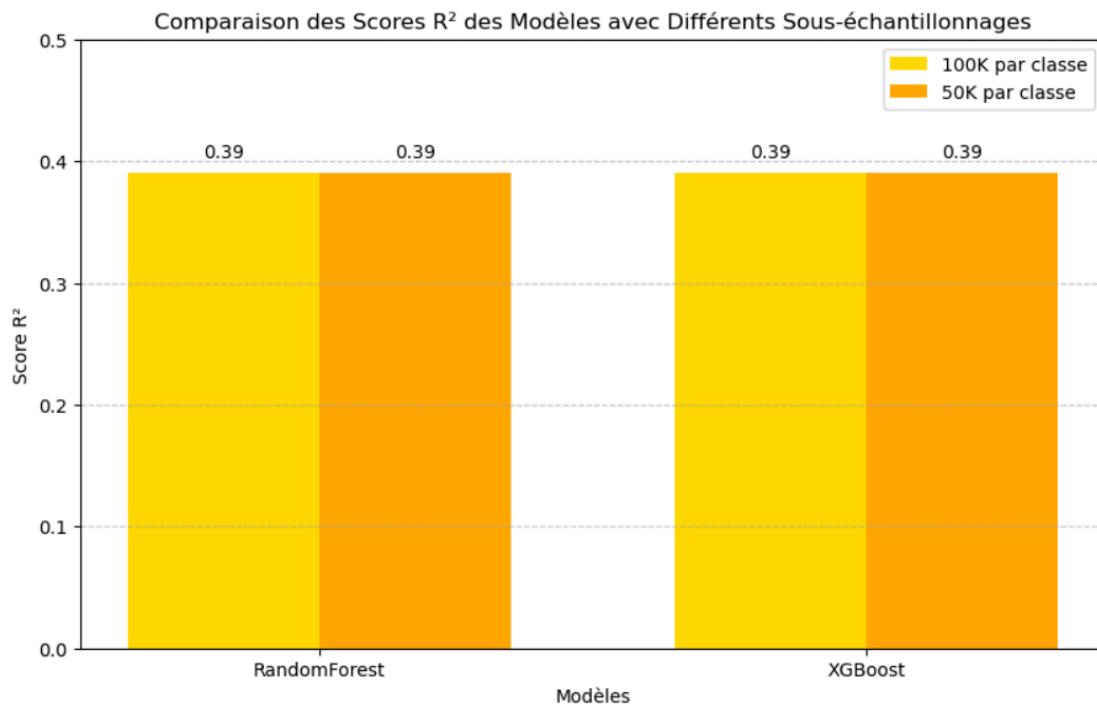
Random Forest et XGBoost se démarquent clairement avec des scores identiques et les plus élevés parmi les quatre modèles. Ils sont donc sélectionnés pour une analyse plus approfondie.

Évaluation des scores R² selon le découpage de la variable cible

Le graphique ci-dessous montre que les scores R² des deux modèles sont moins meilleurs avec une classification de la cible par quartile en comparaison avec un découpage plus arbitraire. **Random Forest** semble être plus robuste face à cette modification, conservant un score R² relativement élevé dans les deux cas.



Impact des Techniques de Sous-échantillonnage sur les Scores R² des Modèles RandomForest et XGBoost



