



DataScientest • com

London pump.PY



Promotion : Octobre 2021 _ Continu _ DS

Christophe Battistelli

Grégory Javey

Ludovic Marécaux

Remerciements	3
1. Introduction	4
2. La base de données	5
2.1. Premier contact avec la base de données	5
2.2. La visualisation des données	7
2.3. Le nettoyage des données	12
2.4. La création de variables basées sur les variables existantes	18
2.5. Intégration de variables externes	23
3. Modèles de prédiction	27
3.1. Prédiction du nombre de camions à envoyer sur l'intervention	27
3.1.1 Travaux préparatoires	27
3.1.2 La métrique choisie	31
3.1.3 Les modélisations	31
3.2. Prédiction de l'atteinte de l'objectif des 360s	35
3.3. Prédiction du temps d'arrivée	47
3.3.1 Travaux préparatoires	47
3.3.2 Modélisation	49
3.3.2.1 Métriques et choix des modèles	49
3.3.2.2 Approfondissement : prévenir les sous-estimations	55
3.3.2.3 Rapprochement avec classification	58
4. Conclusion	61
5. Annexes	63
Annexe 01 : variables du fichier LFB Incident data (1/2)	64
Annexe 01 : variables du fichier LFB Incident data (2/2)	65
Annexe 02 : variables du fichier LFB Mobilisation data	66
Annexe 03 : suppression de 29 variables jugées inutiles	67
Annexe 04 : traitement des valeurs manquantes	68
Annexe 05 : jours de congés	68
Annexe 06 : densité du trafic londonien	70
Annexe 07 : variables du fichier final	71

Remerciements

Nous tenons tout d'abord à remercier Raphaël, de nous avoir suivis tout au long de notre étude, depuis décembre 2021 et pour ses conseils pertinents.

Merci également à Laurène, qui en tant que responsable de la promotion DS octobre 2021 nous a parfaitement guidé tout au long de l'année sur les différents objectifs à atteindre.

Enfin, un grand merci à Daniel et toute l'équipe de DataScientest pour leur disponibilité, leurs réponses à nos questions ainsi que et l'animation des MasterClass et différents ateliers.

Toutes ces personnes ont contribué, par leur disponibilité et leur bonne humeur, à rendre notre parcours enrichissant et motivant.

1. Introduction

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.

Pour la période 2017-2021, la 6^{ème} version du "London Safety Plan" (document rassemblant les missions et objectifs de la brigade des pompiers de Londres, source : <https://www.london-fire.gov.uk/about-us/london-safety-plan/>) était en vigueur, définissant les priorités afin notamment de réduire le temps d'arrivée sur site des camions pour rendre la ville de Londres plus sûre. Un des objectifs fixés était l'arrivée sur chaque lieu d'incident en moins de 360s.

A partir de l'année 2022 et pour les 5 prochaines années, un nouveau plan d'action, London Pump.Py est mis en place. L'objectif principal sera d'analyser et prédire le temps d'intervention et de mobilisation de la brigade des pompiers de Londres.

Pour cela, nous mettrons en place trois modélisations. Dans un premier temps, nous créerons un modèle capable de définir le nombre de camions à envoyer en fonction des éléments connus au moment de l'appel. Ensuite, nous prédirons si les véhicules envoyés sur l'intervention arriveront dans le délai objectif des 360s. Et enfin, nous affinerons notre étude en estimant le temps d'arrivée sur site des premiers secours.

Pour ce travail, nous possédons deux tables composées de jeux de données répertoriant toutes les informations des interventions entre Janvier 2018 et Octobre 2021 :

- Les informations sur l'appareil mobilisé, son lieu de déploiement et les heures d'arrivée sur les lieux de l'incident et
- Les informations sur la date et le lieu de l'incident ainsi que sur le type d'incident traité.

L'ensemble du travail sera effectué sur Google Workspace et notamment l'outil Google Colab pour le partage des notebooks.

Le dossier contenant les documents relatifs à ce projet sont mis à disposition dans un dossier sur Google Drive, organisé en 3 répertoires :

- **data_inputs** : les fichiers de base du projet, ainsi que ceux qui nous ont été utiles pour enrichir les données ;
- **data_outputs** : les diverses sorties des notebooks (la base de données nettoyée, ainsi que les modèle de machine learning
- **notebooks** : les notebooks de nos travaux, classés selon le plan du présent rapport.

([Lien vers le dossier](https://drive.google.com/drive/folders/18SsEHdrwC7M5pxxXwWJRTn-s_lazPpQ?usp=sharing) : https://drive.google.com/drive/folders/18SsEHdrwC7M5pxxXwWJRTn-s_lazPpQ?usp=sharing)

2. La base de données

2.1. Premier contact avec la base de données

Avant même de rentrer dans le détail, nous avons été amenés à faire quelques arbitrages, qui ont été principalement guidés par la nécessité de pouvoir exploiter les données facilement sur Google Colab (volume, temps de traitements).

Le premier arbitrage concerne la période des données à exploiter.

Pour chacun des 2 jeux de données mis à notre disposition, nous avons la possibilité d'exploiter :

- soit un fichier remontant à 2009 (01/01/2009 - 31/10/2021) ;
- soit un fichier des 3 dernières années (01/01/2018 - 31/10/2021).

Nous avons choisi de considérer pour ce projet, les données des **3 dernières années** (nous verrons ultérieurement que le volume d'informations est déjà considérable).

Le second arbitrage concerne le format des fichiers. Nous avons fait le choix de convertir les fichiers disponibles au format excel (xlsx) en format csv.

Même si les fichiers excel (xlsx) sont moins volumineux que leur version csv, il s'est avéré que leur lecture (dans Google Colab, pour une exploitation dans le module Pandas) était très longue, comparé à la version csv.

Dès lors, notre base de travail est constituée des 2 fichiers suivants :

- *"LFB Incident data Last 3 years.csv"*
- *"LFB Mobilisation data Last 3 years.csv"*

Nous allons maintenant présenter brièvement l'architecture de ces données, et les variables.

Chaque incident signalé aux pompiers de Londres (LFB = London Fire Brigade) fait l'objet d'un enregistrement, avec des données précises, identifiant notamment les dates, lieux et types d'incident, ainsi que la caserne de pompiers dont il dépend en théorie.

C'est l'objet principal de la table issue du fichier *"LFB Incident data Last 3 years"*.

A la suite de quoi, les pompiers organisent la réponse opérationnelle, en envoyant un ou plusieurs véhicules de secours sur les lieux de l'incident.

A des fins de compréhension et d'analyse, chacun des véhicules ("pump") intervenant sur l'incident (peu importe la caserne dont il provient), est enregistré, ce qui permet principalement de suivre les étapes de l'intervention, depuis l'heure de l'appel à mobilisation dudit véhicule, jusqu'à son heure de départ du lieu de l'incident (fin de mission).

On trouvera notamment des informations sur sa provenance, afin d'identifier s'il a été envoyé de la caserne où il est habituellement basé, ou bien d'une autre caserne.

Toutes ces informations sont dans la table issue du fichier "**LFB Mobilisation data Last 3 years**".

Dans "**LFB Incident data Last 3 years**" sont ensuite reportés les résumés des données pour les deux premiers véhicules arrivés sur les lieux de l'incident (temps d'attente et caserne de provenance).

D'autres données relatives aux coûts théoriques de traitement des incidents sont également consignées dans ce tableau.

Comme évoqué précédemment, nous rappelons que les fichiers du jeu de données contiennent ces informations pour la **période du 01/01/2018 au 31/10/2021**.

Nous pouvons dès maintenant dresser une synthèse générale de ces 2 fichiers, afin d'avoir une idée du volume des données :

	Number_of_incidents	Number_of_pumps	Average_pump_per_incident
Year			
2018	105974	153675	1.5
2019	105010	154733	1.5
2020	98568	150384	1.5
2021	90328	130598	1.4
All	399880	589390	1.5

("Number_of_incidents" représente le nombre d'incidents contenus dans la table des incidents ; "Number_of_pumps" représente le nombre de véhicules mobilisés sur les incidents ; tout ceci nous permet par la même occasion de chiffrer un nombre moyen de véhicules par intervention, dans "Average_pump_per_incident").

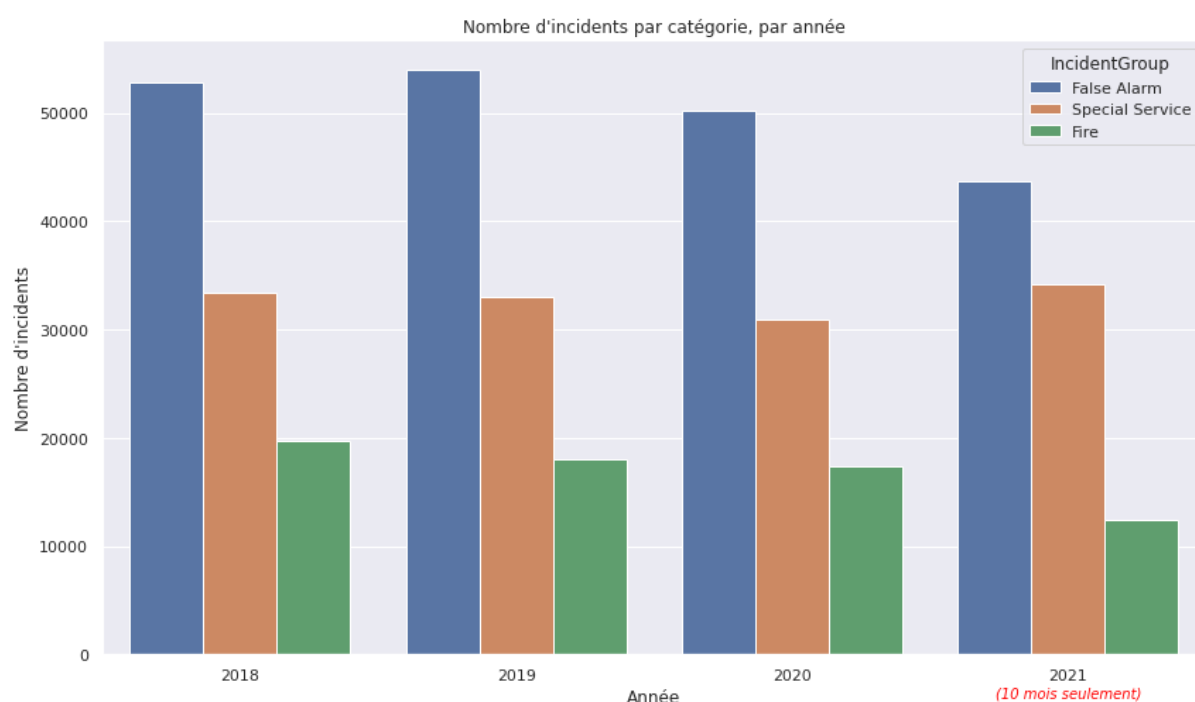
Nous avons mené une analyse précise des variables contenues dans chacune des 2 tables ; cette analyse est disponible en annexe (annexes 01 et 02).

2.2. La visualisation des données

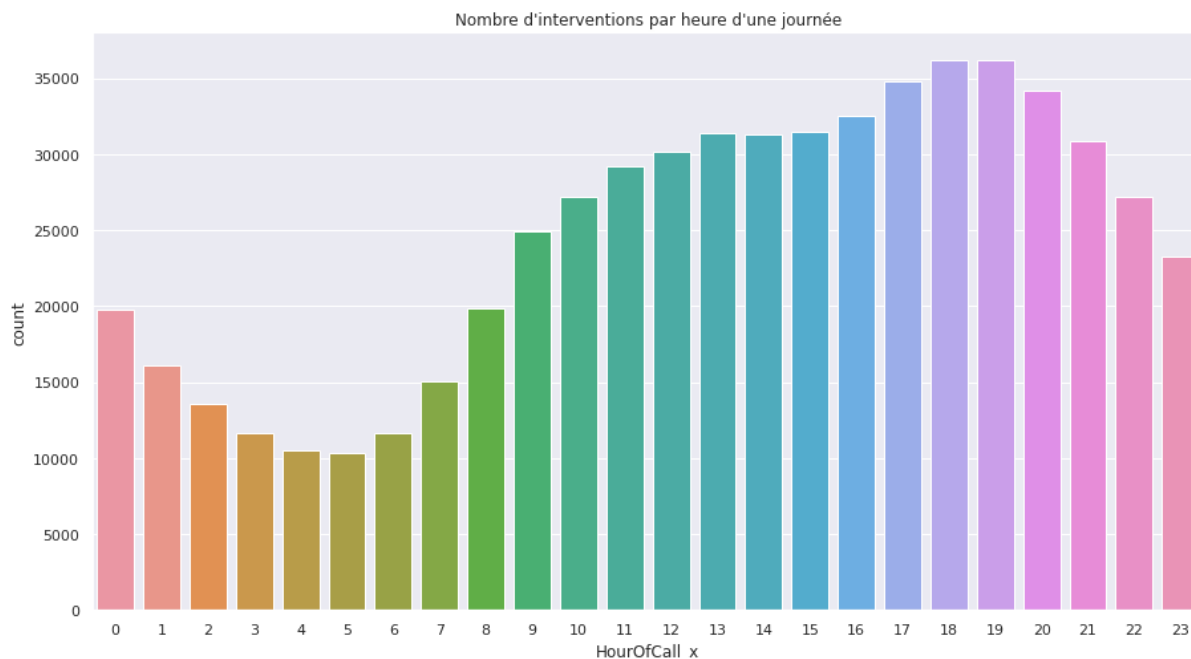
Suivant la sélection et la jointure des deux tables, un des premiers travaux a été de mieux comprendre les données. Pour cela, la visualisation des données et leur représentation graphique est la meilleure méthode.

Le notebook “2.2 Visualisation des données.ipynb” nous permet de représenter ici les graphiques les plus pertinents observés.

Nombre d’incident par année et par type d’incident

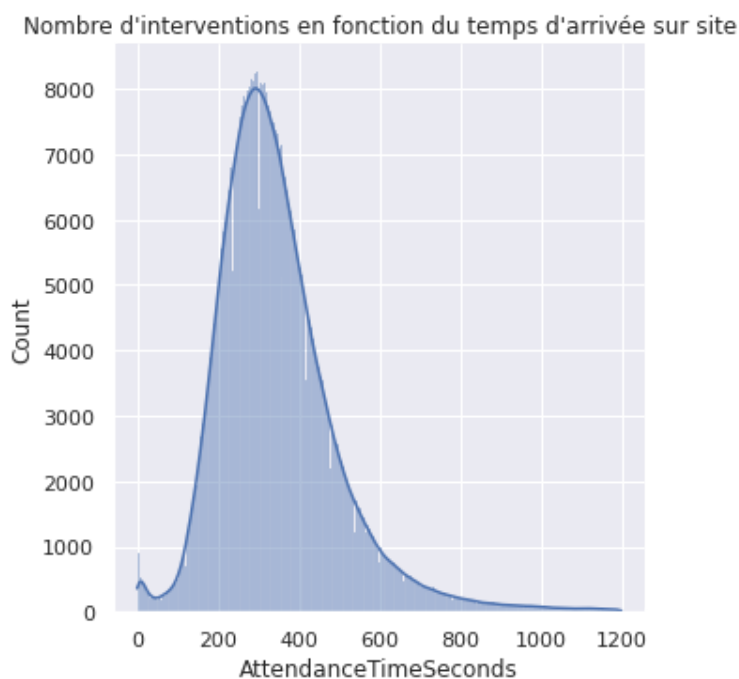


Nombre d'interventions par heure de la journée



Ces deux premiers graphiques nous permettent d'afficher des tendances et des variations du nombre d'incidents en fonction de certains facteurs, l'année, les heures de la journée et le type d'incident. On peut voir très clairement que le nombre d'interventions varie fortement suivant l'heure : moins d'interventions pendant la nuit, et beaucoup entre 17h - 20h, ce qui semble logique avec une activité humaine.

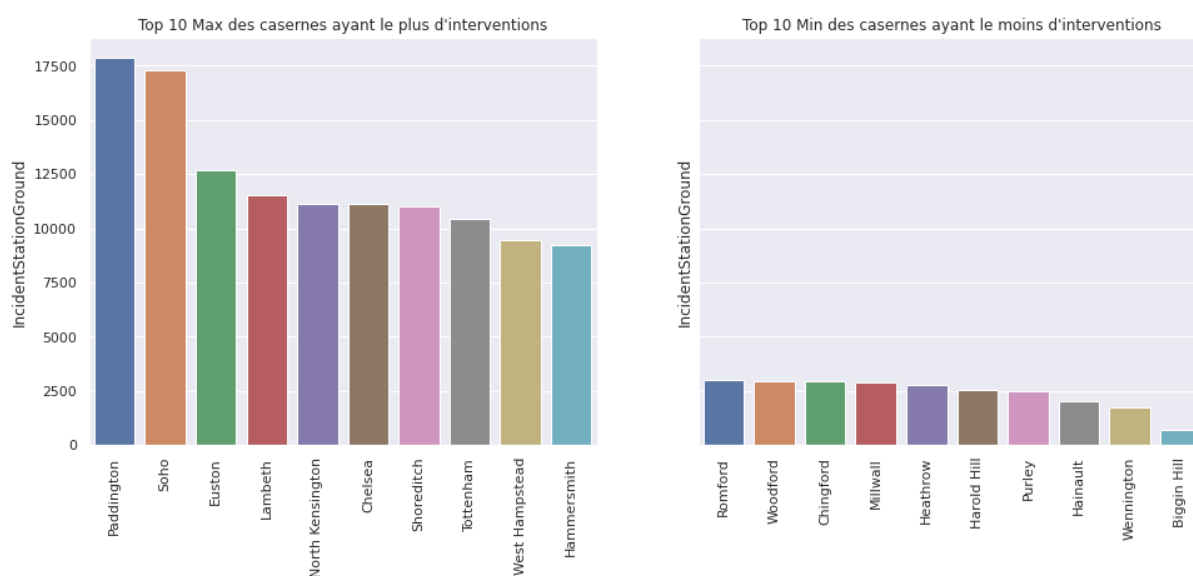
Nombre d'interventions en fonction du temps d'arrivée sur site



On remarque tout d'abord que la distribution de la courbe du nombre d'interventions en fonction du temps d'arrivée est une gaussienne, ce qui est une bonne chose car nombre d'algorithmes sous-tendent que la variable target à une distribution de type gaussienne. On constate aussi un pic aux alentours des 350s, proche de l'objectif fixé des 360s.

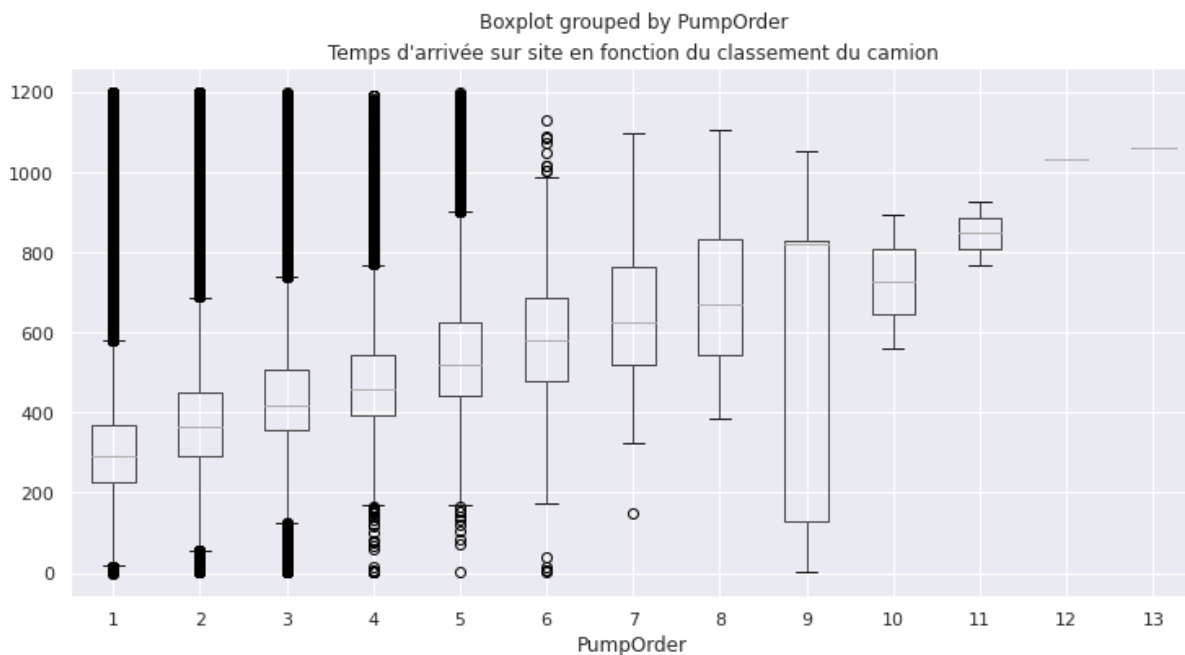
On remarque aussi la présence de valeurs extrêmes avec des temps d'arrivée sur site proche de 0s, et d'autres supérieurs à 1000 s.

Top 10 Min et Max des casernes ayant le plus et le moins d'incidents



Ces graphiques nous donnent le nombre d'interventions par caserne. On s'aperçoit que les casernes les plus importantes situés dans le city center de Londres (Soho par exemple) ont plus de 10x plus d'interventions que d'autres casernes probablement situés aux alentours de la couronne extérieure de Londres.

