



PRÉDICTION DE LA QUALITÉ DE L'AIR

Rapport de projet

ALEXANDRE VEREPT, MOHAMED
BOULANOUAR, MAXIME THOOR

ISEN
ALL IS DIGITAL!
LILLE



PRÉFACE



Après avoir réalisé un premier projet lors du semestre 1 sur le thème de la qualité de l'air, l'ensemble de notre équipe, accompagné du doctorant Kevin Hérissé, a choisi de prolonger l'expérience.

Afin de suivre au mieux la construction du projet, nous vous invitons à aller sur le lien ci-dessous qui mène à notre GitHub, lieu de partage de notre travail.

<https://github.com/AlexandreVerept/AirIQ2>



SOMMAIRE

I - Démarche réalisée

- A. Recherches scientifiques
- B. Modèle choisi et améliorations
- C. Présentation des résultats

II - Réalisation technique

- A. Architecture
- B. Produit fini
- C. Améliorations

III - Gestion de projet

- A. Phase de démarrage
- B. Planification
- C. Exécution



INTRODUCTION

La qualité de l'air est probablement l'une des inquiétudes environnementales et sanitaires majeures de ce nouveau siècle. Aujourd'hui, nous respirons environ 15000 litres d'air par jour et la présence des polluants dans l'air devient de plus en plus alarmante.

Ainsi, savoir déterminer l'indice de qualité de l'air dans la métropole de Lille est un défi important dans un contexte de changement climatique. Pour cela, nous allons à nouveau tenter de prédire cet indicateur à partir d'un modèle de Machine Learning.

Après avoir réussi à prédire l'indice de qualité de l'air à J+1 avec 60% de précision et avec uniquement deux paramètres (la température et l'humidité) lors du projet du premier semestre, nous allons à présent chercher à déterminer cet indice mais cette fois-ci à J+2.

I - DÉMARCHE RÉALISÉE

Aujourd'hui, le calcul de l'indice de qualité de l'air est obligatoire pour les villes de plus de 100 000 habitants ; il s'agit de « l'indice ATMO ». Cependant, il en existe un pour les villes de moins de 100 000 habitants qui prend le nom de « indice de la qualité de l'air simplifié ».

L'indice de qualité de l'air varie entre 1 (très bon) et 10 (très mauvais). Il est calculé par l'observatoire agréé ATMO et est composé de 4 sous-indices, chacun étant représentatif d'un polluant de l'air : particules fines (PM10), ozone (O3), dioxyde d'azote (NO2) et dioxyde de soufre (SO2). L'indice du jour sera donc l'indice le plus élevé des 4 sous-indices.

Le calcul de cet indice est basé sur des stations de fond, c'est-à-dire des stations urbaines et périurbaines situées sur la zone géographique correspondante. Il ne prend pas en compte les stations de mesure le long du trafic (automobile et industrielle). Le tableau suivant indique les seuils réglementaires par polluant. Pour sa prédiction, l'ATMO se base sur ce tableau :

Valeurs réglementaires (concentrations en µg/m3)				
Indice	Poussières en suspension PM10	Dioxyde d'azote NO ₂	Ozone O ₃	Dioxyde de Soufre SO ₂
10 - très mauvais	80+	400+	240+	500+
9 - mauvais	65-79	275-399	210-239	400-499
8 - mauvais	50-64	200-274	180-209	300-399
7 - médiocre	42-49	165-199	150-179	250-299
6 - médiocre	35-41	135-164	130-149	200-249
5 - moyen	28-34	110-134	105-129	160-199
4 - bon	21-27	85-109	80-104	120-159
3 - bon	14-20	55-84	55-79	80-119
2 - très bon	07-13	30-54	30-54	40-79
1 - très bon	0-6	0-29	0-29	0-39

Figure : signification des différents niveaux d'indices de qualité de l'air

Dans son calcul, ATMO prévoit les indices du jour même et du lendemain. Pour obtenir un résultat assez précis, cet observatoire se base sur les modèles de prévisions de la qualité de l'air, les données météorologiques et les résultats des différentes stations. Il faut donc un certain nombre de paramètres et d'outils de précisions pour pouvoir effectuer une prédiction précise et fiable.

Ici, le but de notre modèle est de réussir à prédire la qualité de l'air du lendemain, du surlendemain et à J+3. Pour ce projet, nous travaillons avec la Métropole Européenne Lilloise (MEL), nous les avons rencontré afin de définir précisément les différents seuils de précision. Actuellement, l'indice de qualité de l'air ATMO est précis à 70%, la MEL nous a fixé pour objectif d'atteindre une précision équivalente pour J+1, J+2 et J+3.