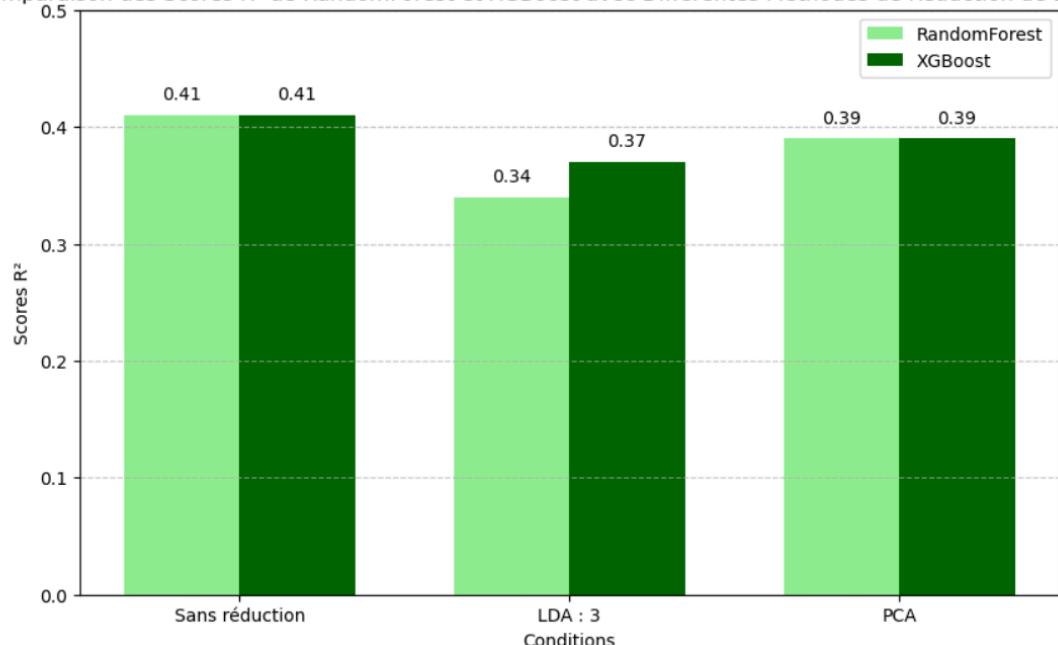
Les modèles RandomForest et XGBoost ont des **scores identiques** sous les deux conditions de sous-échantillonnage. La taille du sous-échantillonnage (100K vs. 50K) ne semble pas influencer les scores R² pour ces modèles et ce jeu de données spécifique. Il reste intéressant de garder le sous-échantillonnage le plus fort pour **réduire le temps de calcul**.



Impact des Méthodes de Réduction de Dimension sur les Scores R² de RandomForest et XGBoost

Les modèles RandomForest et XGBoost affichent des **scores identiques sans réduction de dimension**, mais leurs performances divergent avec les méthodes de réduction. Une LDA affecte négativement la performance (surtout sur le RandomForest). Le PCA offre des performances relativement stables pour les deux modèles, avec des scores légèrement inférieurs à ceux sans réduction, mais mieux que ceux obtenus avec une LDA (notamment pour RandomForest).

Comparaison des Scores R² de RandomForest et XGBoost avec Différentes Méthodes de Réduction de Dimension



Au-delà de la performance des résultats, on note que le XGBoost est bien plus "rapide" pour proposer des résultats que le modèle RandomForest.



Optimisations

Ci-dessous quelques-unes des pistes d'optimisation développées en complément ou remplacement des optimisations déjà exploitées lors de la régression linéaire pour améliorer les performances des différents modèles testés et/ou de réduire le temps de calcul pour leur entraînement.

- Utilisation du grid search

ML	Classification Target	Hyperparamètres	Sampling (under/over/...)	Features	Réduction	Accuracy
RandomForest GridSearch	[0, 251, 325, 421, 1300]	Best parameters : 'bootstrap': False, 'max_depth': 30, 'min_samples_split': 10, 'n_estimators': 200	RandomUnderSampling (42) 400000 entrées			0.43
XGBoost GridSearch	[0, 251, 325, 421, 1300]	'colsample_bytree': 1.0 'gamma': 0.1 'learning_rate': 0.2 'max_depth': 6 'n_estimators': 200 'subsample': 0.8	RandomUnderSampling (42) 200000 entrées (50k par classe)	74	sans	0.48