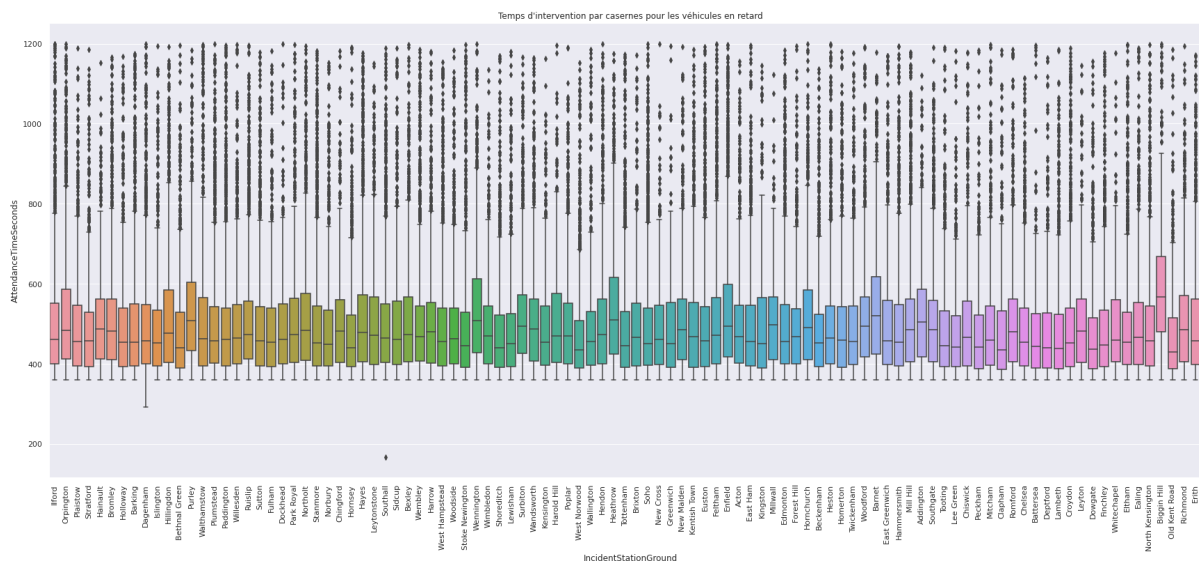
Temps d'arrivée sur site en fonction du classement du camion



Ce graphique nous permet de constater que le classement du camion correspond à son temps d'arrivée et non son départ. Comme attendu, plus le classement du camion est élevé, plus son temps d'arrivée est important.

Encore une fois, on constate aussi que certains camions, même classés 1, arrivent avec un temps supérieur à 1000 voir 1200s.

Temps d'intervention par caserne pour les véhicules hors délais



Ce graphique nous permet d'afficher clairement qu'un véhicule considéré comme hors délai selon les données signifie qu'il est arrivé au-delà de l'objectif fixé des 360s.

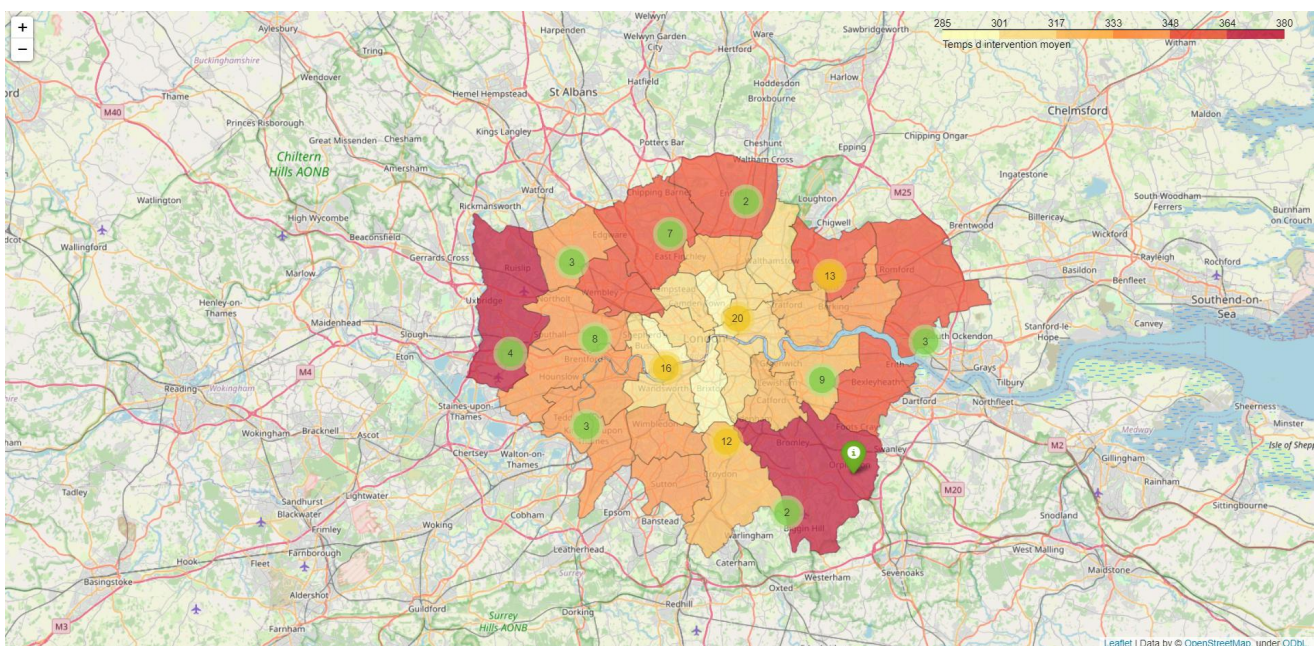
Temps d'intervention par quartier de Londres et répartition des différentes station de Londres

Avec ce [lien](https://ludomare.github.io/) (<https://ludomare.github.io/>) nous pouvons visualiser une carte dynamique des quartiers de Londres : les couleurs indiquant le temps moyen d'AttendanceTimeSeconds sur un quartier. De plus, chaque caserne/station est indiquée par un icône qui lui-même nous informe sur son propre temps moyen d'AttendanceTimeSeconds ainsi que son nombre d'interventions.

Cette carte nous confirme plusieurs éléments déjà énoncés, notamment :

- les casernes du centre sont plus sollicitées que celles en périphérie ;
- les quartiers du centre ont des temps d'intervention plus courts que ceux en périphérie.

On s'aperçoit aussi que la caserne Dartford est en dehors de Londres, et donc que les données la concernant sont à considérer plutôt comme des valeurs aberrantes.



Carte de Londres indiquant les stations et le temps moyen d'intervention (voir le lien)

Après ces premières analyses, nous avons procédé à des nettoyages des données, et à des créations de variables (c'est l'objet du notebook "2.3 - 2.5 data preprocessing.ipynb").

2.3. Le nettoyage des données

En explorant la base de données mergée, on observe que les coordonnées du lieu de chaque événement sont fournies par différentes colonnes dont notamment les colonnes Latitude et Longitude.

Cependant après vérification, nous avons remarqué un nombre important de NaN dans ces colonnes qui potentiellement éliminerait un nombre important de lignes.

De plus, en appliquant ces Latitudes/Longitudes sur un moteur géographique, on remarque que les points ne semblent pas se situer dans Londres.

Nous abandonnons donc ces coordonnées latitude/longitude initiales pour les colonnes Easting_rounded et Northing_rounded (OSGB36 National Grid) qui sont en fait des coordonnées géographiques cartésiennes.

En effet, sur le moteur géographique, les coordonnées révèlent des points qui, eux, se situent bien dans Londres. Easting fait référence à la distance mesurée vers l'est (ou la coordonnée x), tandis que Northing fait référence à la distance mesurée vers le nord (ou la coordonnée y) ainsi que Easting_rounded et Northing_rounded arrondi respectif de Easting et Northing.

Pour être exploitable par la suite (calcul de la distance), il nous faut créer des colonnes Latitude/Longitude à partir de Easting_rounded / Northing_rounded.

Le calcul est relativement simple mais long à appliquer sur l'ensemble du dataframe. Nous l'effectuons donc en amont, ce qui explique que le fichier que l'on récupère dès le début "**LFB Incident data Last 3 years + gps.csv**" intègre déjà cette transformation et les colonnes Latitude et Longitude fraîchement recalculées.

La colonne Ressource_Code est composée d'une lettre et de 3 chiffres. En se référant au site internet "https://fire.fandom.com/wiki/London_Fire_Brigade#Brigade_Profile", on se rend compte que le dernier chiffre de chaque Ressource_Code correspond à la typologie du véhicule d'intervention.

Appliance Glossary

Appliances are assigned radio call-signs based on their respective fire stations. For example, at Paddington Fire Station, call-sign "A-21", the Dual Pump Ladder would be assigned call-sign "A-211".

- 1 - Dual Pump Ladder
- 2 - Pump
- 3 - Turntable Ladder (32M)
- 4 - Turntable Ladder (64M)
- 5 - Aerial Ladder Platform
- 6 - Fire Rescue Unit
- A - Operational Support Unit

Ici dans notre base de données, seuls 2 types de véhicules sont notifiés : Dual Pump ladder et Pump ladder.

De fait, via une boucle, on créera une colonne *Appliance* dans laquelle on indiquera si c'est un Dual Pump ou un Pump, information qui pourrait être potentiellement utile dans notre futur modèle.

- Suppression de colonnes jugées inutiles

En menant l'analyse détaillée des variables constituant chacun des 2 fichiers du jeu de données, nous avons d'ores et déjà pu identifier celles qui ne présentaient aucun intérêt pour la suite de notre projet.

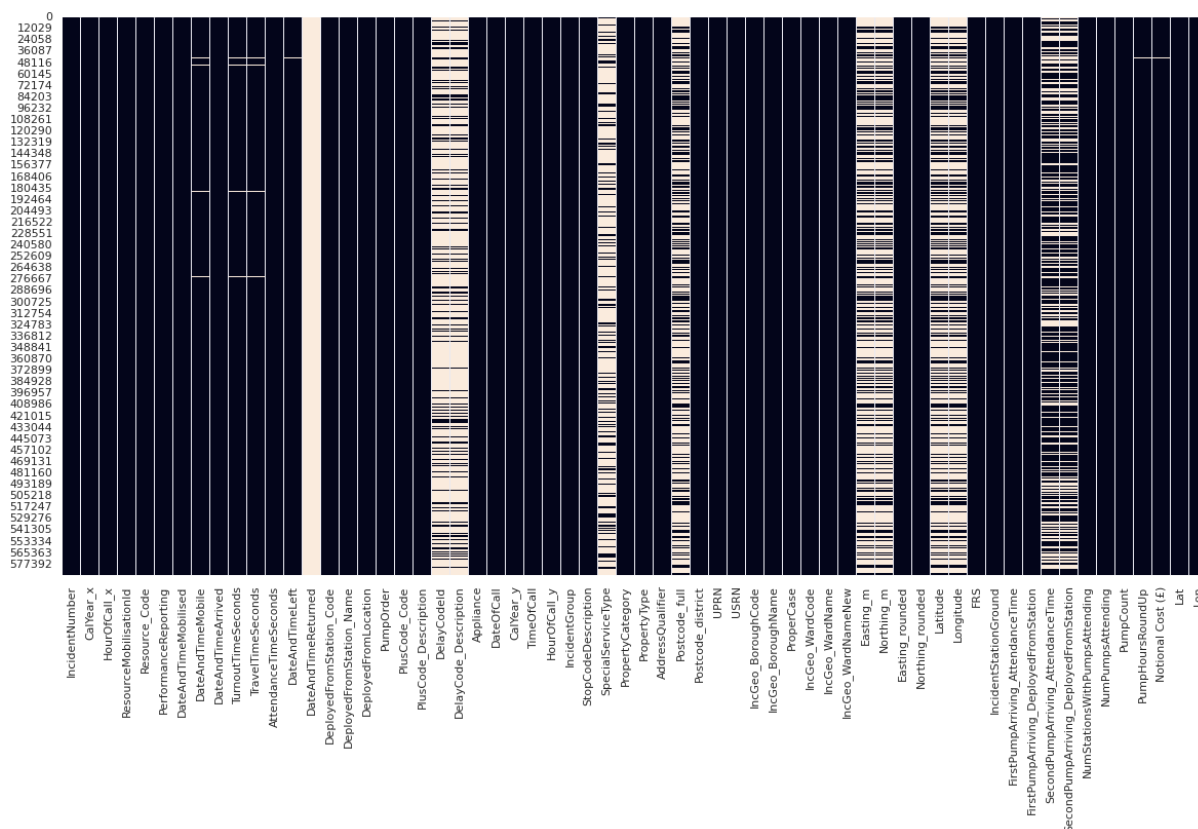
Au total, ce sont 29 variables que nous avons supprimées, selon les grandes thématiques suivantes :

- données postales ;
- données redondantes ;
- modalités ne présentant aucun intérêt ;
- données incomplètes que nous ne pouvons compléter ;
- données que nous avons déjà remplacées ;
- données hors périmètre de notre projet ;
- variable vide.

Pour plus d'explications sur ces choix, voir le tableau en annexe 03.

- Travaux sur les valeurs manquantes

Une visualisation graphique de notre base de données (format heatmap de Seaborn) nous a permis de constater à ce stade, que notre base de données n'était pas exempte de valeurs manquantes ("NaN", identifiés par les lignes claires sur le graphique ci-dessous).



Ce sont précisément 24 variables sur lesquelles il nous a fallu travailler.

Pour la plupart (principalement des variables liées à des dates et heures, ou des variables relatives aux casernes en lien avec l'incident), il s'agissait de variables que nous étions en mesure de reconstituer, en nous appuyant sur d'autres variables (parfois en émettant des hypothèses).

Très peu de ces valeurs manquantes nous ont amenées à des suppressions d'observations.

Pour davantage de détails sur les méthodes utilisées dans le traitement de ces valeurs manquantes, voir le tableau de synthèse en annexe 04.

Il ne nous reste alors que quelques valeurs manquantes, sur lesquelles nous reviendrons ultérieurement :

```
# Contrôle des Nan restant
df.isna().sum()[df.isna().sum() != 0]

DateAndTimeLeft      256
dtype: int64
```

- Formats de colonnes

Nous avons alors renommé et converti certaines variables :

- Lat et Lon ('object') ont été renommées en Latitude et Longitude, et converties au format 'float' ;
- DateOfCall ('object') a été convertie au format datetime ;
- 7 autres variables ont été converties au format integer ('int') :
TurnoutTimeSeconds, TravelTimeSeconds, CalYear_y, HourOfCall_y, NumStationsWithPumpsAttending, NumPumpsAttending, PumpCount.

- Cohérence générale et dernières valeurs manquantes

A cette étape, nous avons fait des contrôles de cohérences supplémentaires, et procédé à certaines corrections complémentaires qui pouvaient être nécessaires.

Nous avons d'abord vérifié la cohérence globale des heures enregistrées dans la base, pour vérifier que la chronologie des événements était correcte, à savoir :

DateAndTimeMobilised < DateAndTimeMobile < DateAndTimeArrived < DateAndTimeLeft
Aucune anomalie n'a été relevée.

Toujours sur les variables temporelles, nous avons vérifié que l'égalité ci-dessous était respectée :

AttendanceTimeSeconds = TurnoutTimeSeconds + TravelTimeSeconds
Aucune anomalie relevée.

Nous avons alors effectué des contrôles sur les variables qui font le décompte des véhicules qui sont intervenus sur les incidents : NumPumpsAttending (nombre de véhicules intervenant sur l'incident) et PumpCount (décompte de véhicules, sans plus de précisions). Nous avons croisé ces informations, par incident, avec le nombre de valeurs de la variable PumpOrder (variable qui est un classement de l'ordre d'arrivée de chacun des véhicules sur l'incident).

Des incohérences sont apparues entre ces 3 variables, pour 17600 incidents, et nous avons constaté la présence de valeurs semblant aberrantes dans la variable PumpCount (voir ci-dessous, des incidents avec des nombres de véhicules allant jusqu'à 250) :

	max_pumpsattending	max_pumpcount	nb_pumporder
IncidentNumber			
055211-05052018	2	1	1
141749-05112020	2	1	1
104919-26072018	2	1	1
010604-26012019	2	1	1
080647-23062018	2	1	1
...
063499-23052019	2	179	2
080303-23062018	7	197	5
018108-12022018	4	198	3
096730-15072018	1	209	1
086232-05072019	1	250	1

17600 rows × 3 columns

Nous avons alors décidé de recalculer la variable NumPumpsAttending en procédant au comptage des PumpOrder de chaque incident, sur l'intégralité de la base (sans se soucier de savoir si cette valeur était à l'origine correcte ou erronée).

Ne maîtrisant pas le contenu de la variable PumpCount, qui nous semble faire double-emploi avec NumPumpsAttending, nous avons décidé de la supprimer.

En lien avec les variables précédentes, nous avons également contrôlé et corrigé la variable NumStationsWithPumpsAttending, décomptant le nombre de casernes qui ont envoyé des véhicules sur les lieux de l'incident.

Un peu plus de 13000 lignes présentaient des anomalies, nous nous sommes alors appuyés sur Resource_Code, dont les 3 premiers caractères identifient la caserne d'affectation du véhicule, afin de corriger ces anomalies.

	IncidentNumber	NumStationsWithPumpsAttending	num_stations_attending_calc
33	000067-01012018	2	1
238625	098637-27072019	2	1
238546	098541-27072019	2	1
238474	098448-26072019	2	1
238473	098447-26072019	2	1
...
508182	053666-07052021	14	13
508183	053666-07052021	14	13
508184	053666-07052021	14	13
508157	053666-07052021	14	13
508185	053666-07052021	14	13

13382 rows × 3 columns

Une autre variable sur laquelle nous avons travaillé est DateAndTimeLeft (heure à laquelle le véhicule a terminé sa mission et a quitté les lieux de l'incident).

Il restait 256 valeurs manquantes (voir précédemment), que nous avons volontairement décidé de solutionner ultérieurement, parce que leur traitement nécessitait d'avoir fiabilisé la variable NumPumpsAttending (ce qui a été réalisé à l'étape précédente).

Pour renseigner ces valeurs manquantes, nous avons procédé en 2 étapes :

- nous avons d'abord calculé une durée moyenne du temps de présence des véhicules, qui sont intervenus sur des incidents similaires. Pour définir la similarité à ce stade de nos travaux, nous avons choisi les variables IncidentGroup et SpecialServiceType (pour caractériser le type d'incident), ainsi que NumPumpsAttending (pour en caractériser la gravité).
- puis nous avons ajouté cette durée calculée à la variable DateAndTimeArrived, pour nous donner une estimation de DateAndTimeLeft.

Lors de l'exploration des données, nous avons constaté qu'une caserne hors périmètre des pompiers de Londres figurait dans la base de données, parce qu'elle avait envoyé un véhicule en renfort sur un incident géré par la brigade de Londres.

Nous avons choisi de supprimer l'incident en question (1 incident composé de 3 véhicules).

A ce stade, nous avons décidé de faire un contrôle de cohérence sur les différents temps calculés, en vérifiant les égalités suivantes :

AttendanceTimeSeconds	=	DateAndTimeArrived	-	DateAndTimeMobilised
TurnoutTimeSeconds	=	DateAndTimeMobile	-	DateAndTimeMobilised
TravelTimeSeconds	=	DateAndTimeArrived	-	DateAndTimeMobile

Aucune anomalie relevée.

Pour en terminer avec les contrôles de vraisemblance, nous avons identifié lors des divers travaux précédents, que les dates/heures de réception des appels par les pompiers (DateOfCall), et les dates/heures de mobilisation des véhicules (DateAndTimeMobilised), n'étaient parfois pas cohérentes.

En affichant la liste des véhicules pour lesquels DateOfCall est strictement supérieure à DateAndTimeMobilised, nous avons eu confirmation de ces anomalies... c'est une liste de plus de 370 000 véhicules qui semblent avoir été mobilisés avant même que l'appel n'ait été reçu !

Erreurs de saisies ? Temps de traitements dans les mises à jour d'informations ? Difficile à dire, les écarts étant très variables, parfois de l'ordre de quelques secondes, parfois précisément d'une heure...

Cette analyse, si elle ne nous a conduit à aucune correction des données, nous a amené à nous interroger sur la variable que nous devons prendre comme référence temporelle pour chaque incident : comme la variable DateOfCall n'est finalement la référence pour aucune des durées calculées dans cette base de données, il nous a paru naturel de définir que DateAndTimeMobilised deviendrait notre référence horaire (date et heure) pour situer l'incident dans le temps.

En conséquence, toutes les variables faisant référence à DateOfCall pourront être supprimées de notre base de données.

2.4. La création de variables basées sur les variables existantes

- Vague / rang de l'appel à mobilisation

Nous avons constaté que les véhicules étaient appelés à se mobiliser “par vague”. Afin de mieux comprendre, voici un exemple, dans lequel nous comptons le nombre de lignes pour l’incident “038964-29032018” (qui compte 9 véhicules) :

IncidentNumber		Chronologie	
DateAndTimeMobilised			
2018-03-29 17:14:39	5	5	véhicules mobilisés à 17:14:39
2018-03-29 17:16:17	1	1	véhicule mobilisé à 17:16:17
2018-03-29 17:16:21	1	1	véhicule mobilisé à 17:16:21
2018-03-29 17:18:28	2	2	véhicules mobilisés à 17:18:28

Il nous a paru intéressant pour la suite de nos travaux, de créer une variable contenant cette information, pour savoir à quelle vague d’appel faisait partie chacun des véhicules.