Pour ce faire, nous avons à notre disposition de nombreuses bases de données en ligne. Avec ce type de données et de modèle, il est évident que nous ne pouvons pas prédire l'indice de qualité de l'air avec 100 % de précision, mais nous devons donner une approximation suffisamment fiable pour pouvoir être utilisée de manière concrète. L'intérêt pour les collectivités locales comme la MEL d'une prédiction à plus long terme est l'anticipation des pics de pollution, qui sont souvent accompagnés de mesures de circulation alternée, de gratuité des transports en commun. Cependant, ce sont des mesures difficiles à mettre en place dans l'urgence, d'où le besoin de les prévoir le plus tôt possible.

A. Recherches scientifiques



Présentation des recherches sur les différents paramètres

Dans un premier temps, nous avons débuté par des recherches concernant notre dataset. La première question que nous nous sommes posée est la suivante : Quels sont les paramètres qui ont un réel impact sur la prédiction de la qualité de l'air ?

Après diverses recherches sur le sujet, nous avons conclu qu'il y a plusieurs facteurs importants pour que la prédiction soit la plus réaliste et la plus fiable possible. Ainsi, l'humidité, la température, la pression, le vent (direction et force) et certains polluants sont des éléments majeurs et indispensables pour évaluer la qualité de l'air. Tous ces paramètres ont été évalué et visualisé sur R Studio afin de tirer des conclusions sur leur influence dans le calcul. Cette visualisation est présente en annexe à partir de la page 42 jusque la page 47.

Pour notre modèle, nous avons choisi d'utiliser 3 datasets car chacun d'entre eux contient des informations pertinentes. Ils sont également disponibles en temps réel ou presque, ce qui est nécessaire pour fournir une prédiction tous les jours. De plus, ils ont notamment l'avantage d'avoir un historique de mesure plutôt long, important pour l'entraînement de notre modèle.

Le tableau ci-dessous regroupe ainsi l'ensemble des datasets utilisés avec leurs caractéristiques :

Nom	Source	Description	Fréquence	Fenêtre de disponibilité
<i>Indice qualité de l'air</i>	MEL	Indice de qualité de l'air	Chaque jour	Fenêtre de 5 ans (dataset envoyé par la MEL + données publiques en ligne)
<i>Données SYNOP Essentielles OMM</i>	Météo France	Large spectre de données météo : humidité, température, pression...	Toutes les 3h	Depuis 1997
<i>Historique de l'indice Atmo</i>	ATMO	Mesures quotidiennes des principaux polluants : NO2, O3, PM10	Chaque jour	Depuis 2012

Figure : Datasets utilisés



Présentation des recherches sur les modèles mathématiques

Par la suite, nous avons étudié les différents modèles mathématiques qui correspondent à notre problème, à savoir la prédiction de séries temporelles multivariées (Multivariate Time Series Forecasting).

Quelques définitions sont nécessaires à la compréhension des caractéristiques de notre série temporelle :

Une **série temporelle** est une liste de dates, dont chacune est associée à une ou plusieurs valeurs. Les séries temporelles sont une manière structurée de représenter des données. Visuellement, il s'agit d'une ou plusieurs courbes qui évoluent au fil du temps. Dans notre cas, la série est **multivariée** car nous utilisons plusieurs variables interdépendantes. Notre série comporte des **saisonnalités**, c'est-à-dire qu'elle a une variation cyclique prévisible dépendant de la période (par exemple l'été et l'hiver n'ont pas du tout les mêmes profils de températures et d'humidité).



Figure : Exemple de séries temporelles avec une saisonnalité

Nous faisons l'hypothèse simplificatrice que notre série est **globalement stationnaire**, ce qui signifie que les espérances, covariances et variances mathématiques sont indépendantes du temps. Pour faire simple, cette supposition nous mène à considérer que les mesures physiques (températures, humidité etc.) à Lille sont globalement constantes d'une saison sur l'autre. Il est à noter que ceci pourrait poser des problèmes de visualisation à l'avenir lors d'années exceptionnelles ou sur le long terme avec le changement climatique.