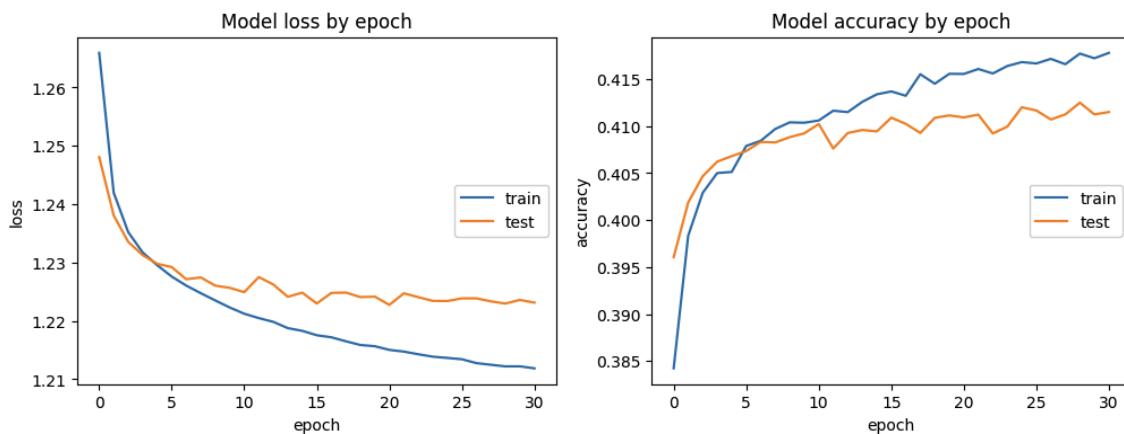
Dans le cas de la classification, le gridsearch a permis d'identifier des hyperparamètres permettant une optimisation amenant les meilleurs scores trouvés à ce stade.



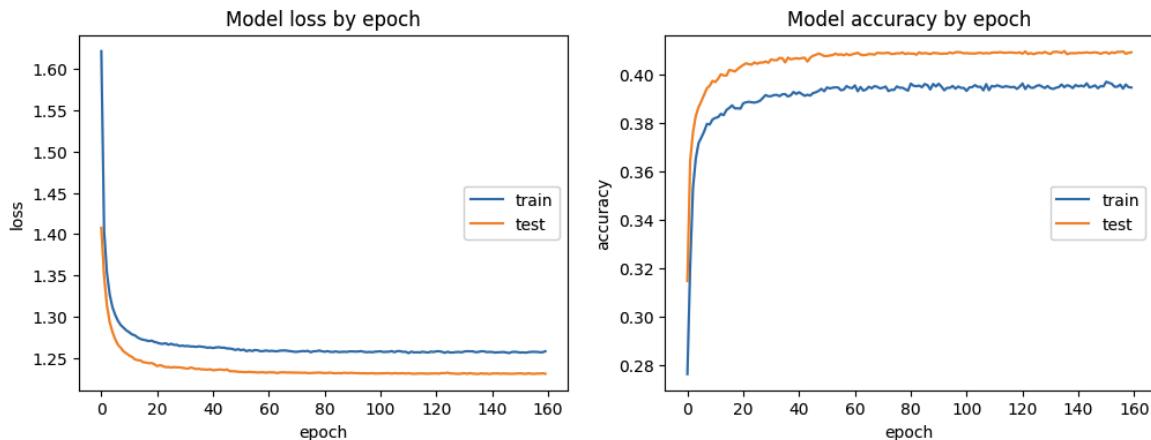
- Utilisation d'un modèle dense de Deep learning :

Deep Learning	Classification Target	Hyperparamètres	Sampling (under/over/...)	Features	Réduction	Accuracy
Dense model	[0, 251, 325, 421, 1300]	4 couches dense epochs = 100, batch_size = 200, validation_split = 0.2	RandomUnderSampling (42) 200000 entrées (50k par classe)	74	sans	0.41
Dense model amélioré	[0, 251, 325, 421, 1300]	Idem entrecoupée de dropout & batchnormalization	RandomUnderSampling (42) 200000 entrées (50k par classe)	74	sans	0.41

Model dense fully connected :



Modèle précédent amélioré avec des couches dropout et des couches batch normalization :