Nous avons choisi d’appeler cette variable Mobilised_Rank.

Dans un tableau intermédiaire, nous avons pour chaque incident, fait le compte du nombre de modalités de DateAndTimeMobilised, puis au moyen d’une boucle, nous avons calculé le numéro d’ordre de la vague de mobilisation.

Ci-dessous pour l’exemple, les données calculées pour le même incident que précédemment :

	Nb_pumps_asked	Mobilised_Rank
DateAndTimeMobilised		
2018-03-29 17:14:39	5	1
2018-03-29 17:16:17	1	2
2018-03-29 17:16:21	1	3
2018-03-29 17:18:28	2	4

Puis nous avons reporté cette variable dans la table principale, sur chacun des véhicules mobilisés (ci-dessous pour l'incident "038964-29032018") :

	IncidentNumber	DateAndTimeMobilised	PumpOrder	NumPumpsAttending	Mobilised_Rank
34824	038964-29032018	2018-03-29 17:14:39	9	9	1
34825	038964-29032018	2018-03-29 17:14:39	3	9	1
34826	038964-29032018	2018-03-29 17:14:39	4	9	1
34827	038964-29032018	2018-03-29 17:14:39	5	9	1
34828	038964-29032018	2018-03-29 17:14:39	6	9	1
34829	038964-29032018	2018-03-29 17:16:17	2	9	2
34830	038964-29032018	2018-03-29 17:16:21	1	9	3
34831	038964-29032018	2018-03-29 17:18:28	7	9	4
34832	038964-29032018	2018-03-29 17:18:28	8	9	4

Une intuition que l'on peut avoir est de se dire que le véhicule arrivant en premier sur les lieux de l'incident devrait logiquement faire partie de la première vague de mobilisation. Nous avons donc croisé ces informations pour le vérifier.

Ci-dessous, 2 synthèses de la répartition des véhicules (en nombre de véhicules, puis en proportion selon la vague d'appel) :

PumpOrder	1	2	3	4	5	6	7	8	9	10	11	12	13
Mobilised_Rank													
1	374207	147248	28246	14294	10875	147	16	6	1	0	0	0	0
2	1578	4718	2635	1395	755	145	76	44	1	0	0	0	0
3	53	87	193	160	51	23	12	4	2	1	1	1	1
4	4	9	8	12	13	3	5	2	1	1	1	0	0
5	0	1	1	3	2	1	0	2	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0

PumpOrder	1	2	3	4	5	6	7	8	9	10	11	12	13
Mobilised_Rank													
1	1.0	0.97	0.91	0.90	0.93	0.46	0.15	0.10	0.2	0.0	0.0	0.0	0.0
2	0.0	0.03	0.08	0.09	0.06	0.45	0.70	0.76	0.2	0.0	0.0	0.0	0.0
3	0.0	0.00	0.01	0.01	0.00	0.07	0.11	0.07	0.4	0.5	0.5	1.0	1.0
4	0.0	0.00	0.00	0.00	0.00	0.01	0.05	0.03	0.2	0.5	0.5	0.0	0.0
5	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.0	0.0	0.0	0.0	0.0
6	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0	0.0	0.0	0.0

Il est flagrant que les véhicules arrivant en premier ou second sur l'incident (PumpOrder = 1 ou 2), sont issus pour la quasi-totalité (100% et 97%) de la première vague des appels à mobilisation.

Autre constat, le nombre de véhicules appelés postérieurement à la première vague est finalement peu élevé.

Sur la base de cette analyse, nous avons alors choisi de limiter notre jeu de données aux véhicules issus de la seule première vague d'appels.

- **Calcul de la distance (vol d'oiseau)**

Intuitivement, il nous paraît logique que le temps d'intervention soit fortement corrélé à la distance entre le camion et le lieu de l'événement. Or cette donnée n'existe pas dans les 2 bases de données originelles. Pour la créer, il nous faut donc 2 informations à savoir les coordonnées géographiques du camion ainsi que les coordonnées géographiques des lieux d'événements. Nous choisissons un calcul dit à "vol d'oiseau" c'est-à-dire une distance euclidienne.

Concernant les coordonnées géographiques du camion, on remarque assez rapidement que 96% des camions partent de leurs Home Station, caserne de rattachement. Nous considérons pour simplifier les calculs que les coordonnées d'origines des camions sont toujours celles de leurs Home Station respectifs. Il nous suffit donc pour cela de récupérer sur Internet la latitude/la longitude de chacune des stations de Londres et de les insérer dans le dataframe via les colonnes Station_Latitude et Station_Longitude.

Concernant les coordonnées géographiques des lieux d'événements, comme indiqué au début de ce rapport, nous nous baserons sur les latitudes/longitudes calculées sur la base de Easting/Northing (voir plus haut).

Pour résumer nous obtenons les latitudes et longitudes des camions ainsi que du lieu de l'événement, reste à ce moment à calculer la distance euclidienne, calcul développé dans le notebook.

Nous créons ainsi la colonne Distance, et supprimons, une fois la distance calculée, l'ensemble des colonnes portant une indication géographique.

- **IncidentGroup / StopCodeDescription / SpecialServiceType**

Lors de nos recherches bibliographiques (notamment *fire-statistics-definitions*), nous avons constaté que les incidents pouvaient facilement être classifiés en plusieurs catégories, plus précisément basées sur la variable IncidentGroup et reprenant les données des variables StopCodeDescription et SpecialServiceType. Nous avons donc créé 3 variables plus pertinentes que celles des jeux de données initiales : IncidentType, IncidentCategorie et FalseAlarme.

Une première hypothèse a notamment été de supprimer l'information False Alarme de la variable IncidentType et de la remplacer par l'information Fire. Afin de ne pas perdre cette information, nous avons créé une variable FalseAlarm. Cette variable ne sera pas utilisée dans ce projet mais pourrait faire l'objet d'une étude dédiée. Nous pourrions imaginer une intelligence artificielle permettant de détecter les fausses alarmes (ou supposée) et ainsi envoyer un nombre limité de moyens.

Les types d'incident repris dans IncidentType sont :

Fire : primary fire, secondary fire, chimney fire, AFA.

Major Environmental Disasters : flooding, hazardous material incidents or spills and leaks.

Domestic Incidents : persons locked in/out, lift releases, suicide/attempts.

Local Emergencies : road traffic incidents, responding to medical emergencies, rescue of persons and/or animals or making areas safe.

Prior arrangement : attend or assist other agencies, which may include some provision of advice or standing by to tackle emergency situations.

La variable IncidentCategory, regroupe, les informations des variables StopCodeDescription et SpecialServiceType, et est une sous catégorie de IncidentType

- **TotalTotalOfPumpInLondon Out et PumpAvailable**

Afin d'avoir une information sur le nombre de camions en intervention et connaître leur disponibilité, nous avons pris le parti de créer deux nouvelles variables :

- TotalOfPumpInLondon_Out : Nombre total de camions en intervention sur Londres au moment de l'appel.
- PumpAvailable : Nombre de camions disponible dans la caserne du lieu de l'incident.

Ces deux variables ont été créées à partir de boucles sur les 300 camions précédemment sorties en interventions au moment de l'incident.

- **Derniers ajustements et contrôles**

A cette dernière étape, nous avons principalement mis en oeuvre les décisions prises précédemment, à savoir :

- suppression des incidents des 300 premières lignes de notre base (dorénavant incomplètes suite au chiffrage du nombre de camions, via les calculs sur des boucles de 300 valeurs glissantes) ;
- suppression des observations des camions mobilisés au delà de la première vague d'appels ;
- création de 5 variables temporelles de référence (année, mois, jour, jour de la semaine et heure), basées sur la date de mobilisation des véhicules, venant remplacer les 6 variables liées aux dates et heures d'appel ;
- et enfin, suppression de variables temporelles complètement inutiles pour la suite (DateAndTimeMobile, DateAndTimeArrived et DateAndTimeLeft).

Puis, nous avons réalisé un dernier contrôle de cohérence, comparant distances (même s'il s'agit d'une distance à vol d'oiseau), et temps de trajets (TravelTimeSeconds).

En calculant une vitesse moyenne en km/h, nous nous sommes rendus compte que si la grande majorité des valeurs étaient plausibles, des valeurs aberrantes subsistaient.

Après analyse, et toujours dans une idée de cohérence de la base de données, nous avons opté pour la suppression des incidents sur lesquels au moins une observation (un véhicule) faisait apparaître une vitesse constatée supérieure à 60 km/h, seuil qui nous paraît adapté, pour un environnement urbain (environ 16500 véhicules étaient concernés, conduisant à la suppression, une fois appliqué le raisonnement par incident, d'environ 27000 observations).

2.5. Intégration de variables externes

Lors des travaux sur les différents modèles de prédictions, nous avons été amenés à compléter les variables dont nous disposons, par des variables externes dont nous pouvons supposer qu'elles peuvent avoir des incidences sur les temps de réponses.

- **Variables des données météorologiques**

Les premières données externes intégrées à notre dataframe sont celles liées aux conditions météorologiques, pouvant notamment impacter les conditions de circulations ou le type d'incident (exemple : inondation).

Ces données réelles, collectées au pas de temps horaire sont issues du site internet : <https://www.visualcrossing.com/weather/weather-data-services>.

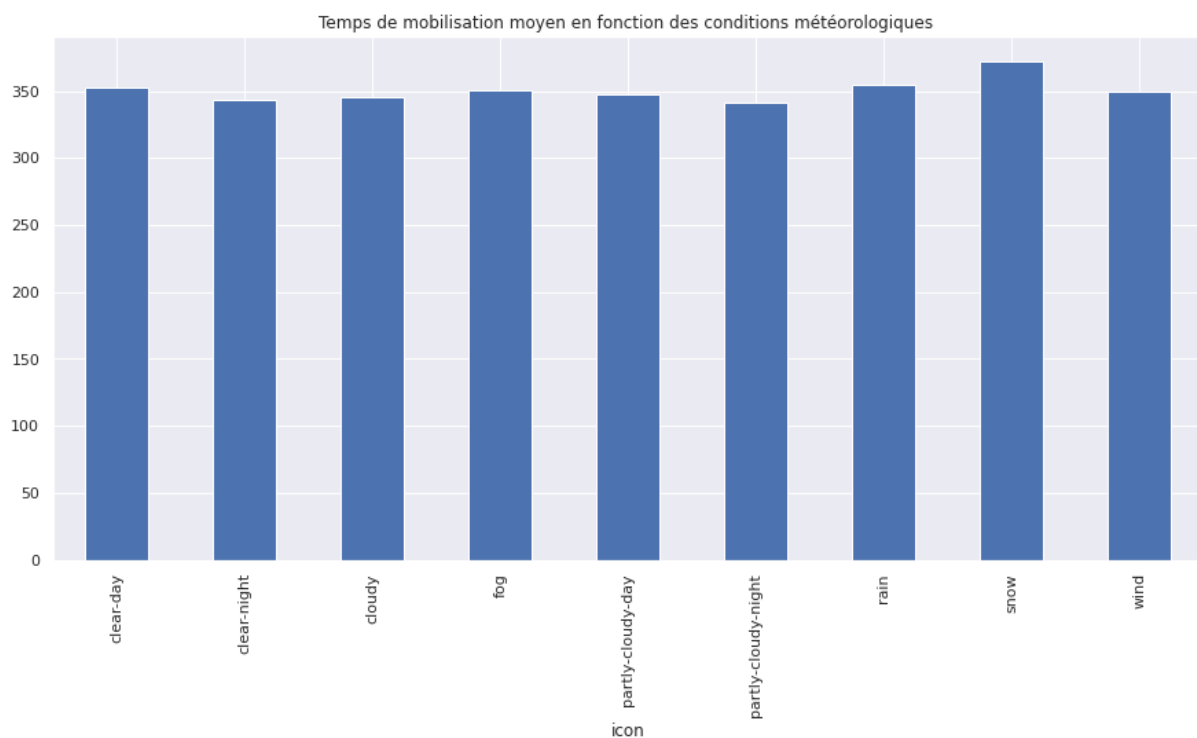
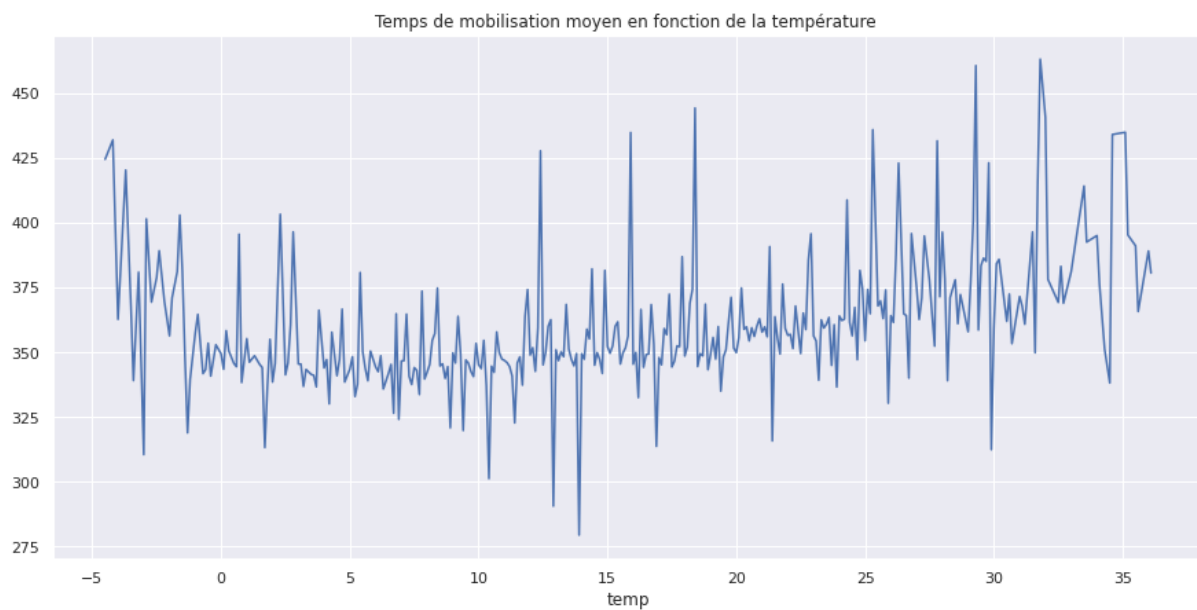
Nous avons donc pu joindre les données à notre dataframe en fonction de l'heure de l'appel de l'incident.

Les variables prises en compte sont :

- temp : la température
- precip : la pluviométrie
- cloudcover : La couverture nuageuse
- visibility : la visibilité
- conditions et icon : la météo (dégagé, nuageux, pluie, vent etc...)

Nous pouvons voir notamment ci-dessous l'impact de la température ou de la météo sur le temps d'intervention.

On s'aperçoit que lors des hautes et basses températures, le temps d'arrivée sur site moyen est plus long que pour les températures modérées. Idem pour période de neige où les délais sont en moyenne un peu plus longs.



- **Variables des données jours fériés et congés**

Il conviendra de distinguer les jours dits de congés des jours fériés.

En effet, concernant les jours fériés, une librairie, `workalendar`, permet facilement d'obtenir, pour un certain nombre de pays, les jours fériés quelque soit l'année. Il suffit du coup de faire appel à une fonction (`is_working_day`) pour savoir si le jour était férié ou non et de réaliser cette opération sur chacune des lignes.

Concernant les jours de congés, nous sommes basées sur les congés scolaires. Nous sommes du coup aidées de documents produits par la ville de Londres. Notre choix s'est porté arbitrairement sur l'arrondissement de Barking and Dagenham (arrondissement du grand Londres) car celui-ci fournissait directement les semaines de vacances scolaires, voir en exemple l'annexe 05. A partir de cette source, nous avons constitué un document excel qui conservait ces dates. Nous avons ensuite injecté ces données dans nos features.

Nous avons donc, à l'issue de ce travail, créé 2 colonnes binaires (0 ou 1), `working_day` correspondant aux jours fériés ou non, ainsi que `school_holidays` correspondant aux vacances scolaires ou non.

- **Variable de densité du trafic dans les rues de Londres**

Sur le site "tomtom.com", nous avons trouvé une page dédiée à l'analyse du trafic londonien (https://www.tomtom.com/en_gb/traffic-index/london-traffic/). Et plus précisément, un état de synthèse sur les 3 dernières années (2019, 2020, 2021), présentant pour chaque année, une synthèse des taux de congestion par heure, pour chacun des jours de la semaine ("weekly traffic congestion by time of day", dont un exemple de l'année 2021 figure en annexe 06).

En procédant par copier / coller dans un tableur, nous avons récupéré ces taux, pour en faire un fichier texte à exploiter dans nos travaux (n'ayant pas cette information pour l'année 2018, nous avons fait le choix de reporter sur cette année les chiffres de 2019).

Par fusion avec notre base de données, nous avons alors intégré cette information dans une variable nommée `congestion_rate`.

A l'issue de ces travaux de nettoyages, corrections et ajouts de variables, notre base de données se compose de :

- **43 variables ;**
- **pour 547865 observations.**

Le résumé figure sur la page suivante.

Une description plus détaillée des variables, regroupées selon la nature des informations qu'elles apportent, est disponible en annexe 07.

DataFrame à l'issue des travaux de nettoyage, corrections et ajouts de variables :

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547865 entries, 0 to 547864
Data columns (total 43 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   IncidentNumber                        547865 non-null object
1   Resource_Code                        547865 non-null object
2   DateAndTimeMobilised                 547865 non-null datetime64[ns]
3   TurnoutTimeSeconds                   547865 non-null int64
4   TravelTimeSeconds                    547865 non-null int64
5   AttendanceTimeSeconds                 547865 non-null int64
6   DeployedFromStation_Code             547865 non-null object
7   DeployedFromStation_Name             547865 non-null object
8   DeployedFromLocation                 547865 non-null object
9   PumpOrder                            547865 non-null int64
10  DelayCode_Description                 547865 non-null object
11  Appliance                            547865 non-null object
12  PropertyCategory                     547865 non-null object
13  PropertyType                         547865 non-null object
14  AddressQualifier                     547865 non-null object
15  IncidentStationGround                 547865 non-null object
16  NumStationsWithPumpsAttending         547865 non-null int64
17  NumPumpsAttending                     547865 non-null int64
18  Mobilised_Rank                       547865 non-null int64
19  IncidentType                         547865 non-null object
20  IncidentCategory                     547865 non-null object
21  FalseAlarm                           547865 non-null int64
22  Distance                             547865 non-null float64
23  TotalOfPumpInLondon_Out              547865 non-null int64
24  PumpByStation                        547865 non-null int64
25  Station_Code_of_ressource            547865 non-null object
26  IncidentStationGround_Code           547865 non-null object
27  PumpOfIncidentStation_Out            547865 non-null int64
28  PumpAvailable                        547865 non-null int64
29  year                                 547865 non-null int64
30  month                                547865 non-null int64
31  day                                  547865 non-null int64
32  weekday                              547865 non-null object
33  hour                                 547865 non-null int64
34  temp                                 547865 non-null float64
35  precip                               547865 non-null float64
36  cloudcover                           547865 non-null float64
37  visibility                           547865 non-null float64
38  conditions                           547865 non-null object
39  icon                                 547865 non-null object
40  workingday                           547865 non-null object
41  school_holidays                      547865 non-null object
42  congestion_rate                       547865 non-null float64
dtypes: datetime64[ns](1), float64(6), int64(16), object(20)
memory usage: 183.9+ MB
time: 469 ms (started: 2022-05-14 18:27:13 +00:00)
```

3. Modèles de prédiction

3.1. Prédiction du nombre de camions à envoyer sur l'intervention

Les notebooks exploités dans ce chapitre sont :

- "3.1 Prédiction du nombre de camion à envoyer sur l'intervention Workbook Optuna.ipynb" : notebook de travail ;
- "3.1 Prédiction du nombre de camion à envoyer sur l'intervention WorkBook.ipynb" : notebook de travail ;
- "3.1 Prédiction du nombre de camion à envoyer sur l'intervention.ipynb" : modèle retenu.

3.1.1 Travaux préparatoires

Notre premier modèle sera la prédiction du nombre de véhicules d'urgence à envoyer sur l'incident. Ce modèle pourrait devenir un outil d'aide à la décision pour l'opérateur, permettant de gagner quelques secondes précieuses quant aux temps d'arrivée sur les lieux de l'incident.

Pour ce modèle, nous utiliserons la base de données retravaillée : *"base_ml.pkl"*.

Cette base de données étant utilisée pour les 3 prédictions détaillées dans ce rapport, un nouveau travail de traitement des données est nécessaire.

Les modèles présentés dans ce paragraphe seront de type classification.

Pour le prétraitement des données, nous avons dans un premier temps réduit la dimension du dataframe en ne conservant qu'une ligne par incident.

Cette première étape nous permet de connaître précisément le nombre d'interventions sur la période étudiée, soit 359 880 incidents.