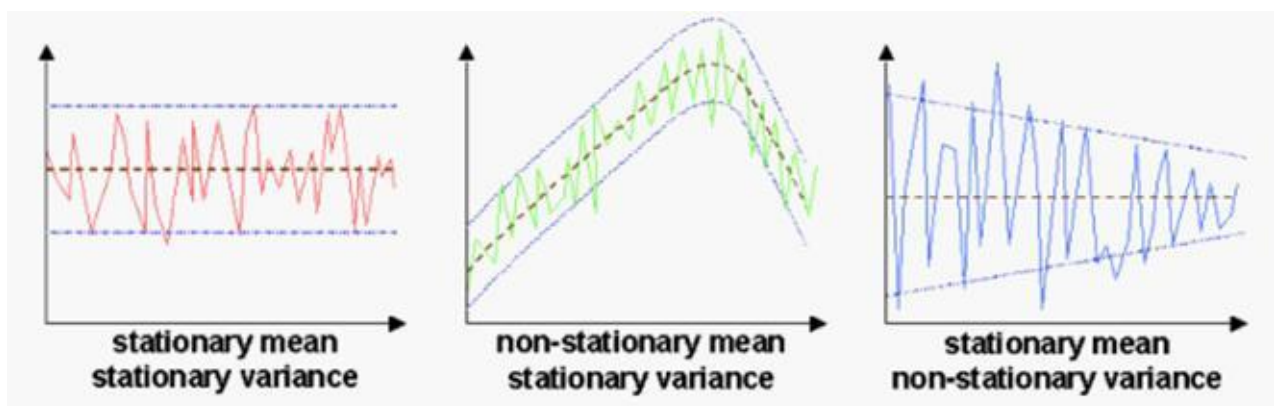


Figure : illustration des différentes caractéristiques de séries non-stationnaires

De gauche à droite, on peut voir : une série dont la moyenne varie au cours du temps, une série dont la variance varie au cours du temps, une série dont la covariance varie au cours du temps.

Nous avons effectué des recherches sur les modèles mathématiques qui pouvaient répondre à notre besoin en respectant toutes nos hypothèses. Cependant, aucun de nous n'a trouvé une modélisation vraiment pertinente (voir en annexe les résultats de nos recherches sur les différents modèles), mais la plus proche était le lissage exponentiel de Holt Winter ou HWES, malheureusement réservée aux études uni-variées.

Suite à nos recherches personnelles et aux différents articles consultés sur le sujet, nous avons conclu que l'utilisation de réseaux de neurones récurrents était la meilleure option pour étudier ce genre de problèmes. Le principe de ces réseaux récurrents est simple : pour calculer le résultat en sortie, la couche de neurones utilise les résultats précédents en plus des données qui lui sont entrées. Une fois la séquence en entrée complète, la couche renvoie un résultat de sortie unique.

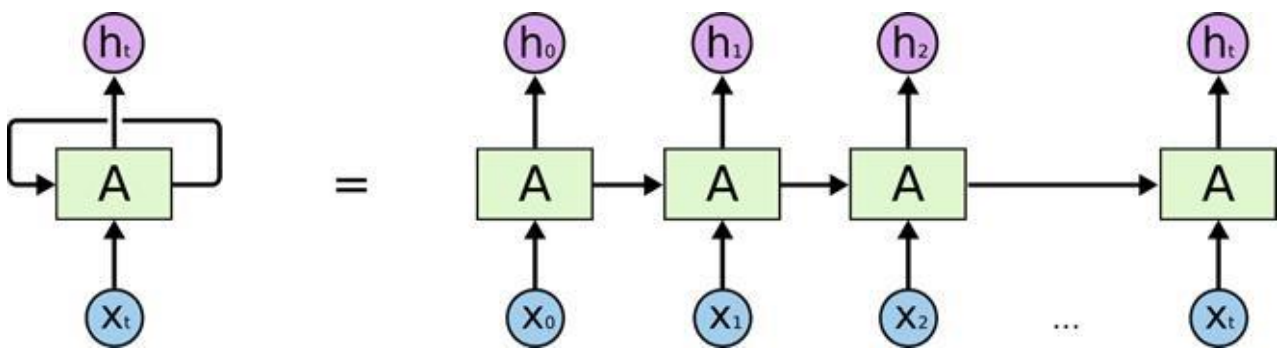
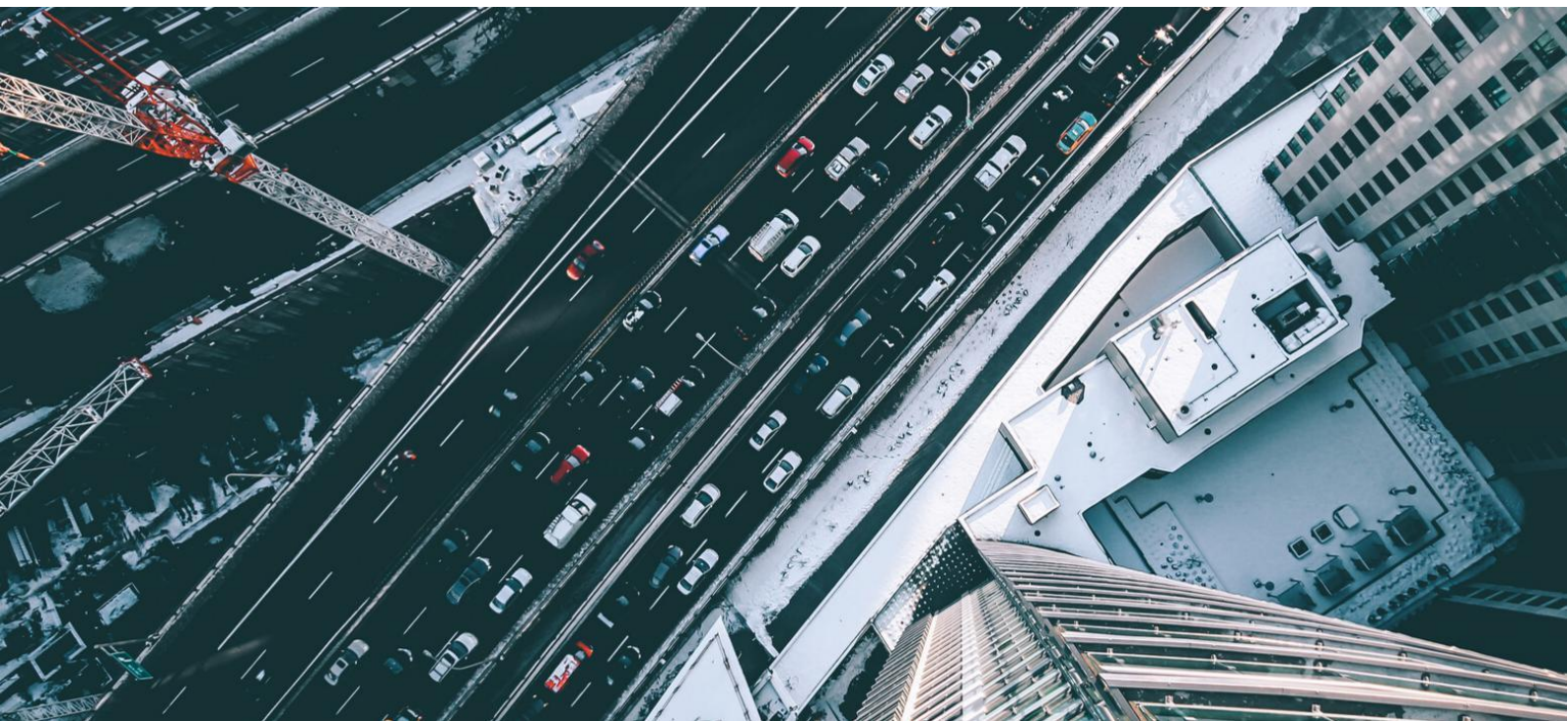