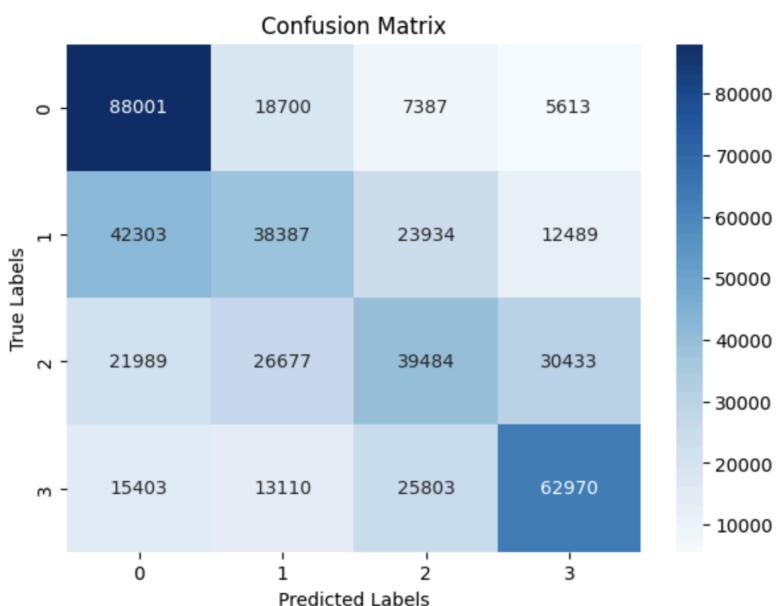
Les résultats du Deep Learning permettent un résultat correct sans révolutionner les résultats précédents. On y constate en plus de cela un léger overfitting sans convergence entre le train et le test.



C. Interprétation des résultats

Globalement, les résultats obtenus par les modèles développés **ne sont pas satisfaisants**. Dans le meilleur des cas, notre modèle se trompe plus d'une fois sur deux pour prédire la catégorie de temps de réponse ("rapide", "moyen", "lent", "très lent").

Ci-dessous la matrice de confusion issue du modèle le plus performant (XGBoost Classifier avec hyper paramètres optimisés) :



On constate une dispersion des mauvaises prédictions avec un pic notamment sur la prédiction de la classe 0 pour des valeurs "vraies" de la classe 1.

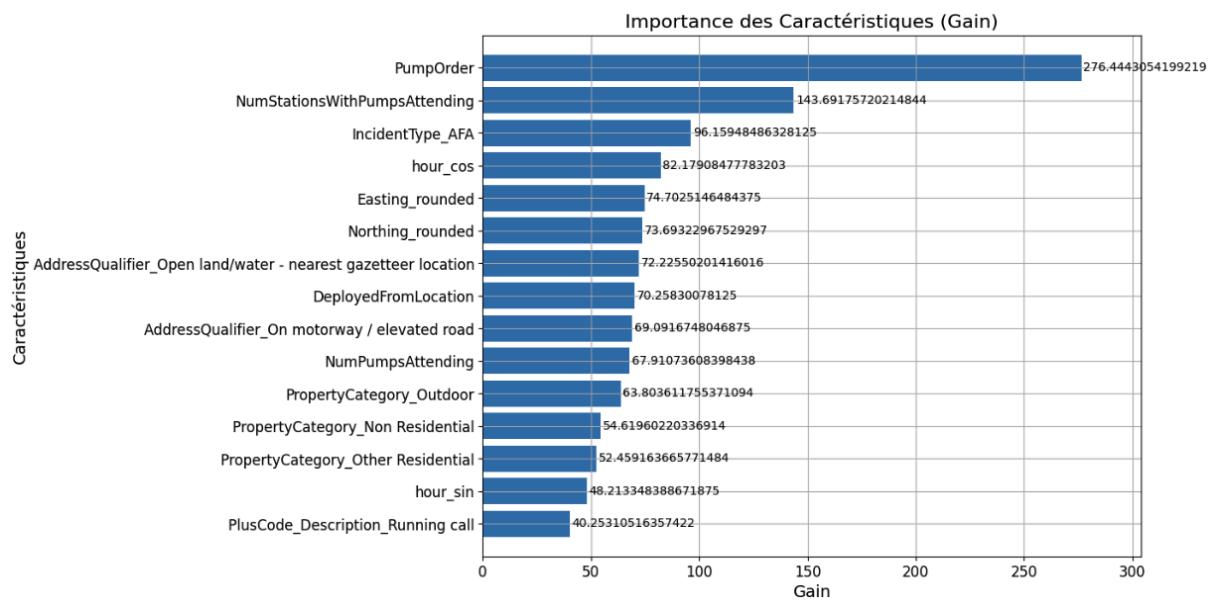
Interprétabilité

Il est également intéressant de comprendre l'impact des différentes variables pour sur le choix de la classe réalisé par le modèle.