```
# On ne conserve qu'une seule ligne par Incident
df = df.drop_duplicates(subset=['IncidentNumber'])
df.shape
```

```
(359880, 43)
```

Nous avons aussi pris la décision de supprimer l'ensemble des interventions catégorisées "Fausse Alarme". Cette information étant inconnue au moment de l'appel, et le dataframe étant suffisamment important, ce choix nous permet d'éviter de définir un type d'incident, `IncidentType` pouvant impacter le résultat de l'algorithme. Cette décision, réduit le dataframe à 307330 incidents.

Ensuite, nous avons sélectionné les features, variables nécessaires et cohérentes à la prédiction et la target, la variable cible.

Target : `NumPumpsAttending`

Features : `PropertyCategory`, `PropertyType`, `AddressQualifier`, `IncidentType`, `IncidentCategory`

Le nombre de features peut paraître faible au regard du nombre total de variables, Nous avons fait le choix de ne sélectionner que les variables permettant de choisir le nombre de camion à envoyer, soit une information sur l'adresse, le type de lieu, le type d'incident.

Concernant les autres features, la plupart sont inconnues au moment de l'appel (informations sur les camions, information sur les délais d'information). Les informations sur la météo et le calendrier (heure, jour, mois, année) ne nous semblent pas pertinentes.

Les features choisies sont de type 'catégorielle' nous appliqueront la fonction de dichotomisation `get_dummies` afin de les transformer en variables indicatrices.

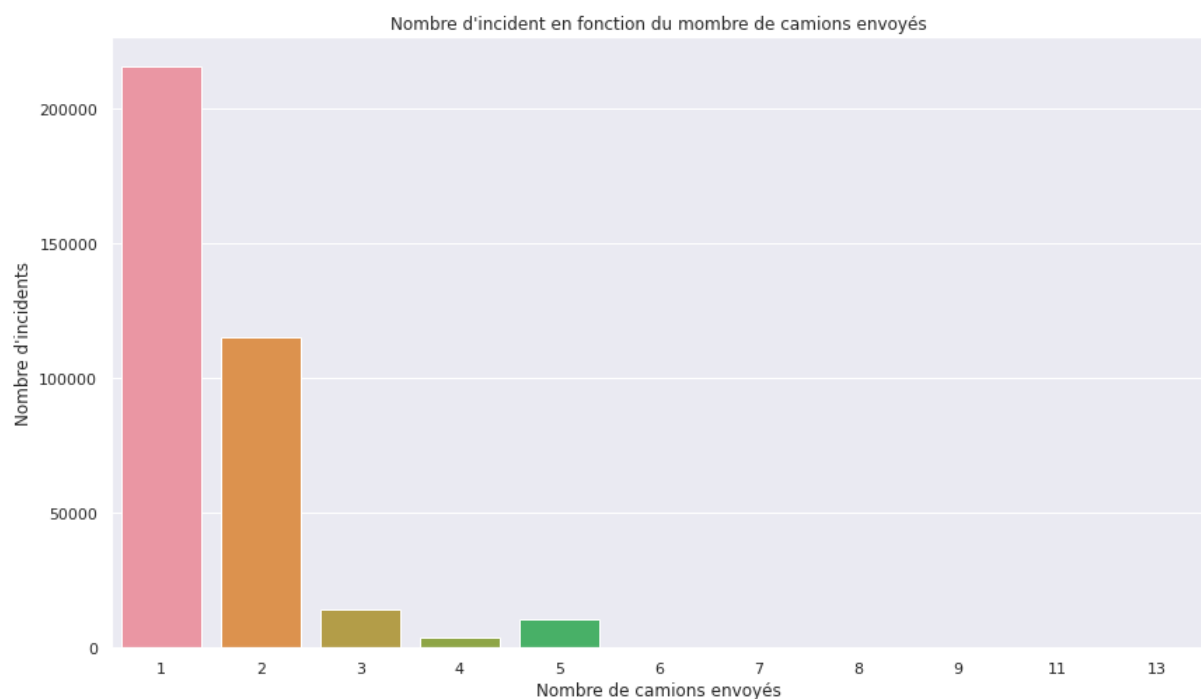
Ces transformations engendrent la création d'un très grand nombre de variables pour arriver à 335 colonnes.

Avant la dichotomisation, nous avons représenté la corrélation entre les différentes variables :



La dernière décision à prendre dans cette étape de prétraitement se porte sur la “target”. Au vue de la répartition déséquilibrée, nous avons décidé de réduire la classification à 5 classes, de 1 à 5 camions. Pour les incidents ayant envoyé 6 camions et plus, nous les conservons et leur appliquons la classe “5 camions”.

Cette classification nous permet d’avoir une représentation la plus réaliste possible.



A noter ici que la réduction du nombre de variables et le choix de réduire la target à ("5 classes") a créé de nombreux doublons, des incidents ayant les mêmes informations sur les variables choisies. Nous faisons le choix de ne pas les supprimer, ce qui réduirait considérablement le nombre d'incidents étudiés et nous ferait perdre l'information de la fréquence d'un incident du même type qui se répèterait.

Après cette première étape de prétraitement, et avant le lancement des modèles de prédictions, il nous faut diviser les matrices en un ensemble d'entraînement et un ensemble de tests. Nous avons choisis respectivement pour chacun des deux ensembles, 80-20%.

Nous avons souhaité travailler sur les mêmes ensemble "train" et "test", nous n'avons donc pas utilisé ici la fonction `train_test_split`. Nous avons défini dès le début du processus, les incidents qui feraient partie de l'un ou l'autre des deux ensembles.

Dans cette partie, avec le travail de prétraitement, nous arrivons à la répartition suivante :

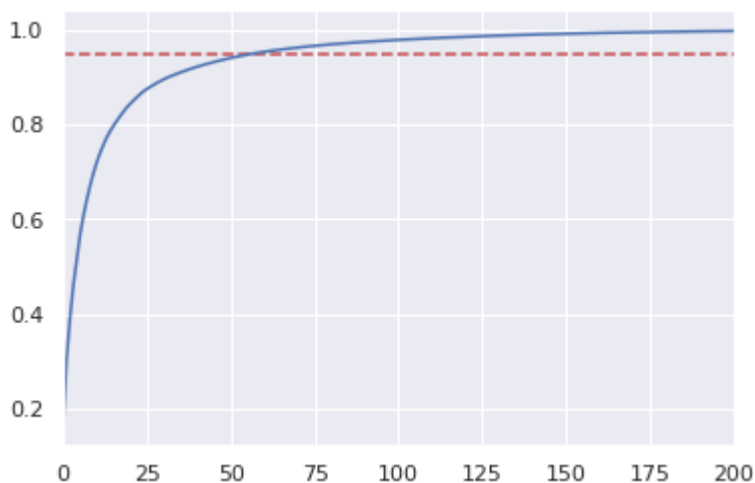
```
Jeux de données créés :  
Dimensions (X_train, y_train) : ((245973, 334), (245973,))  
Dimensions (X_test, y_test)   : ((61357, 334), (61357,))  
Le jeu de données test représente 19.964533237887615 % des incidents
```

Au vue du très grand nombre de colonnes pour le modèle, nous avons étudié la variance des variables.

En appliquant la fonction PCA, nous pourrions réduire assez considérablement le nombre de colonnes (ci-dessous).

Nombre de composantes retenues pour une variance à 95% : 59

Nombre de composantes retenues pour une variance à 99% : 144



Nous n'appliquerons pas cette réduction pour notre modèle car le gain de temps n'est pas significatif.

3.1.2 La métrique choisie

Notre objectif étant de déterminer le nombre de camions à envoyer sur l'intervention, de 1 à 5, nous déterminerons le meilleur modèle à l'aide de la métrique "Accuracy".

L'Accuracy est simplement calculé par la fraction du nombre correct de prédiction sur le nombre total de prédiction.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

3.1.3 Les modélisations

Lors de la phase modélisation, nous avons tout d'abord exécuté un panel assez large des modèles de classification sur notre jeu de données. L'objectif étant d'avoir une première vision assez précise de chacun des modèles.

Nous avons pris le parti de ne pas tester le modèle SVM au vu de la taille de nos jeux de données.

Ci-dessous les différents résultats récoltés :

	Accuracy	
	Train	Test
Decision Tree Classifier, criterion : Entropy	0.816	0.813
Decision Tree Classifier, criterion : Gini	0.694	0.689
Logistic Regression	0.805	0.805
SGD Classifier	0.805	0.806
KNeighborsClassifier	0.809	0.807
XGBoost	0.809	0.809
Random Forest Classifier, criterion : Gini	0.816	0.812
Random Forest Classifier, criterion : Entropy	0.816	0.812
BalancedRandomForestClassifier	0.689	0.688

Les résultats sont étonnamment assez similaires avec un accuracy aux alentours de 0.80, 0.81 exceptés les modèles, Decision Tree Classifier, criterion : Gini et BalancedRandomForestClassifier qui n'atteignent pas les 0.70.

Afin d'affiner ces résultats nous avons ensuite procédé à des modélisations, en utilisant la bibliothèque Optuna, qui est un cadre d'auto-optimisation d'hyperparamètres. Il est principalement utilisé pour le réglage des hyper paramètres de l'apprentissage automatique, donc particulièrement adapté à notre modèle.

Les résultats récoltés sont présentés dans le tableau ci-dessous :

	Accuracy
	Test
Logistic Regression	0.771
SGDClassifier	0.773
XGBClassifier	0.813
KNeighborsClassifier	0.811
RandomForestClassifier	0.709

Les deux modèles les plus performants sont XGBClassifier et KNeighborsClassifier avec un accuracy au dessus de 0.81.

Étrangement, les autres modèles affichent avec Optuna un score inférieur au premier modèle, on peut penser à un overfitting sur ces modèles.

Nous choisissons ici le modèle XGBClassifier par rapport au KNeighborsClassifier par rapport à la vitesse d'exécution du modèle. En effet, dans la pratique, le modèle KNN est plus adapté pour des plus petits dataframes, notamment inférieur à 100 000 lignes.

La matrice de confusion et le rapport de classification finaux nous donnent donc :

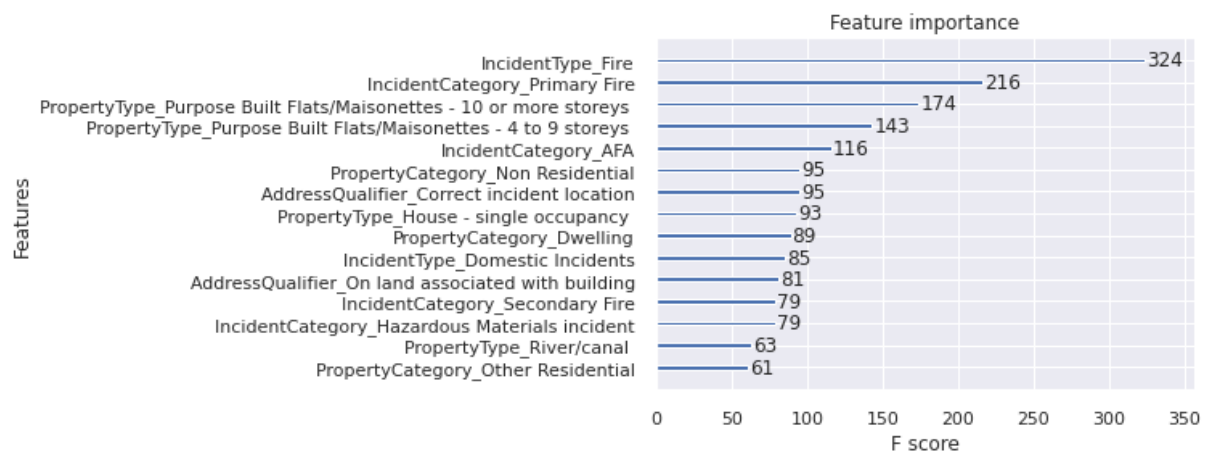
Prédiction	1.0	2.0	3.0	5.0
Realité				
1.0	34563	3254	4	73
2.0	4173	14456	7	105
3.0	449	1802	42	62
4.0	73	414	8	113
5.0	146	1041	0	572

	precision	recall	f1-score	support
1.0	0.88	0.91	0.89	37894
2.0	0.69	0.77	0.73	18741
3.0	0.69	0.02	0.03	2355
4.0	0.00	0.00	0.00	608
5.0	0.62	0.33	0.43	1759
accuracy			0.81	61357
macro avg	0.57	0.41	0.42	61357
weighted avg	0.80	0.81	0.79	61357

On s'aperçoit que nous avons un F1-score élevé pour les 2 premières classes (1 et 2 camions) et correct pour la classe 5 alors qu'elle est très mauvaise, voire absente pour les classes 3 et 4 camions.

Les résultats présentés du modèle pourraient se restreindre à 3 classes, 1 et 2 camions pour la majorité des incidents et 5 camions pour quelques rares incidents plus graves, les classes 3 et 4 étant quasiment inexistantes.

Nous avons aussi voulu afficher les variables les plus importantes de notre modèle. Sans surprise, ce sont les catégories d'incidents (feu mineur, feu de cheminée, alarmes incendie) et les types de lieu (résidentiels ou non résidentiels) qui ont le plus d'importance dans le modèle.



En imaginant aller plus loin, un modèle de ce type pourrait être très important dans l'optimisation des ressources à mettre à disposition d'un incident. Nous avons travaillé sur le nombre de véhicules mais le choix du type de véhicule serait aussi intéressant à étudier. L'utilisation adéquate des ressources permettrait de réduire les coûts en évitant d'envoyer trop de véhicules et ainsi augmenter leur disponibilité. Mais aussi, il permettrait d'envoyer immédiatement les véhicules adaptées à la situation et réduire les temps d'intervention.

3.2. Prédiction de l'atteinte de l'objectif des 360s

Suite à l'analyse des données et de la création de certaines variables, une base de données est finalement définie et sera celle utilisée lors des entraînements des algorithmes : « base_ml.pkl » qui sera utilisé par le dataframe « df », structure de données bidimensionnelle.

Les notebooks exploités dans ce chapitre sont :

- “3.2 Prédiction de l'atteinte de l'objectif des 360s Workbook.ipynb” : notebook de travail ;
- “3.2 Prédiction de l'atteinte de l'objectif des 360s.ipynb” : notebook contenant uniquement le modèle retenu.

Nous voulons dans un premier temps prédire si le temps d'intervention des premiers camions est supérieur à 360sec ou inférieur. En effet, en étudiant des communications effectuées par les pompiers de Londres, la valeur de 360sec est un seuil défini comme objectif important.



Pour transformer notre base de données en un problème de classification binaire, on crée une variable qui sera notre target à savoir `Goal` qui prendra la valeur 0 si inférieur à 360sec et la valeur 1 si supérieur à 360sec.

Indicateur de qualité du modèle

Pour mesurer la qualité de notre algorithme de classification, plusieurs scorers existent : accuracy, f1-score etc. Étant donné qu'il est important pour l'opérateur de détecter les interventions qui nécessitent plus de 360sec, nous nous focaliserons sur le RECALL de la classe 1, tout en considérant quand même d'autres scorers.

- NB : Le recall est également appelé **sensitivity** (sensibilité), **true positive rate** ou encore **hit rate** (taux de détection). Il correspond au **taux d'individus positifs détectés** par le modèle :

$$\frac{TP}{TP + FN}$$

Avec : TP true positive, FN false negative

Le recall mesure donc la capacité du modèle à détecter l'ensemble des individus positifs, dans notre cas, l'ensemble des camions dépassant un temps d'intervention de 360 secondes.

Sélection et transformations des features

On sélectionne les features les plus appropriées. En effet, certaines contiennent l'information d'autres : par exemple « congestion_rate » contient de façon sous-jacente les notions de jour et d'années. Or, des features trop corrélées entre elles pourraient nuire à la qualité du modèle.

On conservera :

DeployedFromLocation, Appliance, PropertyCategory, AddressQualifier, IncidentType, Distance, TotalOfPumpInLondon_Out, Station_Code_of_ressource, IncidentStationGround_Code, PumpAvailable, month, temp, precip, cloudcover, visibility, conditions, workingday, school_holidays, congestion_rate, Goal.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 547865 entries, 0 to 547864
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   DeployedFromLocation                 547865 non-null object
1   Appliance                           547865 non-null object
2   PropertyCategory                     547865 non-null object
3   AddressQualifier                     547865 non-null object
4   IncidentType                         547865 non-null object
5   Distance                             547865 non-null float64
6   TotalOfPumpInLondon_Out              547865 non-null int64
7   Station_Code_of_ressource            547865 non-null object
8   IncidentStationGround_Code           547865 non-null object
9   PumpAvailable                        547865 non-null int64
10  month                                547865 non-null int64
11  temp                                 547865 non-null float64
12  precip                               547865 non-null float64
13  cloudcover                           547865 non-null float64
14  visibility                           547865 non-null float64
15  conditions                           547865 non-null object
16  workingday                           547865 non-null object
17  school_holidays                      547865 non-null object
18  congestion_rate                       547865 non-null float64
19  Goal                                 547865 non-null int64
dtypes: float64(6), int64(4), object(10)
memory usage: 87.8+ MB
```

2 types de features :

- Les features catégorielles
- Les features numériques

Pour les features catégorielles, nous utiliserons un encodage de type OneHot en utilisant `get_dummies()`. Cet encodage très pratique permet de décomposer une feature catégorielle en colonnes, chaque colonne correspondant à une des features catégorielles et prenant les valeurs 0 ou 1 suivant que la valeur apparaissait initialement ou non sur la ligne.

Dans les features numériques, on va distinguer les features purement numériques (comme `Distance`) des features numériques « catégorielles » (comme `workingday`).

Pour les premières, nous utiliserons `StandardScaler()` de la bibliothèque `scikit-learn` qui permet de centrer et réduire chacune de ces features.

Pour les secondes, nous n'appliquerons pas d'opérations dessus.

Nous obtenons une très large matrice, quasi creuse, e.g. remplie de 0 et de 1 ainsi que de valeurs numériques centrées -réduites, avec 255 variables :

```
✓ [14] df.shape  
0 s  
(547865, 255)
```

Création du jeu de d'apprentissage/de validation/ de test

Nous décomposons ensuite notre dataframe en un jeu d'entraînement (`X_train, y_train`) et un jeu de test (`X_test, y_test`), sachant que `y_train` et `y_test` ne contiennent que la target `Goal`, et `X_train/X_test` les features restantes qui serviront à la prédiction.

Pour créer les 2 jeux de données, nous allons nous baser sur les références d'incidents. Pour cela, nous créons au préalable, avant le choix des features à conserver, une fonction qui renvoie les index des lignes des incidents que l'on affecte en test et train via un tirage sans remise.

Une fois fait, la sélection des features effectuée plus haut supprimera la variable `IncidentNumber`.

Pour diminuer d'éventuelles biais du jeu d'entraînement lors de l'apprentissage, nous utiliserons une "cross-validation".

En effet, il est acquis qu'un modèle doit être évalué sur une base de test différente de celle utilisée pour l'apprentissage. Mais la performance est peut-être juste l'effet d'une aubaine et d'un découpage

particulièrement avantageux. Pour être sûr que le modèle est robuste, on recommence plusieurs fois en décomposant le jeu d'apprentissage en k plis : k-1 serviront de plis d'apprentissage et le restant de test, puis on change de plis de test et on recommence jusqu'à ce que chacun des plis ait été le plis de test. Ici nous choisirons K=5.



Nous observons que les classes de la target sont déséquilibrées, il faudra en tenir compte dans la construction de nos futurs modèles.

```
[15] df.Goal.value_counts()

0    336526
1    211339
Name: Goal, dtype: int64
```

1er essai : Régression Logistique

Nous souhaitons commencer un algorithme simple de classification qui sera notre étalon pour tous les autres modèles. La régression logistique s'y prête bien, c'est un modèle simple et robuste, facile à modéliser et à expliquer.

Dans un premier temps, sans tenir compte du déséquilibre, nous obtenons les tableaux suivant qui nous donnent des indications détaillées :

```
Matrice de confusion:
col_0    0    1
Goal
0    59261  7809
1    14149 28280

Rapport des résultats de classification:
              pre      rec      spe      f1      geo      iba      sup
0          0.81      0.88      0.67      0.84      0.77      0.60      67070
1          0.78      0.67      0.88      0.72      0.77      0.58      42429
avg / total      0.80      0.80      0.75      0.80      0.77      0.59      109499

(LogisticRegression(), array([0, 0, 0, ..., 0, 0, 1]))
```

Nous observons que le recall est assez mauvais, seulement 0.67 en moyenne sur les 5-plis ('Résultat de la Cross Validation').

Certainement le déséquilibre entre les classes 0 et 1 influence grandement notre modèle.

En appliquant le modèle sur le jeu de test, on observe que la généralisation du modèle reste toujours problématique, avec un recall sur la classe 1 à 0.67 ('Rapport des résultats de classification')

2ème essai : Régression Logistique « balanced »

Nous tiendrons compte cette fois du déséquilibre en précisant à la régression logistique le paramètre « class_weight= 'balanced' ».

La méthode des **class weights** permet de prendre en compte le caractère biaisé de la distribution du dataset et de créer un *modèle pénalisé*. Il s'agit ici de simplement attribuer des poids différents aux différentes classes de notre dataset, en donnant un poids plus important aux classes minoritaires, afin d'influencer le modèle lors de son entraînement.

Nous pénalisons ainsi plus fortement une erreur de classification d'une classe minoritaire par rapport à une erreur de classification d'une classe majoritaire.