Figure : Explication du fonctionnement d'un réseau récurrent



B. Modèle choisi et améliorations

● Le choix du modèle récurrent

Il existe trois grandes catégories de réseaux récurrents. D'entrée, nous avons éliminé le RNN simple (Recurrent Neural Network), à cause de son problème du Vanishing et Exploding Gradient. En effet, le gradient utilisé lors de l'entraînement devient trop faible ou trop grand, ce qui empêche la mise à jour des poids du modèle, ou alors le changement étant trop grand le modèle ne converge pas vers une solution optimale maximisant la précision.

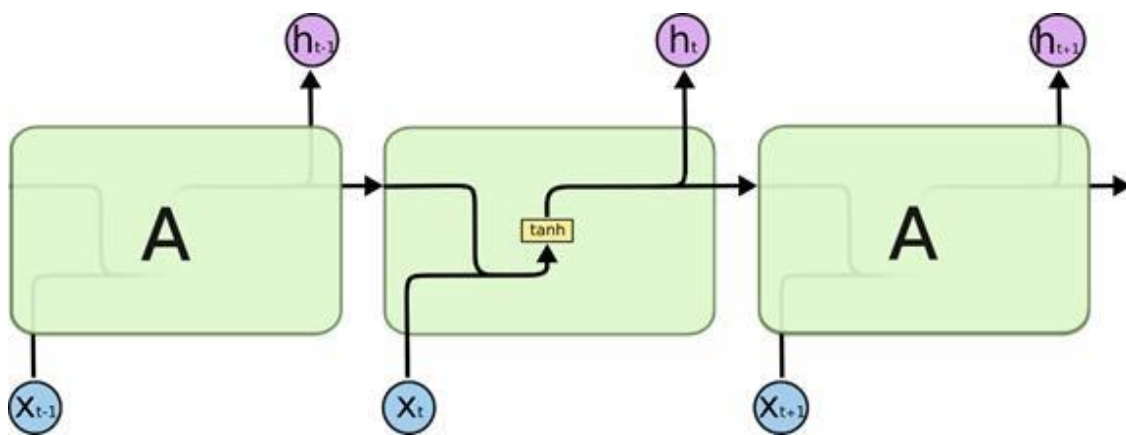


Figure : Réseau RNN

Deux possibilités s'offraient alors à nous, d'un côté la LSTM (Long Short-Term Memory) que nous avons déjà pu utiliser au premier semestre lors de notre première prédiction à $J+1$, et de l'autre un modèle plus moderne qui fonctionne sur le même principe, le GRU (Gated Recurrent Units).