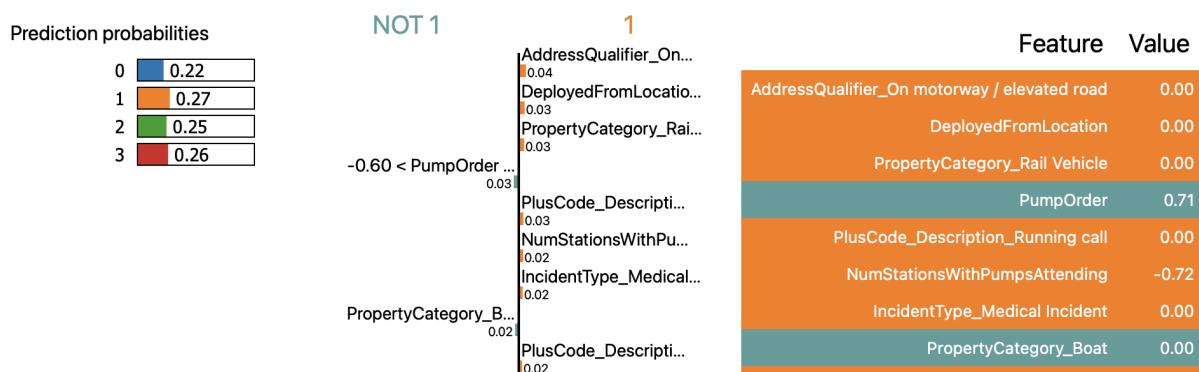
Ci-dessous une vue de l'interpréteur du modèle XGBoost qui donne un gain pour les features en fonction de l'importance qu'elles donnent au choix par le modèle.

On constate que :

- le PumpOrder (soit le numéro d'arrivée du camion sur les lieux) et le NumStationWithPumpAttending (soit le nombre de stations desquelles les camions sont appelés sur l'incident) sont les deux variables les plus influentes sur ce choix.
- Le nombre d'incidents, l'heure de l'incident et la localisation (easting & northing) sont également dans le top 6 des variables classées par gain.



Pour mieux comprendre quelle variable influe sur quelle classe, il est possible d'utiliser l'interpréteur Lime :



- Il est possible d'analyser pour chaque classe de prédiction les features qui permettent au modèle de choisir cette classe ou au contraire de la refuser.
- Vu le nombre de features il apparaît que son analyse globale n'apporte peu au problème à ce stade et il serait plus intéressant à regarder sur un modèle plus performant.



Pour conclure ci-dessous un tableau récapitulatif de l'analyse de l'impact des tentatives réalisées pour améliorer le modèle :

Description des différentes tentatives réalisées pour améliorer la performance du modèle	Impact Perf (+ / = / -)	Impact Temps Calcul (+ / = / -)
Passage sur un problème de classification	+	=
Choix du modèle	+	+
Réduction de dimension via la PCA	=	+
Réduction de dimension via LDA	-	+
Réduction de dimension via Embedded method	+	+
Ajout des données météorologiques	=	=
Standardisation MinMax	-	=
Standardisation Normale	+	=
Ré-équilibrage du jeu de données pour la classification	+	=
Taille de l'Undersampling	=	+
Utilisation de GridSearch	+	-
Utilisation du DeepLearning	+	-



IV. Conclusion

A. Difficultés rencontrées

- La compréhension des datasets a été la première difficulté rencontrée dans le projet : malgré un fichier explicatif fourni avec les datasets téléchargés (fichiers metadata), certaines variables sont restées incomprises. Cette étape a d'ailleurs été la plus longue à réaliser pour l'équipe.
- Les datasets de base sont de plus très volumineux sur le nombre de features comme sur le nombre d'entrées, impliquant des difficultés pour :
 - les intégrer pour le traitement par une partie du groupe,
 - obtenir un dataset identique pour l'entraînement des modèles : la coordination de l'équipe sur le pre-processing n'a pas été simple,
 - la durée des calculs pour les phases d'entraînement des modèles de machine learning avant réduction de dimensions et/ou undersampling (à modérer car aussi dépendante de la puissance computationnelle = machines locales).
- Certains membres de l'équipe ont également rencontré des difficultés vis-à-vis du timing d'intégration des compétences de cours pour répondre aux problématiques du projet.
- Enfin, la principale difficulté rencontrée a été la pertinence des résultats obtenus remettant en cause le travail de feature engineering, preprocessing et le choix des modèles utilisés pour répondre au problème donné.