En prenant 'balanced', l'algorithme va pénaliser les classes majoritaires/minoritaires en associant des poids inversement proportionnel à leur fréquence

Nous obtenons alors le résultat suivant :

```

Cross(lr_balanced, "recall")

Résultat de la Cross Validation:
[0.7518205  0.73992067 0.75238293 0.77360725 0.77588657]
0.758723580605056

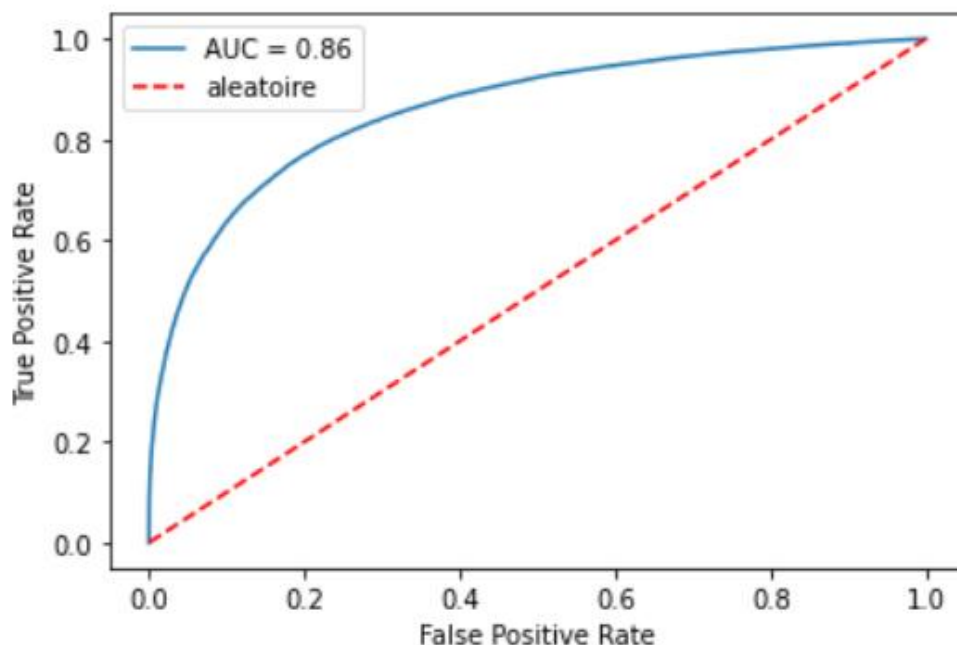
Matrice de confusion:
col_0    0    1
Goal
0      54134 12936
1      10125 32304

Rapport des résultats de classification:
              pre      rec      spe      f1      geo      iba      sup
0      0.84      0.81      0.76      0.82      0.78      0.62      67070
1      0.71      0.76      0.81      0.74      0.78      0.61      42429
avg / total      0.79      0.79      0.78      0.79      0.78      0.62      109499

(LogisticRegression(class_weight='balanced', solver='newton-cg'),
 array([0, 0, 0, ..., 1, 1, 1]))
```

Le recall de la classe 1 s'est sensiblement amélioré passant, pour le jeu de test, de 0.67 à 0.76 (en moyenne) !! Quasiment 0.10 d'amélioration. Le temps de calcul, lui, reste relativement long.

Nous pouvons ensuite tracer la courbe ROC qui permet aussi d'évaluer la performance d'un modèle de classification binaire :



L'aire sous la courbe (AUC) est de $0.86 > 0.5$ (0.5 étant le tirage aléatoire) ce qui est plutôt bien.

D'après la lecture du graphique précédent, le seuil de prédiction pourrait être abaissé par à seuil = 0.19 ce qui donnerait les résultats suivant :

col_0	0	1							
Goal									
0	27717	39353							
1	2213	40216							
			pre	rec	spe	f1	geo	iba	sup
	0		0.93	0.41	0.95	0.57	0.63	0.37	67070
	1		0.51	0.95	0.41	0.66	0.63	0.41	42429
avg / total			0.76	0.62	0.74	0.61	0.63	0.39	109499

Le recall de la classe 1 passe à 0.95 !! ce qui est excellent, les véhicules effectuant plus de 360sec étant très bien détectés.

Cependant, ceci est au détriment de la précision : nous détectons "trop" bien finalement car près de la moitié des camions inférieure à 360sec sont prédits mettant plus de 360sec ! On conservera donc par défaut un seuil à 0.5 pour conserver cet équilibre précision/recall (mesuré par le scorer "f1")

Nous allons dès lors essayer un modèle plus complexe pour mieux capter le pattern et améliorer si possible en même temps le recall.

3ème essai : SGDClassifier

Nous restons dans la famille des algorithmes linéaires mais celui-ci est optimisé par la descente de gradient, le SGDClassifier : le gradient de la perte est estimé chaque échantillon à la fois et le modèle est mis à jour en cours de route avec un programme de force décroissant (aka taux d'apprentissage).

Nous utilisons toujours le « class_weight= 'balanced' » pour tenir compte du déséquilibre.

Nous obtenons alors :

```
[28] sg = SGDClassifier(class_weight="balanced", random_state=42)
      model, y_pred = Cross(sg, "recall")
```

Résultat de la Cross Validation:

```
[0.80933633 0.79231543 0.80599136 0.82564679 0.74800189]
0.7962583624415369
```

Matrice de confusion:

col_0	0	1
Goal		
0	52493	14577
1	9291	33138

Rapport des résultats de classification:

	pre	rec	spe	f1	geo	iba	sup
0	0.85	0.78	0.78	0.81	0.78	0.61	67070
1	0.69	0.78	0.78	0.74	0.78	0.61	42429
avg / total	0.79	0.78	0.78	0.78	0.78	0.61	109499

Un recall sur la classe 1 de 0.78 et une précision qui reste acceptable à 0.69, notre modèle stochastic devient notre meilleur modèle.

Nous essayons ensuite plusieurs modèles différents et utilisons Optuna sur certains modèles afin de les améliorer. Optuna est un framework d'optimisation des hyper-paramètres, particulièrement apprécié lors des concours Kaggle.

Le tableau suivant présente le fruit de ces tentatives :

	Recall	
	Train	Test
LogisticRegression	0.67	0.67
LogisticRegression (class_weight="balanced")	0.76	0.76
Logistic Regression (class_weight="balanced", Optuna)	0.76	0.76
SGDClassifier (class_weight="balanced")	0.80	0.78
SGDClassifier (class_weight="balanced", Optuna)	0.79	0.80
LinearSVC (class_weight="balanced")	0.76	0.76
LinearSVC (class_weight="balanced", Optuna)	0.75	0.75

Première remarque, une fois la méthode class_weight utilisée, les modèles produisent des résultats assez similaires. Les écarts sur le jeu de test sont de l'ordre de 5%. Deuxième remarque, contrairement à ce qu'on aurait pu espérer, Optuna n'améliore que peu les modèles.

2 modèles sont au-dessus du lot : le SGDClassifier et le SGDClassifier optimisé par Optuna. Après observations des résultats sur le Y_test et de la matrice de confusion de chacun de ces 2 modèles, notre choix se porte sur le **SGDClassifier optimisé par Optuna**.

Recherche d'un meilleur modèle avec TPOT

Tpot est un package d'apprentissage automatique automatisé en python qui utilise des concepts de programmation génétique pour optimiser le pipeline d'apprentissage automatique.

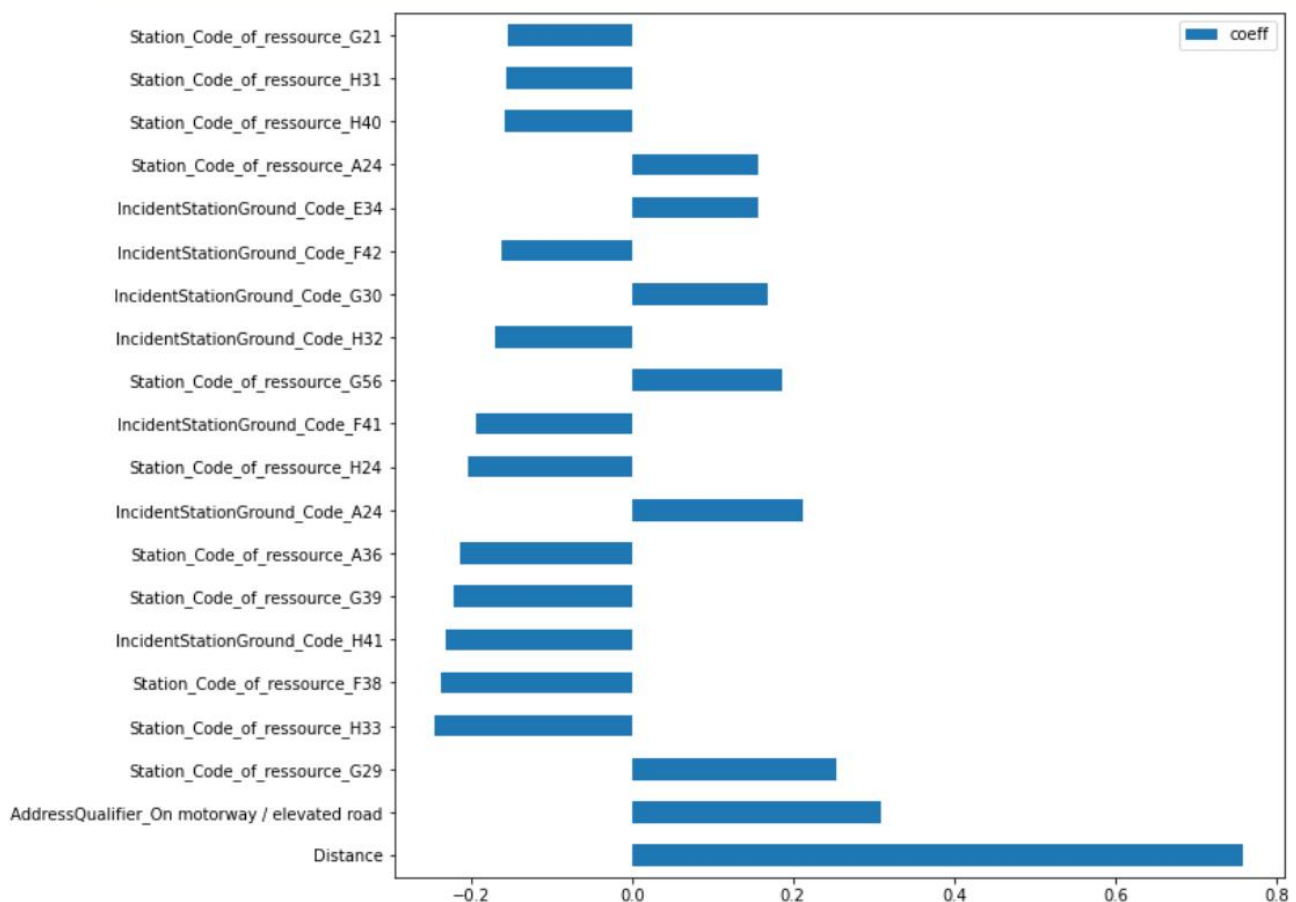
Il automatise la partie la plus fastidieuse de l'apprentissage automatique en explorant intelligemment des milliers de possibles pour trouver le meilleur paramètre possible qui convient à vos données.

Tpot est construit sur scikit-learn.

Après plusieurs heures d'utilisation (~3h), le meilleur modèle trouvé propose un recall pour la classe 1 à 0.78, derrière notre meilleur modèle.

Explications des prédictions du modèle sélectionné

Nous allons observer l'importance des features dans la prédiction, nous limitons l'affichage aux 10 plus importantes features, en fonction du coefficients trouvé par notre modèle :



Nous observons que la feature `Distance` possède une part contributive très importante dans la prédiction de notre modèle : son importance est considérable. Logique, plus l'évènement est loin des stations, plus les camions mettront de temps à y parvenir.

`AddressQualifier_On motorway/elevated road` joue un rôle important, à moindre mesure. Cette feature est en fait une des valeurs de la colonne originelle `AdressQualifier` : dès que l'intervention est sur autoroute, le temps prédit va être impacté à la hausse.

Dans la phase d'exploration/visualisation des données, nous avons remarqué que les temps d'intervention variaient en fonction des stations. Ce que nous retrouvons ici le confirme : les identifiants de certaines stations font partie des 10 features les plus importantes de notre modèle.

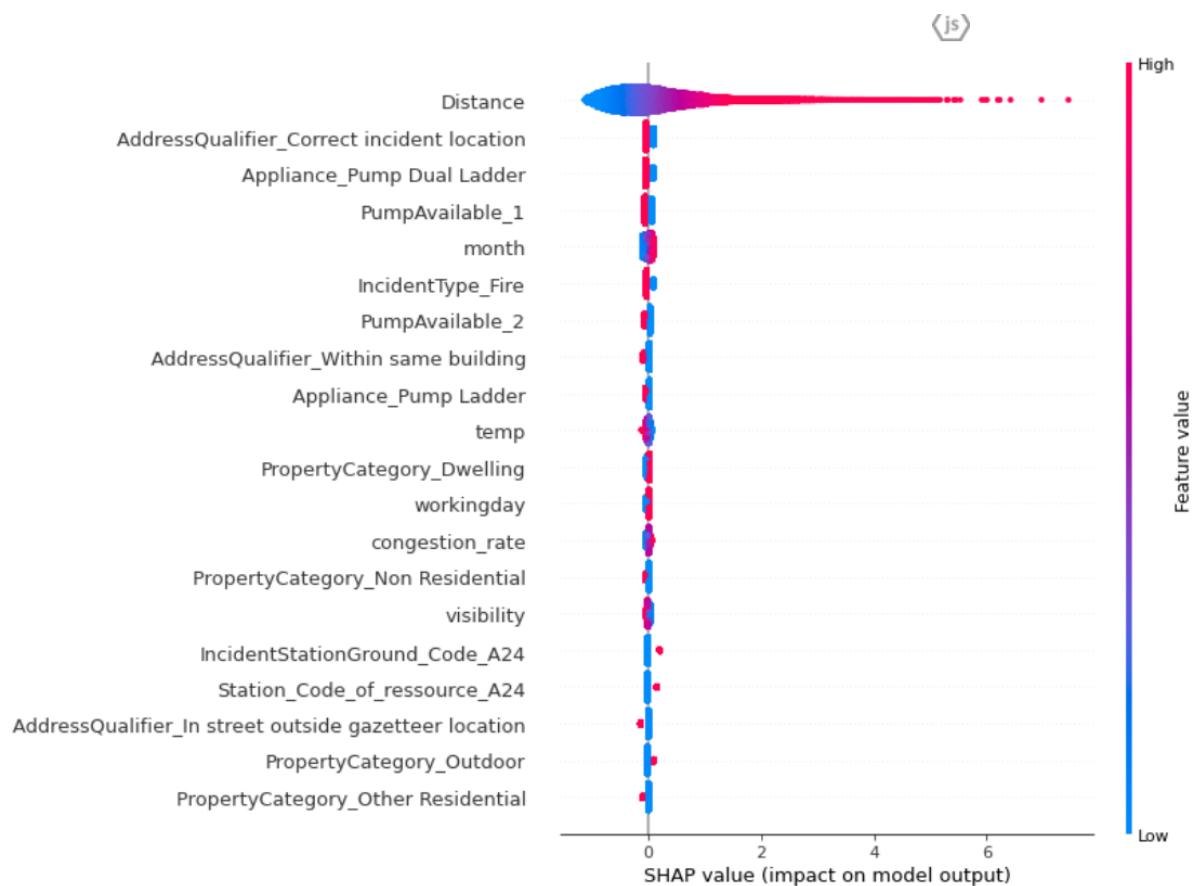
Une autre manière de visualiser la structure des prédictions de notre modèle est de se baser sur les valeurs de Shapley. La valeur de Shapley provient de la théorie des jeux : l'idée est de moyenner l'impact qu'une variable a pour toutes les combinaisons de variables possibles.

On va donc chercher à calculer cette valeur pour toutes les variables à chaque exemple du dataset.

Grâce à la valeur de Shapley, on peut déterminer l'effet des différentes variables d'une prédiction pour un modèle qui explique l'écart de cette prédiction par rapport à la valeur de base. Cependant la valeur Shapley est très coûteuse car complexité importante.

L'algorithme TreeExplainer de la librairie Shap permet de diminuer fortement cette complexité en optimisant via des ensembles d'arbres de décisions.

Intéressons nous d'abord à l'importance globale des variables calculées par les valeurs de Shap :



Grâce au fait que les valeurs sont calculées pour chaque exemple du dataset, il est possible de représenter chaque exemple par un point et ainsi avoir une information supplémentaire sur l'impact de la variable en fonction de sa valeur.

Ici, Distance (qui est la feature la plus importante et de loin) a un impact positif quand la valeur de cette feature est élevée : plus la distance est élevée, plus le temps d'intervention dépasse les 360 secondes.

De la même manière, on ne dépassera pas les 360 secondes si le camion est un Dual Pump, si l'AdressQualifier est Correct incident location, si IncidentType est Fire etc...

Intéressons nous du coup sur l'impact des features localement à travers plusieurs exemples, tout d'abord lorsque le modèle prédit un temps inférieur à 360 secondes et qu'il prédit juste :

Test data (actual observation): 0
Model's prediction: 0



Test data (actual observation): 0
Model's prediction: 0



En rouge, les variables qui ont un impact positif (contribuent à ce que la prédiction soit plus élevée que la valeur de base) et, en bleu, celles ayant un impact négatif (contribuent à ce que la prédiction soit plus basse que la valeur de base)

Nous étudions deux exemples d'interventions dont les valeurs de Shap sont moins élevées que la valeur de base : pour le premier -0.15 et pour le deuxième -0.41, la valeur de base du dataset étant à -0.04883.

Nous voyons cependant que les raisons du modèle pour lesquelles le temps d'intervention est inférieur à 360 secondes sont différentes :

- dans le premier exemple, les features Month(1), PumpAvailable_2(1), PropertyCategory_Non Residential(1) contribuent à ce que la valeur Shap de la prédiction soit plus basse que la valeur Shap de base et donc que le modèle prédise un temps inférieur à 360 secondes, alors que dans le deuxième, ce sont les features Distance(-0.5198), AddressQualifier_In street outside gazetteer location(1) et month(1) ;

- dans le premier exemple, les features `Appliance_Pump_Dual_Ladder(0)`, `Distance(0.02251)`, `PumpAvailable_1(0)` contribuent à ce que la valeur Shap de la prédiction soit plus haute que la valeur Shap de base et donc que le modèle prédise un temps supérieur à 360 secondes, alors que dans le deuxième ce sont les features `PropertyCategory_Outdoor(1)`, `AddressQualifier_Correct incident location(0)` et `IncidentStationGround_Code_G28(1)`.

Autre exemple, le modèle prédit ici que le temps d'intervention dépassera 360 secondes et cette prédiction est juste :



La valeur de Shap est de 1.54, supérieur à la valeur de base donc le temps d'intervention sera prédit supérieur à 360 secondes. `Distance(2.282)` contribue fortement à ce que la valeur Shap dépasse la valeur de base et donc à dépasser 360 secondes.

3.3. Prédiction du temps d'arrivée

Le dernier point de la problématique que nous nous sommes fixée, concerne l'annonce d'un temps d'attente estimée de l'arrivée des secours sur les lieux de l'incident.

Les notebooks exploités dans ce chapitre sont :

- ["3.3 Prédiction temps arrivée Workbook.ipynb"](#) : le notebook de travail, contenant toutes les études exposées ci-dessous ;
- ["3.3 Prédiction temps arrivée.ipynb"](#) : le notebook de mise en oeuvre du seul modèle retenu.