Voici un aperçu rapide de la structure d'une LSTM et d'un GRU (la différence majeure réside dans les portes utilisées mais le principe de fonctionnement est identique) :

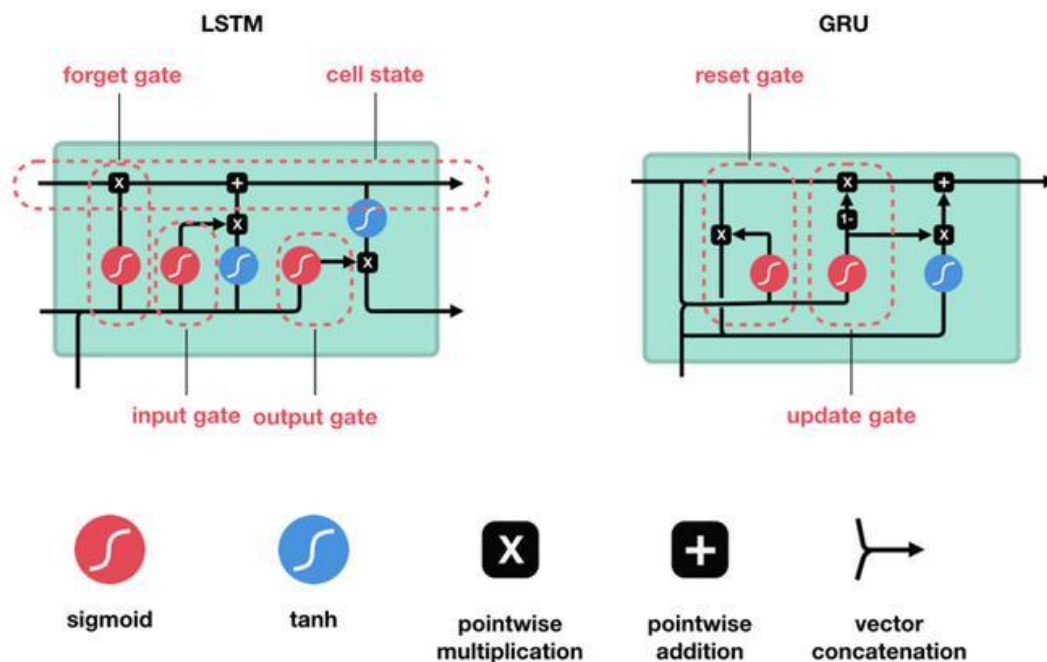
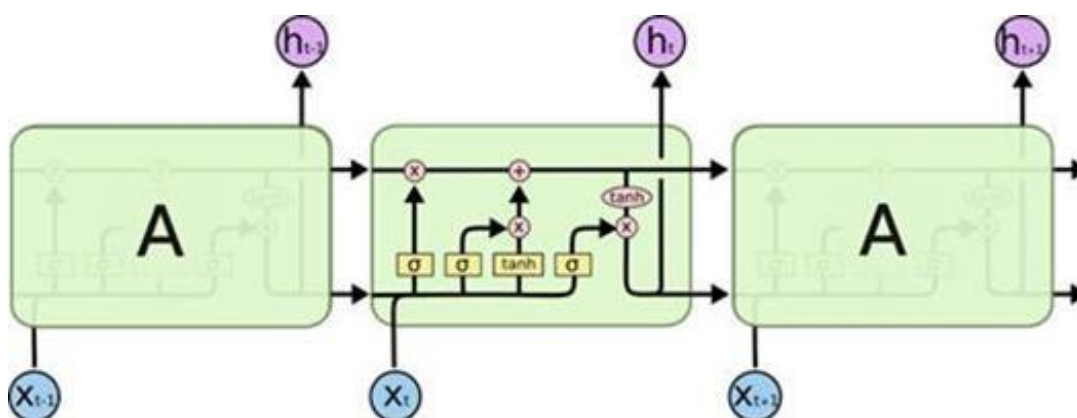
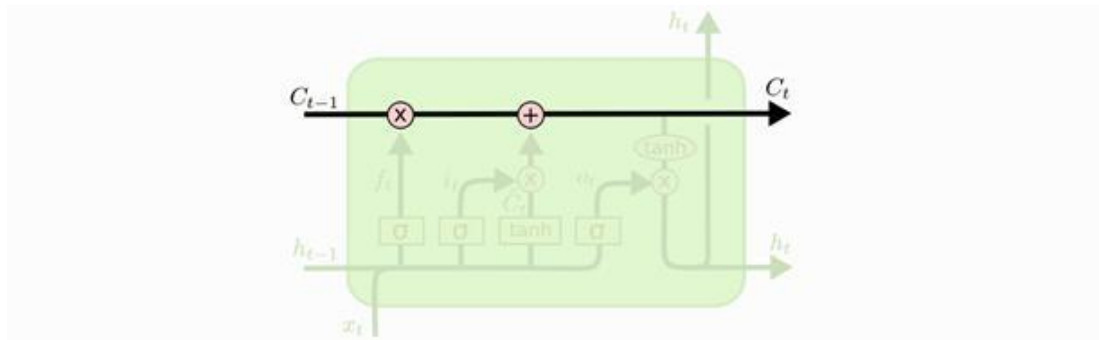