B. Bilan

Pour conclure sur le projet, il est d'abord nécessaire de rappeler l'objectif donné en introduction : **prédire le temps nécessaire à un véhicule de pompiers pour arriver sur les lieux d'un incident** après avoir été sollicité et mobilisé (une fois la décision d'intervenir prise).

Deux approches de modélisation ont été prises par le projet :

- Une **approche de régression** avec pour objectif une prédiction relativement précise (avec une fenêtre d'erreur) de la valeur en seconde du temps de réponse.
- Une **approche de classification** avec pour objectif de catégoriser le temps d'intervention dans des classes déterminée par l'équipe : "rapide", "moyen", "lent", "très lent".

Dans les deux cas les résultats obtenus n'ont pas été à l'attendu :

- Un coefficient de détermination ne dépassant pas les 0.33 avec une marge d'erreur de l'ordre de 100 secondes dans le cas de la régression.
- Une accuracy ne dépassant pas 0.5 dans le cas de la classification.

La principale conclusion en l'état est l'**inutilité des modèles développés en l'état** par qui que ce soit, il reste néanmoins utile de poursuivre le projet pour :

- la London Fire Brigade qui pourrait s'en servir pour prédire les durées d'intervention de ses équipes et anticiper des faiblesses dans le temps ou dans l'espace dans le court terme,
- Les services publics (gouvernement, mairie de Londres, ...) pourraient s'en saisir pour identifier ces mêmes faiblesses et mettre en place des plans d'action sur le long terme.

L'analyse data proposée en première partie de projet a tout de même permis d'identifier des impacts sur le temps moyen de mobilisation de :

- La **géolocalisation** => temps de mobilisation plus long en périphérie qu'en hyper centre,
- La **temporalité** => le jour de la semaine et l'heure ont un impact notable,
- La **typologie** d'intervention.

Ce bilan négatif est à mitiger car il a permis pour la première fois à l'équipe de mettre en œuvre un certain nombre de compétences acquises lors de la formation.

Il a également permis d'identifier des pistes pour poursuivre ce projet et amener peut-être à de meilleurs résultats utiles dans le futur. Ces pistes sont détaillées dans le paragraphe suivant.



C. Suite(s) du projet

En ouverture, plusieurs options s'offrent à nous pour mieux répondre à notre problématique :

- 1) Revoir (une fois de plus) le **feature engineering** : certaines variables existantes méritent une analyse plus fine que celle accordée dans le temps du projet (la typologie d'incidents,)
- 2) Améliorer la **classification** proposée (idéalement en échangeant avec le métier (la Fire Brigade de Londres en l'occurrence) pour mieux identifier les durées d'intervention que l'on peut considérer comme problématiques.
- 3) **Enrichir** à nouveau le dataset à l'aide de données pertinentes (données de circulation ? moyens financiers des brigades rapporté au nombres d'habitants de la zone qu'elles couvrent ? ...)
- 4) **Séparer le problème** en plusieurs, par exemple en considérant que les moyens mis en oeuvre ne sont pas comparables et donc les variables impactantes également :
 - Un problème "incendie"
 - Un problème "secours à personne (hors incendie)"
- 5) Sur les modèles :
 - Développer l'approche du machine learning avec des **pouvoirs de calcul** permettant de faire tourner des modèles plus gourmands ou avec des **hyperparamètres** plus diversifiés.
 - Développer l'approche **Deep Learning** avec des réseaux de neurones plus complexes,

Dans le cas d'une amélioration des performances des modèles de manière substantielle, il sera alors possible de proposer ceux-ci à des interlocuteurs orientés métier.