3.3.1 Travaux préparatoires

Partant de notre base nettoyée et complétée, et comme nous allons nous intéresser à la même variable cible que le modèle précédent (le temps de réponse), mais sous un angle différent (régression au lieu de classification), nous avons fait le choix de conserver les mêmes variables explicatives, résumées ci-dessous (après modification de certains formats de variables) :

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547865 entries, 0 to 547864
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   AttendanceTimeSeconds                 547865 non-null int64
1   DeployedFromLocation                 547865 non-null object
2   Appliance                           547865 non-null object
3   PropertyCategory                    547865 non-null object
4   AddressQualifier                    547865 non-null object
5   IncidentType                        547865 non-null object
6   Distance                             547865 non-null float64
7   TotalOfPumpInLondon_Out             547865 non-null int64
8   Station_Code_of_ressource           547865 non-null object
9   IncidentStationGround_Code          547865 non-null object
10  PumpAvailable                       547865 non-null object
11  month                               547865 non-null uint8
12  temp                                547865 non-null float64
13  precip                              547865 non-null float64
14  cloudcover                          547865 non-null float64
15  visibility                          547865 non-null float64
16  conditions                          547865 non-null object
17  workingday                          547865 non-null uint8
18  school_holidays                    547865 non-null uint8
19  congestion_rate                     547865 non-null float64
dtypes: float64(6), int64(2), object(9), uint8(3)
memory usage: 92.9+ MB
```

Pour la suite des travaux :

- AttendanceTimeSeconds sera dans cette partie, notre variable cible ;
- les 19 autres variables seront nos variables explicatives :
 - les 9 variables explicatives de type 'object' seront dichotomisées (par la fonction get_dummies) ;
 - les 7 variables explicatives de type 'int64' et 'float64' feront l'objet d'une normalisation ;
 - les 3 variables explicatives de type 'uint8' ne feront l'objet d'aucun traitement : la variable month que nous gardons, ainsi que les variables workingday et school_holidays (variables binaires de type 0/1).

Nous nous sommes livrés à une analyse rapide des coefficients de corrélation :

AttendanceTimeSeconds	1
Distance	0.66
TotalOfPumpInLondon_Out	0.1
PumpAvailable	-0.14
month	0.02
temp	0.04
precip	0.01
cloudcover	-0
visibility	0
conditions	0.01
workingday	0.02
school_holidays	-0
congestion_rate	0.05
location_home_other	0.06
appliance	0.03
prop_cat	0.03
addr_qualifier	0.04
inc_type	0.01
station_ressource	0.01
station_ground	0.01
AttendanceTimeSeconds	-

De cette analyse, nous pouvons constater que :

- aucune variable ne présente de forte corrélation (coefficient supérieur à 0.8) à AttendanceTimeSeconds ;
- seule une variable (Distance) domine les autres, avec un coefficient de 0.66, ce qui confirme l'intuition que nous pouvions avoir (la distance nous semble être la principale origine du temps d'intervention : plus l'incident est éloigné, plus le temps de réponse augmente) ;
- dans une moindre mesure, on constate que le nombre de véhicules disponibles influence également le temps d'intervention (plus le nombre de véhicules disponibles pour l'intervention est élevé, au mieux les pompiers peuvent organiser une réponse en un temps inférieur).

Ces coefficients reflètent à notre sens parfaitement la mission confiée au pompiers, à savoir une intervention qui soit la plus rapide possible, en toute circonstance.

Les jeux de données d'entraînement et de test, toujours créés en cohérence avec chacun des modèles précédent (sélection réalisée sur les références d'incidents), se résument ainsi :