Figure : Structure d'une cellule de LSTM et de GRU

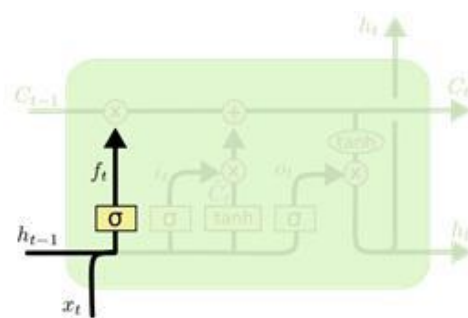
● LSTM



La clé des LSTM (également valable pour le GRU) est l'état de la cellule (ou état caché), et la ligne horizontale passant par le haut du diagramme. En effet, l'état de la cellule est mis à jour par différentes portes, elles-mêmes influencées par les données d'entrées et la sortie de la LSTM lors de l'itération précédente. Ceci permet d'utiliser les informations d'une prédiction pour améliorer la suivante, et donc de pouvoir gérer des informations avec une dépendance temporelle.

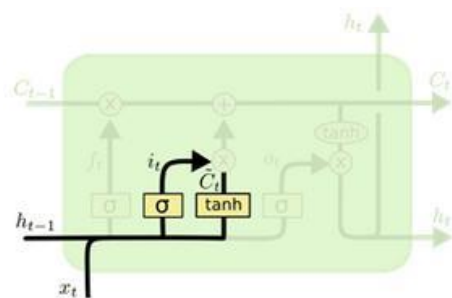


Chaque cellule est composée de plusieurs portes, avec chacune une opération mathématique différente. La première porte est une « forget gate », qui sélectionne les informations à garder venant de la cellule précédente à l'aide d'une fonction sigmoïde.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

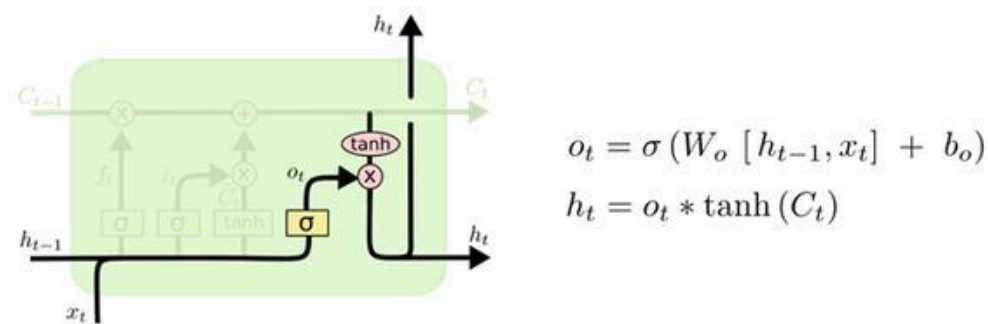
L'opération suivante met à jour le nouvel état de la cellule avec les informations sélectionnées lors des étapes précédentes.



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Enfin, nous utilisons une sigmoïde et une fonction tangente hyperbolique pour choisir ce qui sera sélectionné dans la sortie de la cellule.



GRU

Le principe global du GRU est identique à la LSTM mais avec une entrée pour chaque terme de la séquence, un état de cellule, et une sortie en bout de chaîne, mais les portes diffèrent légèrement :

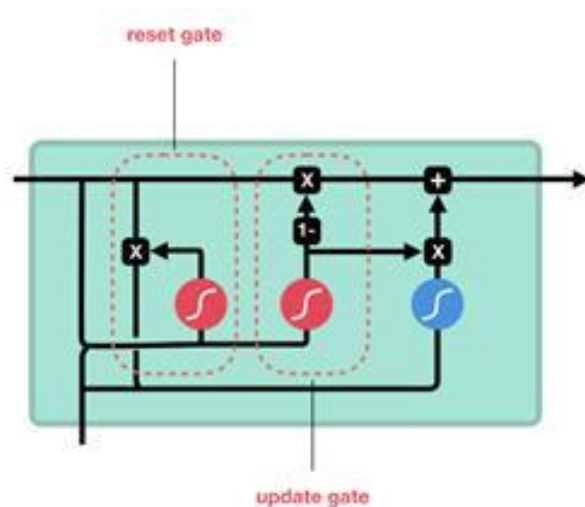


Figure : Structure du GRU

L'**update gate** agit de manière similaire à la porte d'oubli et d'entrée d'une LSTM. Elle décide des informations à conserver et de celles à oublier.