D. Bibliographie

Liens utilisés pour l'importation des données utilisées dans le projet :

- [London Fire Brigade Incident Records - London Datastore](#)
- [London Fire Brigade Mobilisation Records - London Datastore](#)
- [Docs | Open-Meteo.com](#)
- [Download UK Postcodes with Latitude and Longitude](#)



E. Annexe

```
Project Architecture
-----
LICENSE
README.md           <- the top-level README for developers using this project.
data                <- not on Github (only in .gitignore)
|   processed        <- the final, canonical data sets for modeling.
|       dataviz.csv
|       ML_data.csv
|
|   raw              <- the original, immutable data dump.
|       Incident
|           LFB Incident data from 2009 - 2017.csv
|           LFB Incident data from 2018 onwards.xlsx
|
|       Mobilisation
|           LFB Incident data from 2009 - 2017.csv
|           LFB Incident data from 2018 onwards.csv.xlsx
|           LFB Incident data from 2018 onwards.xlsx
|
|   merged_data.csv
|
external            <- External data, added to broaden the dataset or to help plot
|   ukpostcodes.csv
|   weather.csv
|
models              <- Where models are saved to be loaded again if needed
|
notebooks           <- Jupyter notebooks. Naming convention is a number and also exploration & dataviz
|   __init__.py      <- Makes src a Python module
|   utils.py         <- Scripts that contains useful functions
|   01_importation_fusion.ipynb    <- Scripts to turn imported data into a single dataset
|   02_weather.ipynb     <- Scripts to import weather data and save it int the project
|   03_featuring.ipynb    <- Scripts to do the feature selection and prepare the dataset to be processed for the machine learning
|   04_regression.ipynb    <- Scripts to run regression models
|   05_classification.ipynb    <- Scripts to run classification models
|   06_deep_learning.ipynb    <- Scripts to run deep_learning models
|   KF_dataviz.ipynb      <- Scripts to run data visualization (by Keyvan)
|   KF_exploration.ipynb    <- Scripts to run data exploration
|   SL_dataviz.ipynb      <- Scripts to run data visualization (by Suzanne)
|
reports             <- The reports that you'll make during this project as PDF
|   figures          <- Generated graphics and figures to be used in reporting
|
requirements.txt    <- The requirements file for reproducing the analysis environment, e.g.
|                     generated with `pip freeze > requirements.txt`
environement.yml   <- Environment export generated by `conda env export > environment.yml`
|
src                <- Source code for use in this project.
|   __init__.py      <- Makes src a Python module
|   utils.py         <- Scripts that contains useful functions
|   main.py          <- Scripts to use to run and save the different models directly (regression / classification / deep learning)
|   read_models.py    <- Scripts to use to load the different models from the models folder
|   preprocessing.py  <- Scripts that contains the necessary functions to do the preprocessing
|   regression.py    <- Scripts that contains the necessary functions to run regression models
|   classification.py <- Scripts that contains the necessary functions to run classification models
|   deep_learning.py  <- Scripts that contains the necessary functions to run deep learning models
```



Pour l'utilisation du code et la génération des modèles, après avoir enregistrer les données conformément à l'architecture ci-dessus, suivre l'ordre suivant :

- 1) Utiliser les notebooks numérotés dans l'ordre :

01_importation_fusion.ipynb
02_weather.ipynb
03_featuring.ipynb

- 2) Il existe ensuite 2 possibilités :

- a) Utilisation des notebooks (présents dans le dossier notebooks dans l'architecture du projet) :

04_regression.ipynb
05_classification.ipynb
06_deep_learning.ipynb

- b) Utilisation du *main.ipynb* situé dans le dossier src dans l'architecture du projet qui appellera les fonctions sous forme modulaire et se servira des codes suivants :

utils.py
preprocessing.py
regression.py
classification.py
deep_learning.py
utils.py

- 3) De manière optionnelle, il est également possible d'ouvrir les notebooks *exploration.ipynb* et *dataviz.ipynb* présents dans le dossier notebooks.
- 4) Il est également possible de charger les modèles enregistrés à l'aide du notebook *read_model.ipynb* présent dans le dossier src.