```
Dimensions (X_train, y_train) : ((438366, 254), (438366,))
Dimensions (X_test, y_test)   : ((109499, 254), (109499,))
Le jeu de données test représente 20.0 % des incidents, soit 20.0 % du
nombre total de véhicules.
```

3.3.2 Modélisation

3.3.2.1 Métriques et choix des modèles

Concernant les **métriques d'évaluation** des modèles, nous sommes dans un premier temps partis sur une évaluation basée sur le coefficient de détermination (R^2), ainsi que le racine de l'erreur quadratique (RMSE).

Rapidement, compte tenu des faibles niveaux de R^2 (de l'ordre de 0.40), et des niveaux élevés de RMSE (généralement supérieure à 100), il nous a paru nécessaire, pour une meilleure compréhension des résultats, d'avoir une métrique exprimée dans l'unité de notre variable cible, c'est-à-dire en secondes.

Aussi, nous avons défini comme métrique principale de décision, **l'erreur absolue moyenne** (Mean Absolute Error, ou **MAE**), que nous avons complétée par l'erreur absolue médiane (Median Absolute Error, qui elle, sera moins sensible aux quelques grandes erreurs qui pourraient être faites par le modèle) ; ces 2 métriques étant exprimées dans l'unité de notre variable cible, donc en secondes. Ces métriques seront calculées tant pour le jeu de données d'entraînement, que pour le jeu de test.

Dans le choix des modèles, compte tenu de la taille importante des jeux de données (nombre d'observations et de variables après dichotomisation), et après quelques essais, nous avons très vite fait le choix d'exclure les algorithmes de type KNeighbors ou RandomForest (temps de traitement élevés).

Dans un premier temps, c'est un modèle LinearRegression que nous avons mis en oeuvre :

- sur la totalité des variables ;
- mais aussi sur la seule variable `Distance` (la plus corrélée au temps de réponse), afin de voir si elle ne pouvait suffire à elle seule, à obtenir des résultats satisfaisants.

Ce premier modèle nous servira ensuite de référence, pour juger des performances des autres modèles.

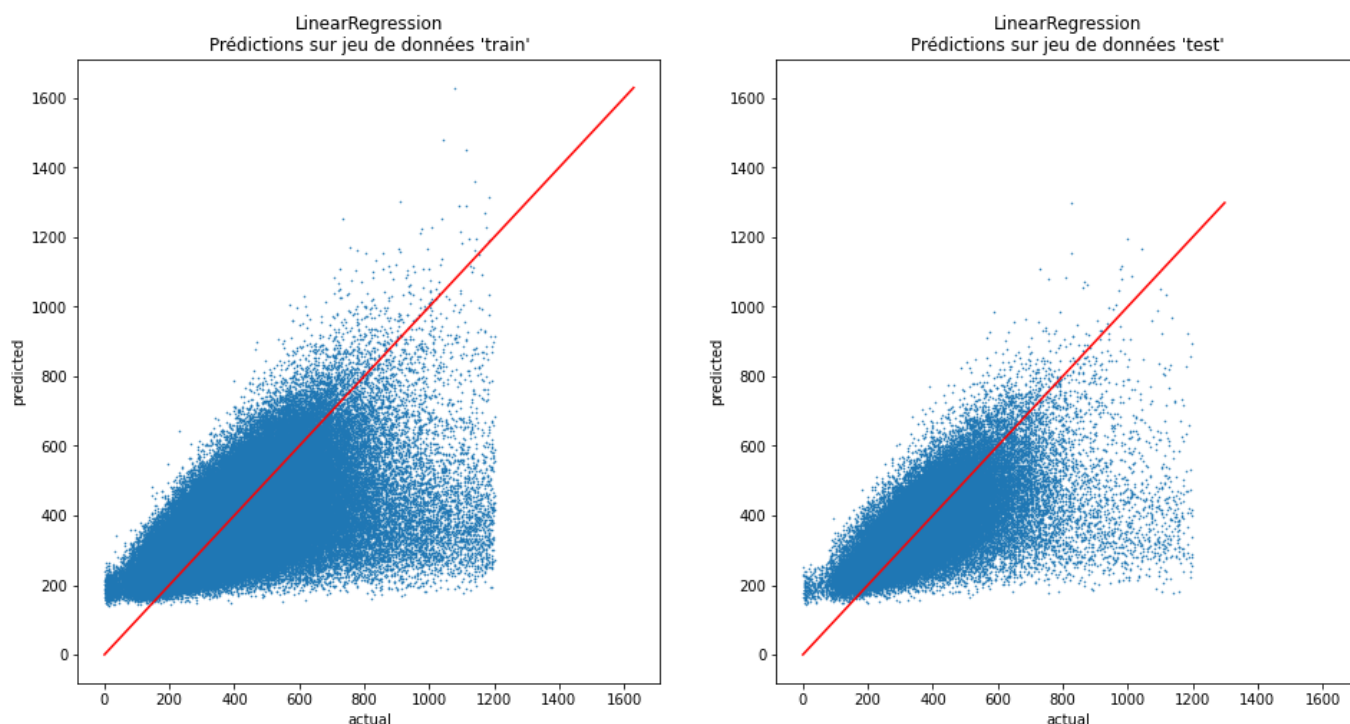
Les métriques obtenues sur la régression linéaire, sont les suivantes :

Métrique	LinearRegression	LinearRegression_distance_only
R ² train	0.46	0.43
R ² test	0.45	0.43
RMSE train	106.00	109.00
RMSE test	106.00	108.00
Mean AE train	70.00	72.00
Mean AE test	70.00	72.00
Median AE train	51.00	54.00
Median AE test	51.00	53.00

Ce modèle nous permet d'atteindre des prédictions avec une MAE de 70 secondes, autant sur l'ensemble d'entraînement que sur l'ensemble de test. Nous ne notons donc pas de sur-apprentissage (ce qui est également confirmé par le niveau du coefficient R², quasiment identique sur nos 2 jeux de données).

La considération de la seule variable `Distance`, nous permet déjà d'atteindre une MAE de 72 secondes, ce qui confirme qu'à elle seule, cette variable permet déjà d'atteindre un niveau correct de prédictions, et que les autres variables sont peu porteuses d'informations pour les prédictions.

Nous avons également représenté graphiquement les temps prédits, en fonction des temps réels, sur chaque jeu de données (ainsi que la droite rouge représentant le but idéal, "actual = predicted") :



Cette représentation sur ce premier modèle, nous permet de constater :

- que les temps très courts ne sont pas appréhendés correctement (ce modèle semble de base, avoir un plancher de prédictions situé autour de 175 secondes) ;

- que les temps longs semblent encore plus mal appréhendés (le nuage de points peine à se regrouper autour de la droite “actual = predicted”).

Pour la suite de nos investigations, nous avons alors testé d’autres modèles, dont certains que nous avons identifiés en cherchant les modèles qui seraient les plus adaptés à gérer des valeurs extrêmes.

Nous avons travaillé sur les modèles BayesianRidge, SGDRegressor, XGBoost, MLPRegressor (réseau de neurones, dans Scikit-Learn), mais aussi PassiveAggressiveRegressor, et HuberRegressor (ces 2 derniers modèles étant ceux trouvés pour une meilleure gestion de valeurs extrêmes).

Sur chacun de ces modèles, pour toutes les variables mais aussi pour la seule variable `Distance`, nous nous sommes globalement livrés aux mêmes travaux :

- tenter de trouver les meilleurs paramètres (via utilisation de GridSearchCV) ;
- entraîner le modèle ;
- générer les prédictions ;
- calculer les métriques ;
- représenter graphiquement les nuages de points des prédictions vs temps réels ;
- et enfin, analyser l’interprétabilité, via les valeurs de Shapley et les coefficients les plus significatifs des variables.

Sans entrer ici dans le détail de chacun des modèles expérimentés, nous présentons ci-dessous la synthèse des métriques.

A noter que les modèles basés sur la seule distance ont été exclus, car présentant systématiquement des performances moindres par rapport aux autres modèles.

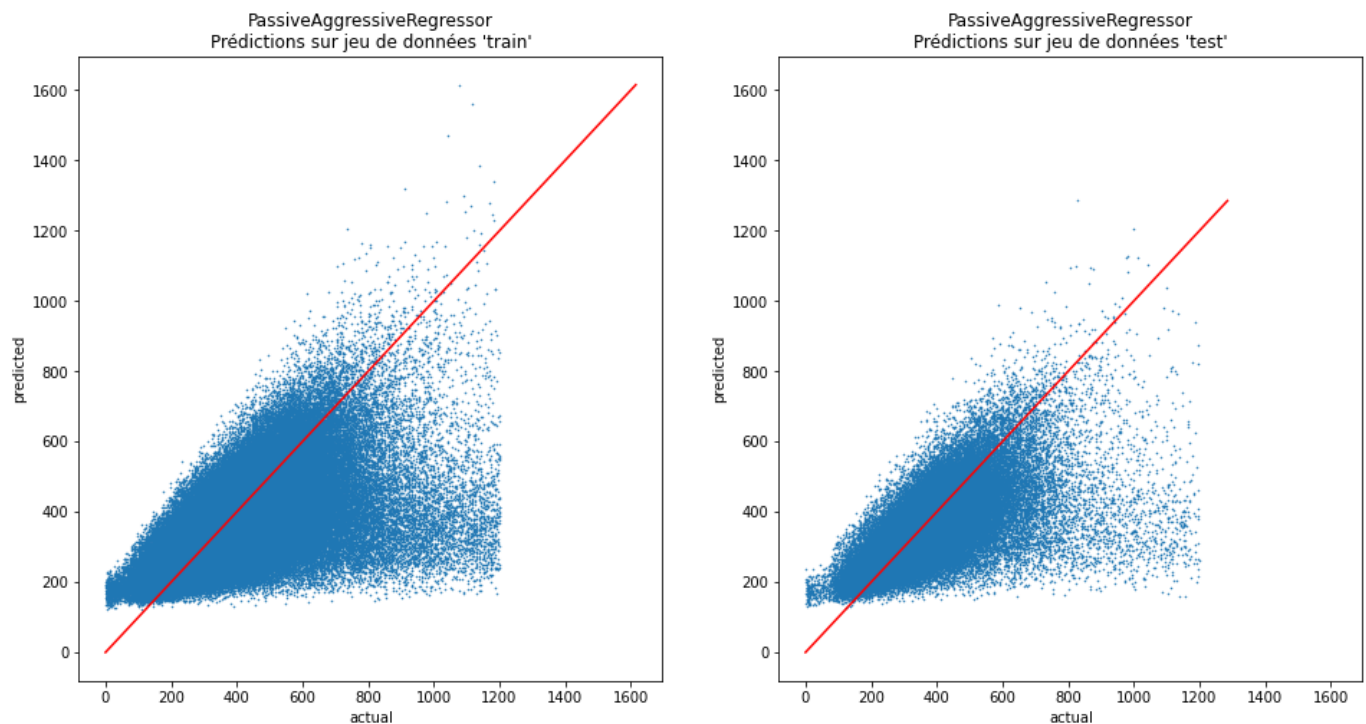
	LinearRegression (lr)	Bayesian Ridge (br)	PassiveAggressive Regressor (par)	XGBoost (xgb)	SGDRegressor (sgd)	HuberRegressor (hub)	MLPRegressor (mlpr)
R² train	0.46	0.46	0.44	0.56	0.46	0.43	0.43
R² test	0.45	0.45	0.44	0.48	0.45	0.43	0.43
RMSE train	106.00	106.00	107.00	95.00	106.00	108.00	109.00
RMSE test	106.00	106.00	107.00	103.00	106.00	108.00	108.00
Mean AE train	70.00	70.00	68.00	62.00	70.00	68.00	73.00
Mean AE test	70.00	70.00	68.00	67.00	70.00	67.00	72.00
Median AE train	51.00	51.00	47.00	46.00	51.00	45.00	54.00
Median AE test	51.00	51.00	47.00	48.00	51.00	45.00	54.00

(en-tête de colonne = nom du modèle + son abréviation utilisée dans les noms de variables)

Les conclusions à la lecture de ces métriques, et par les analyses graphiques, sont les suivantes :

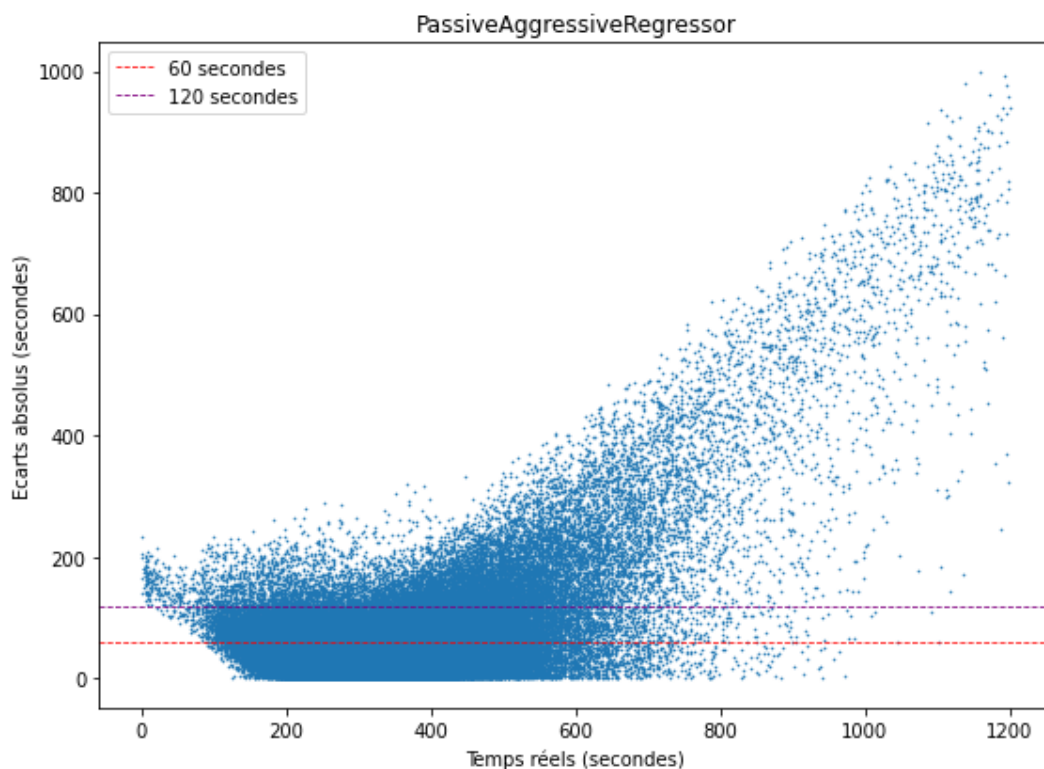
- le modèle présentant les MAE les plus basses est celui pour lesquels nous constatons une tendance flagrante au sur-apprentissage (XGBoost) ;
- tous les modèles sous-estiment clairement les temps de réponses dès lors qu'ils sont en réalité supérieurs à 600 secondes (comme évoqué pour la régression linéaire), ce qui est l'erreur de prédiction la plus dérangeante (risques humain et matériel, si le temps annoncé est sous-estimé) ;
- tous les modèles sauf XGBoost sur-estiment les temps les plus courts (déjà évoqué pour la régression linéaire), ce qui est à notre sens, moins problématique, dans la mesure où l'on risque d'annoncer un temps qui pourra être en réalité plus court ;
- le modèle qui nous semble le plus intéressant, serait PassiveAggressiveRegressor, car présentant les caractéristiques que nous estimons les plus équilibrées : un niveau de MAE acceptable de 68 secondes, des métriques globalement identiques entre les jeux d'entraînement et de test, donc sans sur-apprentissage, pour un temps d'entraînement très court, par rapport à d'autres modèles, avec en prime, une erreur absolue médiane parmi les plus faibles.

Ci-dessous, les nuages de points des temps prédits en fonction des temps réels, pour le modèle PassiveAggressiveRegressor :

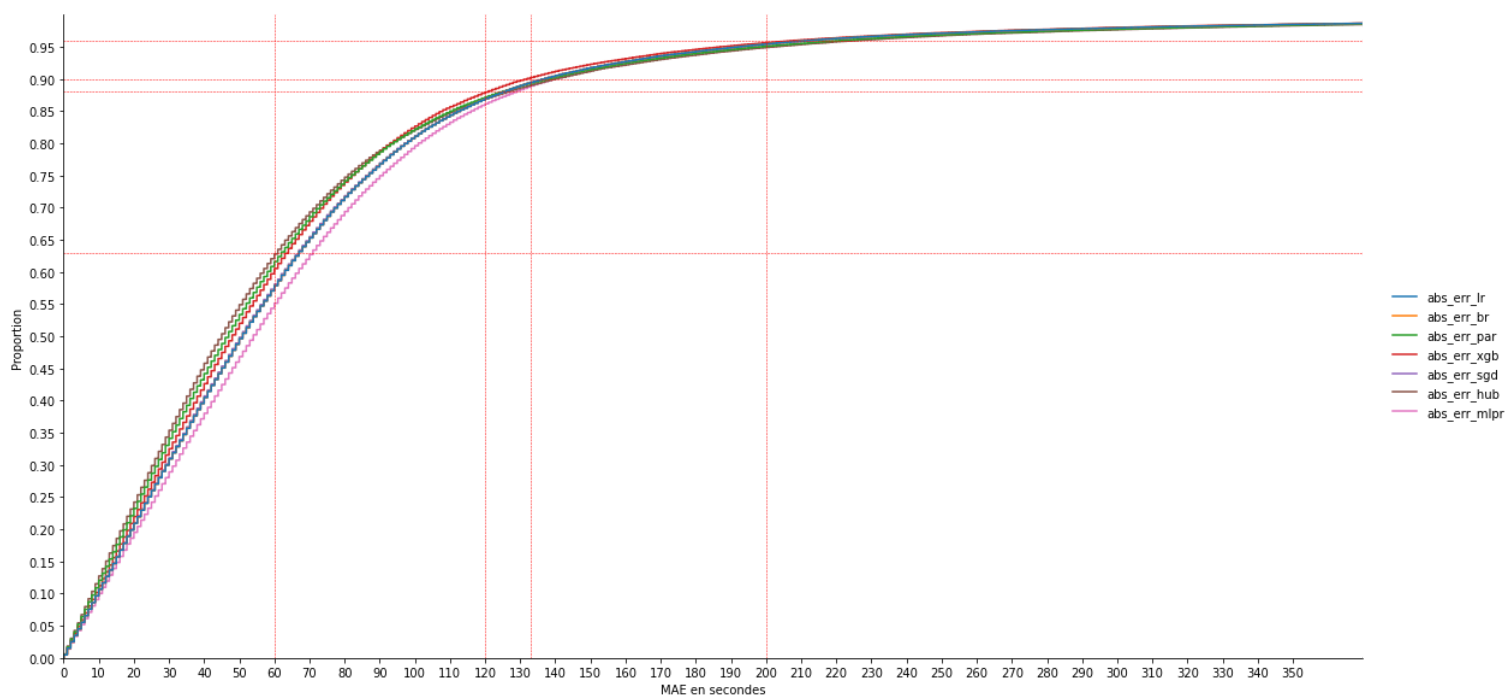


Afin de compléter notre analyse, et choisir le meilleur modèle, nous avons alors étudié la répartition des écarts absolus, en 2 temps :

- une analyse graphique du nuage de points des écarts absolus :
(exemple pour PassiveAggressiveRegressor)



- une analyse de la répartition de ces écarts (sous un angle “probabilité d’avoir un niveau de MAE de moins de x secondes”) :



Ce graphique nous permet de constater 2 choses.

Tout d'abord, passé un seuil situé aux environs de 120 secondes, il n'y a plus de différence notable entre tous les modèles testés : la probabilité d'avoir des prédictions avec des niveaux de MAE situés entre 0 et 120 secondes est relativement proche entre tous les modèles (une analyse graphique plus fine, montre que cette probabilité se situe entre 86 et 87,8%).

Ensuite, nous confirmons que le modèle PassiveAggressiveRegressor est le modèle qui se situe juste derrière XGBoost, notamment en termes de probabilité de prédictions avec des MAE comprises entre 0 et 120 secondes. Notre première impression est confirmée : le modèle XGBoost étant en sur-apprentissage, nous retiendrons donc le modèle **PassiveAggressiveRegressor**, pour les raisons évoquées précédemment.

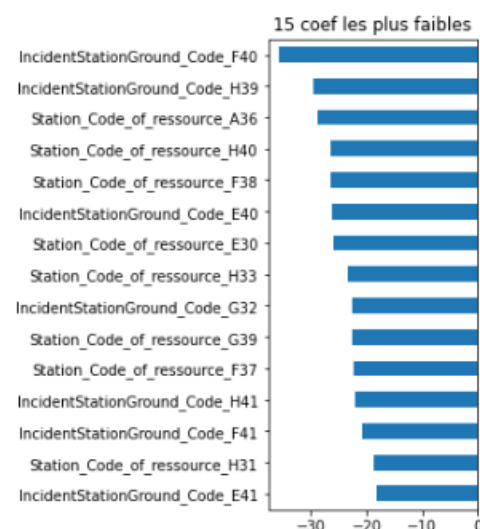
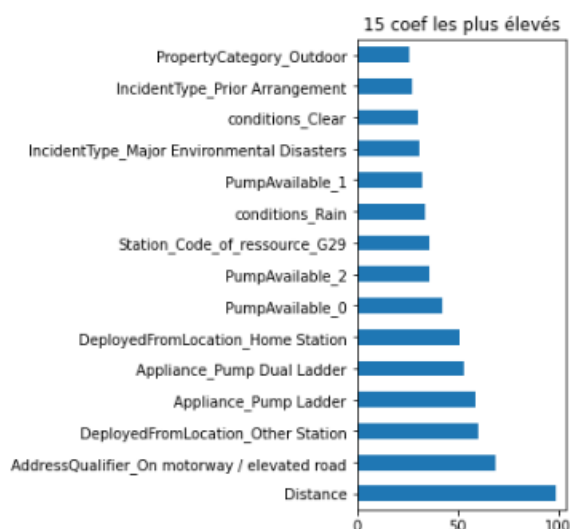
Ce modèle présente par ailleurs des importances de variables qui nous semblent cohérentes, tant d'un point de vue d'une analyse en fonction de leurs valeurs Shap, que des coefficients affectés à chaque variable (voir les graphiques ci-après).

Modèle : PassiveAggressiveRegressor

Valeur de Shapley

Expected Value: [325.09995681]





Toutefois, n'ayant pas trouvé de modèle en mesure de pallier directement aux limites identifiées précédemment (la difficulté à prédire les temps supérieurs à 600 secondes), nous avons alors poussé les travaux, pour pouvoir identifier les facteurs générant ces longs temps d'interventions, et ainsi envisager une "alerte" à mettre en place lors de la génération de prédictions, pour signaler les risques de sous-estimations des temps annoncés.

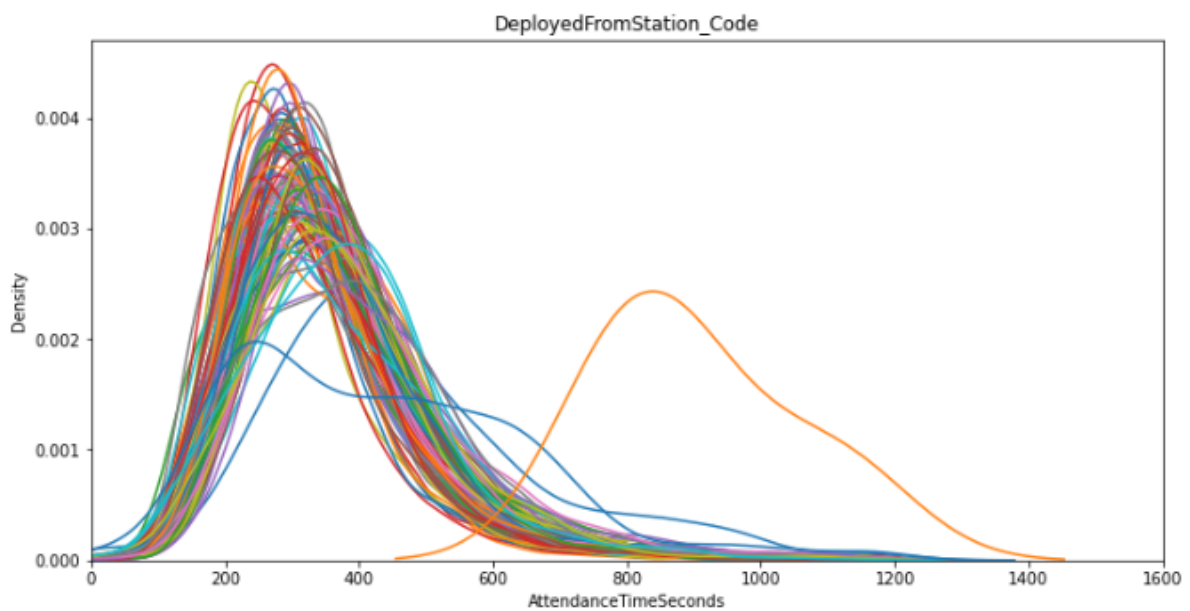
3.3.2.2 Approfondissement : prévenir les sous-estimations

Dans un premier temps, par une analyse graphique, nous avons essayé d'identifier les variables qui pouvaient être de nature à générer les temps de réponses les plus longs.

Au moyen des estimations de densités, nous avons observé que certaines modalités des variables `DeployedFromStation_Code` et `AddressQualifier` présentaient des densités différentes des autres.

Concernant `DeployedFromStation_Code` (variable que nous exploitons dans nos modèle, via la variable `Station_Code_of_ressource`), il nous faudra identifier les casernes dont les densités des temps d'interventions diffèrent de la plupart des autres modalités :

- une modalité (courbe jaune) en particulier, est complètement décalée vers des niveaux de temps plus élevés, avec un sommet situé aux environ de 825 secondes ;
- une autre modalité, en bleu, ne présente pas la forme en cloche, caractéristique de la plupart des modalités, mais une forme plus écrasée, "à 2 bosses" ;
- enfin, quelques modalités sont un peu plus décalées sur la droite du graphique que les autres.



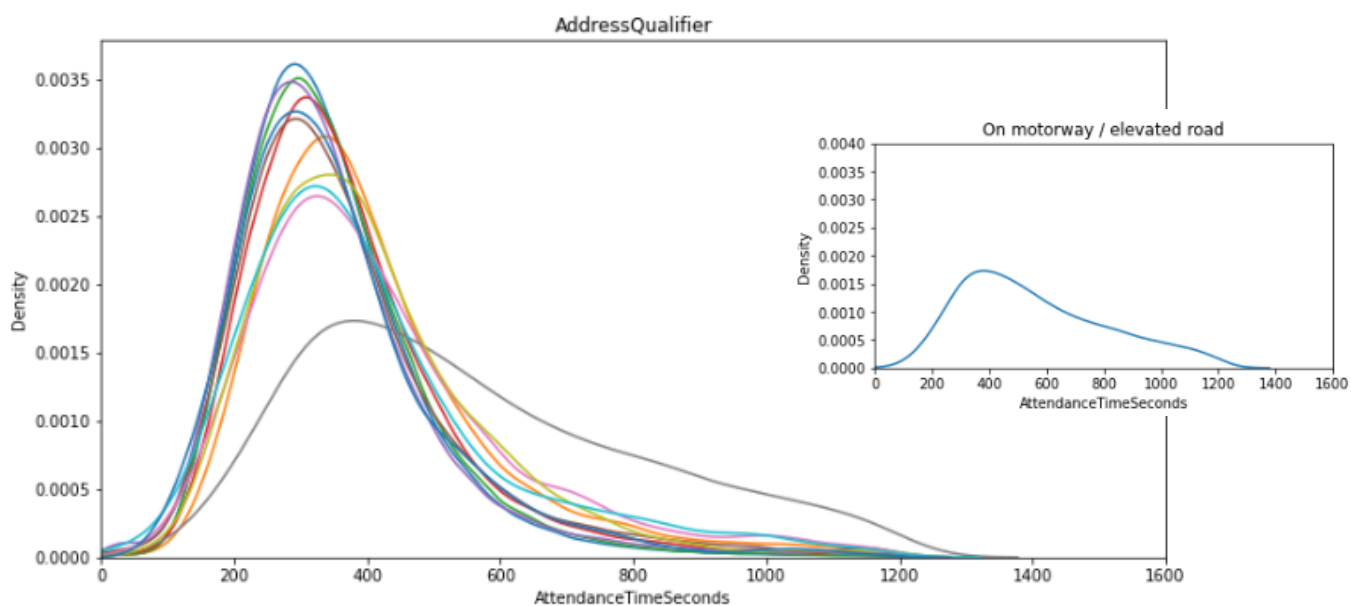
Pour identifier ces casernes, la méthode a été de calculer pour chacune d’elles, sur la base des courbes ci-dessus, la probabilité d’avoir des temps compris entre 600 et 1600 secondes, et de garder les casernes présentant ces probabilités les plus élevées (supérieures à 10%).

C’est ainsi un “top 11” des casernes (nom et codes) présentant les probabilités les plus élevées d’avoir des temps compris entre 600 et 1600 secondes, qui a été établi :

['Ruislip', 'Biggin Hill', 'Wennington', 'Hayes', 'Heathrow', 'Sidcup', 'Mill Hill', 'Orpington', 'East Greenwich', 'Wallington', 'Wembley']

Pour **AddressQualifier**, la tâche a été plus simple, une seule modalité étant différente des autres (courbe en gris sur le graphique qui suit).

Il s’agit de la modalité “On motorway / elevated road” (isolée dans le graphique superposé à droite).



En croisant ces informations (par identification des véhicules, du jeu de test, rentrant dans ces critères), avec les niveaux d'erreurs absolues calculés pour chacun des modèles que nous avons mis en oeuvre, nous confirmons l'analyse.

Le tableau ci-dessous présente les niveaux de MAE atteints sur le jeu de test, en regroupant les observations selon les critères identifiés précédemment :

	abs_err_xgb	abs_err_sgd	abs_err_hub	abs_err_par	abs_err_br	abs_err_lr	abs_err_mlpr
under_estimated							
0	66.0	68.0	66.0	67.0	69.0	69.0	71.0
1	80.0	83.0	80.0	80.0	83.0	83.0	84.0
2	185.0	185.0	181.0	180.0	185.0	185.0	186.0
3	204.0	205.0	185.0	180.0	204.0	205.0	180.0
<i>under_estimated</i>	<i>Que contient cette catégorie ?</i>			<i>Commentaire</i>			
3	MAE des véhicules d'une caserne du "top 10" ET en intervention sur "On motorway / elevated road"			MAE très élevée			
2	MAE des véhicules en intervention sur "On motorway / elevated road"			MAE très élevée			
1	MAE des véhicules d'une caserne du "top 10"			MAE élevée			
0	MAE des autres véhicules / type de lieux.			MAE en phase avec le modèle dans son ensemble			

Sur la base de ces résultats, l'idée a alors été d'envisager la mise en place d'une alerte à 2 niveaux :

- risque de sous-estimation significative par le modèle (si `AdressQualifier` = "On motorway / elevated road") ;
- risque de sous-estimation légère par le modèle (si `Station_Code_of_ressource` est une caserne du "top" des casernes présentant des temps élevés).

Mais en se référant aux coefficients que le modèle a attribué aux variables, ainsi qu'à une analyse plus fine des prédictions sur véhicules intervenant "On motorway / elevated road", nous constatons que :

- la variable présentant le second coefficient le plus élevé (valeur : 69) est précisément "AddressQualifier_On motorway / elevated road". Ce qui signifie que l'impact significatif de cette modalité de `AddressQualifier` est déjà pris en compte dans notre modèle, et que, par déduction, le niveau de MAE élevé que nous constatons semble davantage concerner des sur-estimations ; nous le confirmons effectivement par une analyse rapide :
 - Mean et Median AE sont respectivement de 256 et 212 secondes pour les seules prédictions montrant des sur-estimations ;
 - contre Mean et Median AE de 110 et 107 secondes pour les sous-estimations.
- les casernes du top présentant des temps de réponses habituellement élevés ont pour coefficient des valeurs comprises entre -6 et +18 ; nous pouvons raisonnablement penser que l'impact de ces variables est légèrement sous-estimé.

En conséquence, nous ne mettrons en place qu'une alerte sur les casernes présentant des temps habituellement plus élevés que les autres :

Si Station Code of ressource est une caserne du "top" des temps élevés
Alors risque de sous-estimation légère du temps par le modèle
(valeur 1 de la catégorie "under_estimated" du tableau précédent)

3.3.2.3 Rapprochement avec classification

Pour ces derniers travaux dans le cadre des modèles de régression, nous avons essayé de rapprocher les prédictions de ces modèles, sur le jeu de test, avec ceux de la classification précédente (véhicules arrivés ou non dans le délai de 360 secondes).

Pour cela, nous avons procédé à une conversion des prédictions des temps, en classes :

- 0 pour les temps prédits inférieurs ou égaux à 360 secondes ;
- 1 pour les temps prédits supérieurs à 360 secondes.

Nous avons alors réalisé une comparaison de tous les modèles de régression avec le meilleur modèle de classification (dont nous avons récupéré les prédictions au préalable), mais nous ne rentrerons pas ici dans le détail de cette comparaison, nous nous concentrerons sur le modèle PassiveAgressiveRegressor.

Voici tout d'abord le résumé de la comparaison des 3 informations (classes réelles vs classes de la classification vs classes de la régression) ; cela permet de constater que pour 73 % des observations, les classes sont identiques.

IncidentType			
class_AttendanceTimeSeconds	pred_classif	class_pred_par	
0	0	0	51345
	1	0	7484
		1	8241
1	0	0	8690
	1	0	5149
		1	28590

79935 observations aux classes identiques, soit 73.00 %

La même analyse a ensuite été réalisée pour ne comparer que la régression par rapport aux classes réelles ; nous arrivons à un taux de 79,84 % de classes correctement prédites :

IncidentType		
class_AttendanceTimeSeconds	class_pred_par	
0	0	58829
	1	8241
1	0	13839
	1	28590

87419 observations aux classes identiques, soit 79.84 %

Une autre synthèse très intéressante, a consisté à comparer simplement entre elles, classification et régression ; 88,46 % des prédictions sont identiques. On se rend ainsi compte d'un fonctionnement relativement proche des 2 modèles : un incident classifié 0 est prédit avec un temps inférieur ou égal à 360 secondes, et inversement, un véhicule dont le temps prédit sera supérieur à 360 secondes, est classifié 1 par la classification.

IncidentType					
pred_classif	class_pred_par		class_pred_par	0	1
0	0	60035	Prédictions classif		
1	0	12633	0	60035	0
	1	36831	1	12633	36831

96866 observations aux classes identiques, soit 88.46 %

Pour terminer, nous avons calculé un rapport de classification, sur les données issues du modèle de régression.

class_pred_par		0	1		
class_AttendanceTimeSeconds					
0		58829	8241		
1		13839	28590		
	precision	recall	f1-score	support	
0	0.81	0.88	0.84	67070	
1	0.78	0.67	0.72	42429	
accuracy			0.80	109499	
macro avg			0.79	0.78	0.78
weighted avg			0.80	0.80	0.80

En comparant ces scores à ceux issus de la classification, nous constatons que les prédictions issues du modèle de régression ne permettent pas d'atteindre un niveau de recall sur la classe 1 (les véhicules arrivant en plus de 360 secondes) comparable à celui de la classification (80 %).

Nous constatons par ailleurs, que dans ces 2 modèles (classification et régression), ce sont les variables Distance et AddressQualifier_On motorway / elevated road qui présentent les coefficients les plus élevés ; ce qui, avec la comparaison des 2 modèles entre eux, laisse supposer un fonctionnement relativement proche.

Pour conclure ce chapitre consacré aux travaux de régression, aucun modèle ne nous aura permis de réduire le niveau d'erreur des prédictions (apprécié par la MAE) à un niveau inférieur à 68 secondes ; bien évidemment, c'est un temps encore relativement élevé, compte tenu des enjeux humains.

Pour cette tâche, c'est le **modèle Passive Aggressive Regressor** que nous conserverons donc, en y apportant un **commentaire** dès lors qu'il s'agira de l'appliquer à un cas concret, quand il sera nécessaire d'avertir des risques de sous-estimation. Cela pourra se faire par la prise en compte "hors algorithme de machine learning", du fait que certaines casernes ont tendance à présenter des temps d'intervention plus élevés (voir les quelques exemples d'application à des incidents précis du jeu de test, en fin du notebook consacré à ce modèle).

4. Conclusion

Dans ce rapport, et après un nettoyage et un traitement des données, nous avons mis en place 3 modèles de Machine Learning afin d'améliorer le fonctionnement des pompiers de Londres :

- Un modèle de classification permettant de prédire le nombre de véhicules à envoyer
- Un modèle de classification prédisant si les véhicules d'intervention arriveront dans les délais objectifs des 360s
- Un modèle de régression permettant d'estimer le temps exact d'arrivée sur site des premiers secours.

Ces trois modèles de prédiction, distincts par leur méthode, peuvent être liés et utilisés successivement lors d'un appel d'urgence. On pourrait notamment imaginer la mise en place de deux nouveaux outils à disposition des opérateurs téléphoniques pour leur permettre de mieux gérer l'urgence et ainsi gagner sur le temps de mobilisation :

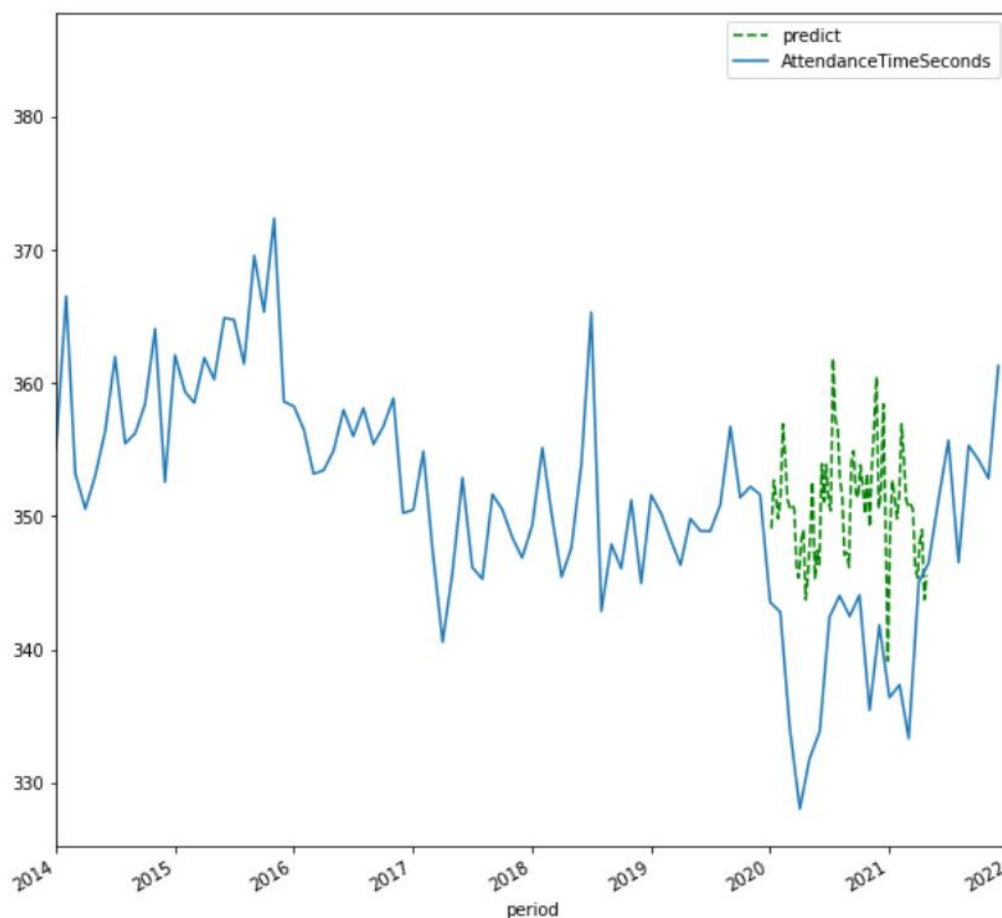
- une aide à la décision du nombre de camion à envoyer sur site en fonction du lieu et du type d'incident
- une prédiction du temps d'arrivée sur site des secours dans le but d'en informer la personne au téléphone et de prendre les meilleures dispositions en attendant l'arrivée des premiers secours.

Ces travaux ont été le fruit de nombreuses investigations, discussions et formulations d'hypothèses, tant les possibilités d'analyses qui nous étaient offertes étaient nombreuses, et le sujet pouvant rapidement nous emmener sur d'autres pistes très intéressantes à explorer.

Ainsi, il nous semble important ici, de signaler quelques limites et pistes d'améliorations de nos travaux :