La **reset gate** est une autre porte utilisée pour décider de la quantité d'informations passées à oublier.

L'état caché précédent est combiné avec l'entrée du réseau et normalisé (par une tangente hyperbolique). Par la suite, les données jugées superflues vont être annulées grâce à la sortie de l'update gate. On y ajoute les coordonnées de l'état caché précédent jugées « inutiles » (en ayant, cette fois, annulé toutes les coordonnées pertinentes).

Conclusion sur le choix

Après différentes recherches et comparaisons, nous avons choisi de privilégier le modèle du GRU. En effet, ce dernier nous fournit des résultats équivalents à la LSTM, mais avec près d'un tiers de temps en moins. Cela s'explique par le fait que la GRU possède deux portes contre trois pour la LSTM (en plus de la sortie).

Dans notre cas précis, le GRU ne comporte que des avantages.

Améliorations de notre modèle

En plus des grands concepts que nous venons d'aborder, nous avons également mis en place certaines astuces pour tenter d'améliorer nos résultats qui ne méritent pas une partie à part entière mais que nous allons présenter rapidement ici.



Le k-fold et la cross-validation

Le K-fold est utilisé lors de l'entraînement de notre modèle : nous découpons notre dataset d'entraînement en k morceaux, qui seront utilisés à tour de rôle en validation, comme l'illustre le schéma suivant :

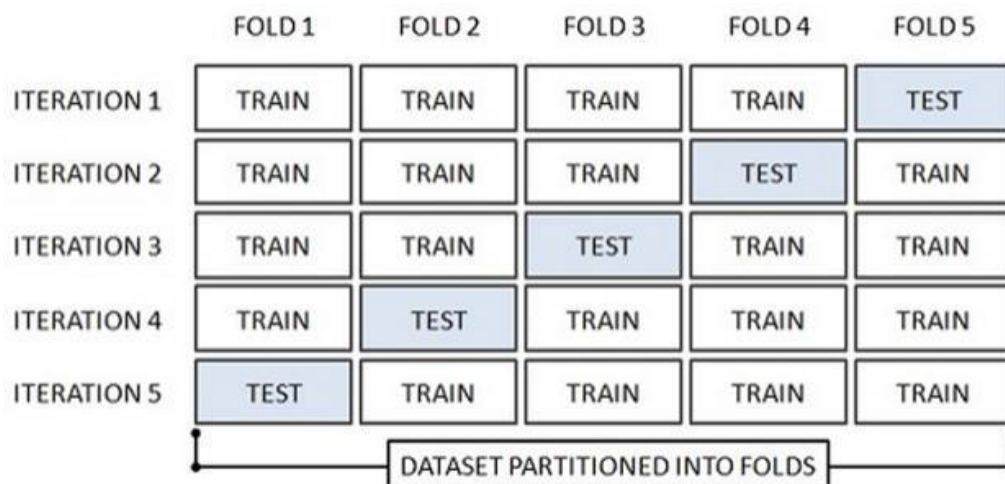


Figure : Structure d'utilisation du k-fold

Ce processus est nommé validation croisée (ou cross-validation) et permet de diminuer l'overfitting.

Training, test et validation

Afin de valider nos résultats sur nos différents modèles, nous utilisons un dataset de test en plus de la cross-validation. Ainsi notre dataset d'environ 3 ans est découpé en 80% pour le K-fold et 20% pour le test.

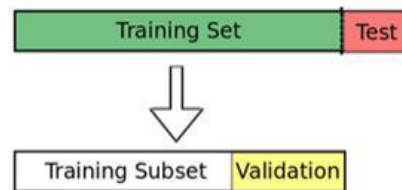


Figure : Découpage en training, test et validation

Classification ou régression

Dans ce projet, l'objectif en sortie est d'obtenir l'indice de qualité de l'air, soit un entier de 1 à 10. Deux options s'offraient donc à nous.

La première était de réaliser un modèle qui effectue une régression afin d'obtenir un réel entre 1 et 10 et arrondir à l'entier le plus proche. Au premier semestre, c'est ce que nous avons utilisé lors de notre projet.

La seconde option, celle que nous avons retenue, consiste à réaliser un modèle de classification avec en sortie un vecteur probabilité pour chacun des entiers entre 1 et 10. Cette seconde option nous permet d'avoir de bien meilleurs résultats expérimentaux, dus au moins en partie à la complexité des relations entre chacune des données que nous utilisons.

Le fine tuning

Cette méthode consiste à n'entraîner qu'une partie du réseau en gelant la mise à jour de certains poids lors de l'entraînement de notre réseau. Cela peut nous aider à augmenter la précision des différentes sorties indépendamment les unes des autres. Nous n'avons pas eu le temps de mettre en place tout ce que nous voulions faire avec cette idée mais c'est une piste d'amélioration.