- ces modèles ont été réalisés avec les données fournies par le site internet des pompiers de Londres ainsi que le rajout de quelques variables externes. Malheureusement, certaines données manquantes ou imprécises, pour lesquelles nous avons dû émettre des hypothèses, pourraient nous avoir limité dans les résultats obtenus par les différents modèles ;
- un apport de données relatives à l'organisation des brigades, des conditions d'astreintes du personnel, ou encore des distances exactes à parcourir en fonction du point de départ réel des véhicules, nous auraient sans doute permis d'atteindre de meilleurs résultats ;
- nous noterons aussi que l'apport des variables externes que nous avons identifiées, a eu un effet plutôt réduit dans les analyses liées au temps de réponses. Nous n'avons amélioré que très légèrement les performances. Pour exemple, les véhicules de pompiers étant prioritaires et les distances faibles, les conditions de trafic ou météorologiques n'impactent finalement que peu le temps de réponse ;

- un autre axe qui n'a pas été étudié ici, et qui pourrait faire l'objet d'un plan d'action dédié dans l'optique d'améliorer un peu plus le temps d'intervention et notamment la disponibilité des camions, serait de travailler sur les fausses alarmes. On pourrait imaginer créer un modèle de détection de ces fausses alarmes et ainsi éviter des déplacements inutiles ;
- d'autre part, nous avons pris le parti de travailler sur le temps complet d'arrivée sur site, qui pour mémoire, se compose du temps de mobilisation (des équipes / du matériel), et du temps de trajet. En effet, il nous était délicat de travailler sur le temps de mobilisation isolément car celui-ci dépend de plusieurs facteurs non renseignés dans les bases de données (organisation de la caserne, état des équipes avant l'évènement, matériels prêts dans le camion ou à embarquer dans le camion etc...).
Dans la perspective d'une optimisation des temps de réponses de la brigade des pompiers de Londres, l'idée pourrait être de travailler sur chacun des 2 temps, pour identifier précisément les variables qui impactent l'un et l'autre (et ainsi essayer de réduire le temps de mobilisation) ;
- pour terminer, nous avons remarqué un impact important de la crise "Covid" en 2020 sur le temps d'intervention. Un axe de travail, que nous n'avons pas choisi, pourrait être de tenter de corriger précisément cet impact sur chaque véhicule avant de construire nos modèles.
Ci-dessous un graphique présentant une correction des temps hebdomadaire moyen avec la fonction SARIMAX :



5. Annexes

Annexe 01 : variables du fichier LFB Incident data (1/2)

Fichier :	LFB Incident data Last 3 years.csv		
Colonne	Groupe d'information	Détail de la nature d'information	Nb modalités
IncidentNumber	Identifiant incident	Identifiant de l'incident	
DateOfCall	Date de l'appel	Date complète de réception de l'appel	
CalYear		Année de réception de l'appel	
TimeOfCall		Heure précise (à la seconde) de réception de l'appel	
HourOfCall		Heure de réception de l'appel	
IncidentGroup	Classification de l'incident	Type d'incident	3
StopCodeDescription		Détail du type d'incident	10
SpecialServiceType		Niveau de détail plus avancé pour les incident de type "Special Service"	22
19 variables	Lieu de l'incident	Détaillées sur la page suivante	
FRS	Caserne du lieu d'incident	Ville dont dépend la caserne des pompiers concernés	1
IncidentStationGround		Caserne dont dépend le lieu de l'incident	103
FirstPumpArriving_AttendanceTime	2 premiers véhicules mobilisés sur l'incident	Temps de réponse opérationnelle (en sec.) du 1er véhicule = heure arrivée sur les lieux - heure début mobilisation	
FirstPumpArriving_DeployedFromStation		Caserne de provenance du 1er véhicule	104
SecondPumpArriving_AttendanceTime		Temps de réponse opérationnelle (en sec.) du 2ème véhicule = heure arrivée sur les lieux - heure début mobilisation	
SecondPumpArriving_DeployedFromStation		Caserne de provenance du 2ème véhicule	104
NumStationsWithPumpsAttending	Comptage des véhicules et coût théorique	Nombre de casernes qui ont envoyé des véhicules sur l'incident	11
NumPumpsAttending		Nombre de véhicules envoyés sur l'incident	13
PumpCount		"Comptage des véhicules" mais incohérences (ne pas tenir compte)	79
PumpHoursRoundUp		Temps passé (heures) par les véhicules sur l'incident (arrondi à l'heure sup.)	
Notional Cost (£)		Coût théorique (en GBP) du temps passé sur l'incident	

Annexe 01 : variables du fichier LFB Incident data (2/2)

Fichier : LFB Incident data Last 3 years.csv (suite)			
Colonne	Groupe d'information	Détail de la nature d'information	Nb modalités
PropertyCategory	Lieu de l'incident	Catégorie du lieu de l'incident	9
PropertyType		Détail du lieu de l'incident	283
AddressQualifier		Localisation de l'incident par rapport à l'adresse enregistrée	11
Postcode_full		Code postal	
Postcode_district		Code postal district	
UPRN		Unique Property Reference Number	
USRN		Unique Street Reference Number	
IncGeo_BoroughCode		Code arrondissement	
IncGeo_BoroughName		NOM arrondissement (en majuscules)	
ProperCase		Nom arrondissement	
IncGeo_WardCode		Code quartier	
IncGeo_WardName		Nom quartier	
IncGeo_WardNameNew		Nom quartier nouveau (identique à colonne précédente)	
Easting_m		Latitude (OSGB 1936 / British National Grid)	
Northing_m		Longitude (OSGB 1936 / British National Grid)	
Easting_rounded		Latitude (OSGB 1936 / British National Grid) arrondie (50)	
Northing_rounded		Longitude (OSGB 1936 / British National Grid) arrondie (50)	
Latitude		Latitude (GPS / WGS84)	
Longitude		Longitude (GPS / WGS84)	

Annexe 02 : variables du fichier LFB Mobilisation data

Fichier :	LFB Mobilisation data Last 3 years.csv		
Colonne	Groupe d'information	Détail de la nature d'information	Nb modalités
IncidentNumber	Identifiant incident	Identifiant de l'incident	
CalYear	Date de l'appel	Année	
HourOfCall		Heure	
ResourceMobilisationId	Ressources mobilisées	Identifiant de mobilisation (probablement un n° d'ordre d'intervention)	589390
Resource_Code		Code ressources	142
PerformanceReporting	Performance reporting	Utilisé pour reporter les véhicules 1 et 2 dans la table incident	3
DateAndTimeMobilised	Timing de l'opération par véhicule	Heure de mobilisation (très proche de l'heure d'appel du tableau incident)	
DateAndTimeMobile		Heure départ de caserne	
DateAndTimeArrived		Heure arrivée lieu incident	
TurnoutTimeSeconds		Temps de mobilisation (en sec.) = heure de départ en intervention du véhicule - heure début mobilisation	
TravelTimeSeconds		Temps de trajet (en sec.) = heure arrivée sur les lieux - heure départ en intervention	
AttendanceTimeSeconds		Temps de réponse opérationnelle (en sec.) = TurnoutTimeSeconds + TravelTimeSeconds = heure arrivée sur les lieux - heure début mobilisation	
DateAndTimeLeft		Date et heure du départ des lieux de l'incident	
DateAndTimeReturned		Date et heure de retour à la caserne (non utilisé)	1
DeployedFromStation_Code		Code de caserne de départ du véhicule	109
DeployedFromStation_Name		Nom de caserne de départ du véhicule	109
DeployedFromLocation		Véhicule envoyé de sa caserne de base / d'une autre caserne	3
PumpOrder		Ordre d'arrivée des véhicules envoyés sur l'incident	13
PlusCode_Code	PlusCode	Pas de sens pour l'analyse (valeur unique)	1
PlusCode_Description		Pas de sens pour l'analyse (valeur unique)	
DelayCodeId	Retard sur l'intervention	Code retard	11
DelayCode_Description		Description du retard	11

Annexe 03 : suppression de 29 variables jugées inutiles

Typologie	Explication	Variables supprimées
Données postales	10 variables permettant la localisation des incidents, en utilisant des données de type "postales" (adresses, quartiers, etc.) ont été écartées. Nous leur avons préféré les coordonnées GPS.	IncGeo_BoroughCode, IncGeo_BoroughName, IncGeo_WardCode, IncGeo_WardName, IncGeo_WardNameNew, Postcode_district, Postcode_full, ProperCase, UPRN, USRN
Données redondantes	<p>6 variables ont pu être supprimées car faisant de toute évidence double emploi avec d'autres.</p> <p>C'est en grande partie (5 variables) une conséquence de la fusion des 2 fichiers puisque pour mémoire, le fichier des incidents intègre des données relatives aux 2 premiers véhicules arrivés sur les lieux de l'incident (données provenant du fichier des mobilisations).</p> <p>Nous avons également supprimé une colonne d'une même variable, pour laquelle nous avons à la fois son identifiant, ainsi que son libellé.</p>	<p>PerformanceReporting, FirstPumpArriving_AttendanceTime, FirstPumpArriving_DeployedFromStation, SecondPumpArriving_AttendanceTime, SecondPumpArriving_DeployedFromStation</p> <p>DelayCodeId</p>
Modalités	<p>3 variables ne présentaient chacune, qu'une seule et unique modalité, et étaient donc sans intérêt (car n'apportant en réalité aucune information).</p> <p>A l'inverse, une des variables n'était qu'un identifiant unique de l'intervention du véhicule. Nous avons donc autant d'identifiants que de lignes dans notre base, ce qui ne présentait aucune utilité pour le projet.</p>	<p>FRS, PlusCode_Code, PlusCode_Description</p> <p>ResourceMobilisationId</p>
Données incomplètes	4 variables contenant des coordonnées géographiques présentaient de nombreuses valeurs manquantes. Sachant qu'il nous était impossible de compléter ces données, nous avons choisi de les supprimer.	Easting_m, Latitude, Longitude, Northing_m
Données remplacées	<p>Les 2 seules coordonnées GPS les plus complètes étaient des coordonnées arrondies (à la cinquantaine près), dans un système OSGB 1936 / British National Grid.</p> <p>Comme vu précédemment, nous avons procédé à leur conversion vers un format WGS 84, rendant dès lors les variables d'origine, inutiles.</p>	Easting_rounded, Northing_rounded
Hors périmètre	<p>2 variables qui nous semblent relatives à des calculs de coûts théoriques d'intervention.</p> <p>Nous les avons supprimées parce qu' d'une part, notre projet n'a aucune vocation à analyser des coûts ou des rentabilités, et d'autre part, parce que leur contenu, leur mode de calcul n'est pas connu.</p>	Notional Cost (£), PumpHoursRoundUp
Variable vide	1 variable qui n'est constituée que de valeurs manquantes (NaN).	DateAndTimeReturned

Annexe 04 : traitement des valeurs manquantes

Variable avec NaN	Commentaire et résolution
DateOfCall	<p>2317 NaN, que nous avons retrouvées également sur 14 autres variables (liées ou non à DateOfCall).</p> <p>Ces observations étant très incomplètes, nous avons fait le choix de leur suppression pure et simple.</p> <p>Pour ne pas conserver d'incidents incomplets (du point de vue des véhicules intervenus), et ainsi garder une base de données la plus cohérente possible, la méthode que nous avons retenue a été de supprimer les <u>incidents</u> concernés par les 2317 NaN, et non simplement les 2317 lignes.</p> <p>Au final, ce sont les observations de 1963 incidents qui ont été supprimées.</p>
DateAndTimeMobile TurnoutTimeSeconds TravelTimeSeconds DateAndTimeMobilised	<p>On s'est aperçu que lorsque DateAndTimeMobile était manquante (3803 lignes), alors TurnoutTimeSeconds et TravelTimeSeconds étaient également manquantes, mais en revanche, AttendanceTimeSeconds existait.</p> <p>La méthode a été de :</p> <ul style="list-style-type: none"> - calculer un TurnoutTimeSeconds théorique (proportion de AttendanceTimeSeconds) ; - recalculer : $TravelTimeSeconds = AttendanceTimeSeconds - TurnoutTimeSeconds$; - puis calculer : $DateAndTimeMobile = DateAndTimeMobilised + TurnoutTimeSeconds$. <p>Suite à quoi nous avons traité les dernières valeurs manquantes sur TravelTimeSeconds : 61 valeurs ont été corrigées (étaient dues à ce qui nous a semblé être des inversions dans la saisie des données : pour ces véhicules, les heures de départ de la caserne étaient supérieures aux heures d'arrivée sur les lieux de l'incident), et les incidents concernant 19 autres valeurs manquantes ont été supprimés.</p> <p>Enfin, sur les dernières valeurs manquantes de TurnoutTimeSeconds (6 observations), nous avons fait le choix de fixer la valeur à 0 (valeur qui existe par ailleurs dans d'autres observations).</p>
DeployedFromStation_Code DeployedFromStation_Name	<p>13 véhicules concernés ; nous avons choisi de considérer que les véhicules étaient partis de leur caserne d'appartenance (= les 3 premiers caractères de Resource_Code), puis de corriger DeployedFromLocation (valeur = Home Station) et DeployedFromStation_Name en conséquence.</p>
DeployedFromLocation	<p>il restait 324 valeurs manquantes.</p> <p>Cette variable se basant sur d'autres, maintenant complètes, nous avons pu procéder à leur calcul, selon la règle suivante :</p> <p>si le code caserne compris dans Resource_Code (3 premiers caractères) est égal à la variable DeployedFromStation_Code, alors DeployedFromLocation prend la modalité "Home Station", sinon "Other Station".</p>
SpecialServiceType	<p>461649 valeurs manquantes, dont :</p> <ul style="list-style-type: none"> - 4 qui sont effectivement des SpecialService, dont la description (StopCodeDescription) est "Use of Special Operations Room" : nous avons choisi de copier cette description dans la variable SpecialServiceType ; - 461645 interventions qui ne concernent pas des SpecialService, il est donc logique de n'avoir aucune valeur dans cette variable. Nous avons choisi de créer une modalité "Not Special Service" pour remplacer ces valeurs manquantes.
DelayCode_Description	<p>456496 NaN sur cette variable, qui sont parfaitement compréhensibles : ce sont des interventions sur lesquelles aucun retard n'a été constaté (et n'a donc besoin de justification).</p> <p>Nous avons fait le choix de remplacer ces valeurs manquantes par une modalité "No delay".</p>

Annexe 05 : jours de congés

LONDON BOROUGH OF BARKING AND DAGENHAM SCHOOL TERMS AND HOLIDAYS 2020/2021

The 5 non-contact days + 2 days holiday* are to be determined by individual schools.

*The 2 days are to be used at schools' discretion.

1 September 2020 = First Day of Autumn Term
23 July 2021 = Last Day of Summer Term

SEPTEMBER 2020

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

OCTOBER 2020

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

NOVEMBER 2020

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

DECEMBER 2020

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY 2021

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY 2021

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

MARCH 2021

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

APRIL 2021

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MAY 2021

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

JUNE 2021

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

JULY 2021

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

AUGUST 2021

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Autumn term: Tuesday 1 September 2020 - Friday 18 December 2020 = 74 school days

Spring term: Monday 4 January 2021 - Thursday 1 April 2021 = 59 school days

Summer term: Monday 19 April 2021 - Friday 23 July 2021 = 64 school days

SCHOOLS TO ENSURE THAT THERE ARE 190 TAUGHT DAYS. THIS LEAVES 5 NON-CONTACT DAYS AND 2 DAYS HOLIDAY TO BE DETERMINED BY SCHOOLS.

DATES OF THE 2 DAYS HOLIDAY AND 5 NON-CONTACT DAYS MUST BE AGREED AND PUBLISHED ON SCHOOLS' WEBSITES BY 1ST SEPTEMBER OF THE PRECEDING ACADEMIC YEAR.



Bank holidays





School holidays

Annexe 06 : densité du trafic londonien

(exemple des données pour l'année 2021)

WEEKLY TRAFFIC CONGESTION BY TIME OF DAY

What time was rush hour in London?

2021	2020		2019		 Best time to avoid		
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
12:00 AM	14%	7%	5%	6%	7%	8%	12%
	10%	3%	2%	3%	3%	5%	9%
02:00 AM	7%	1%	1%	1%	2%	3%	6%
	6%	0%	0%	0%	1%	1%	4%
04:00 AM	2%	0%	0%	0%	0%	0%	2%
	0%	1%	1%	1%	1%	1%	0%
06:00 AM	0%	16%	17%	17%	16%	14%	2%
	2%	40%	43%	42%	40%	34%	7%
08:00 AM	5%	48%	54%	53%	50%	43%	12%
	11%	34%	39%	38%	37%	34%	21%
10:00 AM	18%	30%	32%	33%	33%	34%	31%
	27%	32%	33%	34%	35%	40%	40%
12:00 PM	35%	34%	35%	36%	37%	45%	47%
	36%	34%	35%	36%	37%	46%	48%
02:00 PM	33%	36%	37%	38%	40%	51%	44%
	30%	43%	45%	47%	49%	62%	39%
04:00 PM	28%	44%	49%	51%	53%	 64%	36%
	28%	45%	50%	53%	55%	62%	34%
06:00 PM	27%	36%	40%	44%	46%	53%	32%
	23%	24%	26%	29%	31%	38%	28%
08:00 PM	19%	17%	18%	19%	21%	26%	23%
	14%	13%	14%	15%	16%	19%	19%
10:00 PM	12%	11%	13%	14%	16%	18%	18%
	10%	8%	10%	11%	13%	16%	17%

Annexe 07 : variables du fichier final

Fichier :	base_ml.pkl		
Colonne	Groupe d'information	Détail de la nature d'information	Nb modalités
IncidentNumber	Identifiant incident	Identifiant de l'incident	359880
PropertyCategory	Lieu de l'incident	Catégorie du lieu de l'incident	9
PropertyType		Détail du lieu de l'incident	282
AddressQualifier		Localisation de l'incident par rapport à l'adresse enregistrée	11
IncidentStationGround		Caserne dont dépend le lieu de l'incident	102
IncidentType	Classification de l'incident	Type d'incident	6
IncidentCategory		Catégorie d'incident	29
FalseAlarm		Identification des fausses alarmes	2
TotalOfPumpInLondon_Out	Force d'intervention des pompiers	Nombre total de véhicules en intervention sur Londres au moment de l'appel	
PumpByStation		Nombre total de véhicules dans la caserne dont dépend l'incident	
PumpOfIncidentStation_Out		Nombre de véhicules de la caserne dont dépend l'incident, déjà en intervention	
PumpAvailable		Nombre de véhicules disponibles dans la caserne dont dépend l'incident	
Resource_Code	Ressources mobilisées	Code de la ressource mobilisée (caserne d'appartenance + type de véhicule)	142
DeployedFromStation_Code		Caserne de base (code) du véhicule en intervention sur l'incident	102
DeployedFromStation_Name		Caserne de base (nom) du véhicule en intervention sur l'incident	102
DeployedFromLocation		Véhicule envoyé depuis sa caserne de base ou depuis une autre caserne	2
PumpOrder		Ordre d'arrivée des véhicules envoyés sur l'incident	9
Appliance		Type de véhicule envoyé	2
NumStationsWithPumpsAttending		Nombre de casernes qui ont envoyé des véhicules sur l'incident	
NumPumpsAttending		Nombre de véhicules envoyés sur l'incident	
Mobilised_Rank		Vague de mobilisation	1
Distance		Distance entre lieu de l'incident et la "Home Station" du véhicule mobilisé	
Station_Code_of_ressource	Timing de l'opération par véhicule	Caserne de base du véhicule en intervention sur l'incident	102
IncidentStationGround_Code		Caserne dont dépend le lieu de l'incident	102
DateAndTimeMobilised		Heure de mobilisation	
TurnoutTimeSeconds		Temps de mobilisation (en sec.) = heure de départ en intervention du véhicule - heure de mobilisation	
TravelTimeSeconds		Temps de trajet (en sec.) = heure arrivée sur les lieux - heure départ en intervention	
AttendanceTimeSeconds		Temps de réponse opérationnelle (en sec.) = TurnoutTimeSeconds + TravelTimeSeconds = heure arrivée sur les lieux - heure de mobilisation	
DelayCode_Description	Retard sur l'intervention	Description du retard constaté	11
temp	Données météorologiques	Température à l'heure de l'incident	
precip		Pluviométrie à l'heure de l'incident	
cloudcover		Couverture nuageuse à l'heure de l'incident	
visibility		Visibilité à l'heure de l'incident	
conditions		Conditions	7
icon		Conditions (plus de détail)	9
workingday	Vacances	Jours fériés	2
school_holidays		Jours de congés (scolaires)	2
congestion_rate	Trafic routier	Densité du trafic (moyenne annuelle par heure / jour de la semaine)	
year	Variable temporelle	Année de l'incident (basée sur date et heure de mobilisation)	
month		Mois de l'incident (basée sur date et heure de mobilisation)	
day		Jour de l'incident (basée sur date et heure de mobilisation)	
weekday		Jour de la semaine de l'incident (basée sur date et heure de mobilisation)	
hour		Heure de l'incident (basée sur date et heure de mobilisation)	