Validation des résultats

Nous avons fait de nombreux tests, c'est pourquoi nous avons dû recourir à des outils de visualisations tels que Tensorboard ou des notebooks réalisés par nos soins afin de présenter les résultats automatiquement.

Exemples de visualisations :

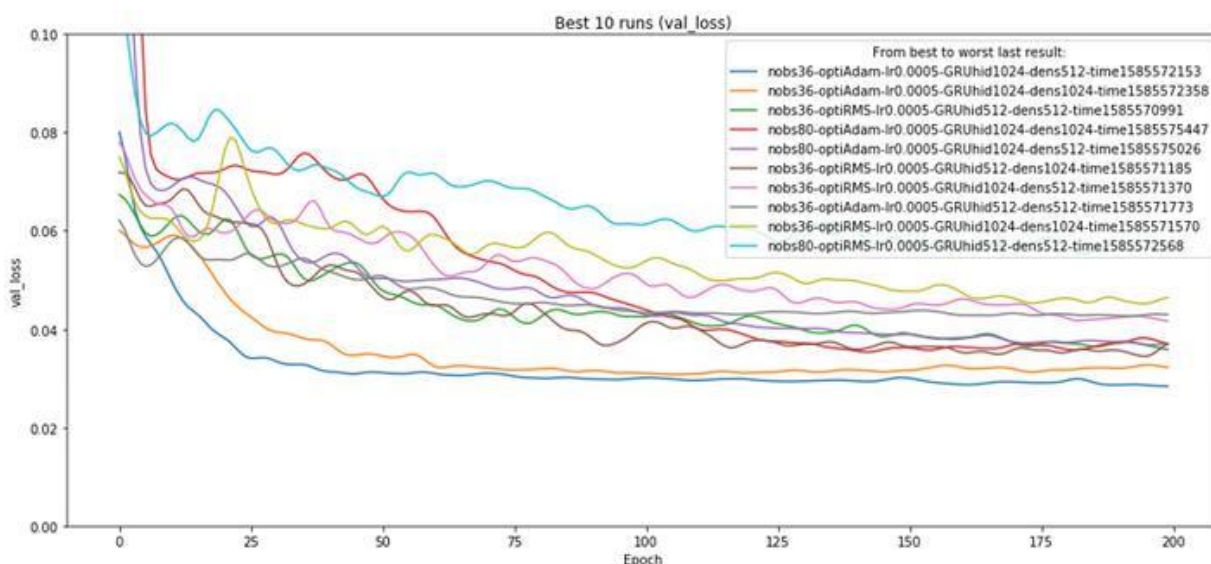


Figure : Visualisation sur Jupyter, lorsque nous avons fait varier différents paramètres tels que l'optimizer ou l'architecture (présenté à titre illustratif)

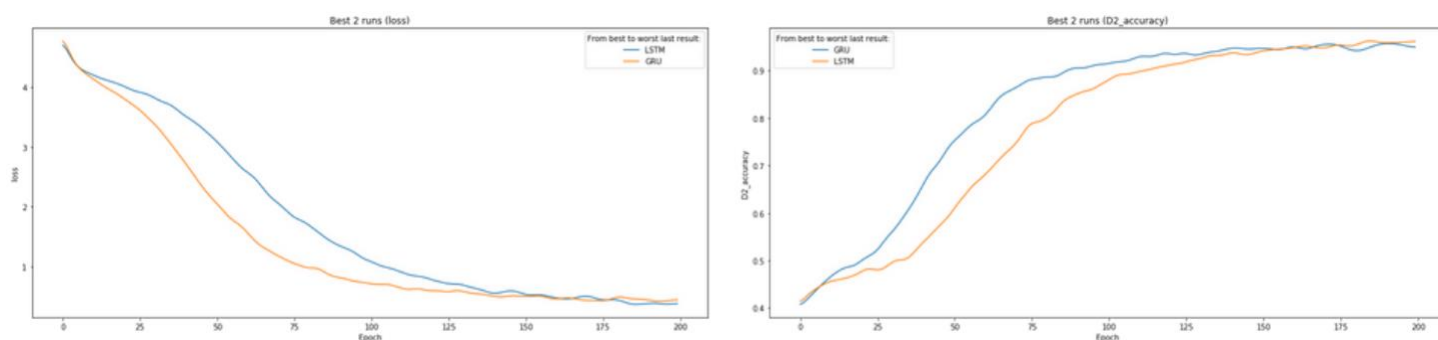


Figure : comparaison LSTM/GRU avec la perte et la précision à J+2

C. Présentation des résultats

● Structure

Le cheminement qui nous a conduit à ce modèle a déjà été en partie présenté dans ce dossier, mais nous allons à présent nous concentrer sur la dernière version utilisée dans notre script de prédiction.

Voici un schéma global de l'architecture finale de notre réseau, avec dans chaque case bleue une couche de neurones différente :

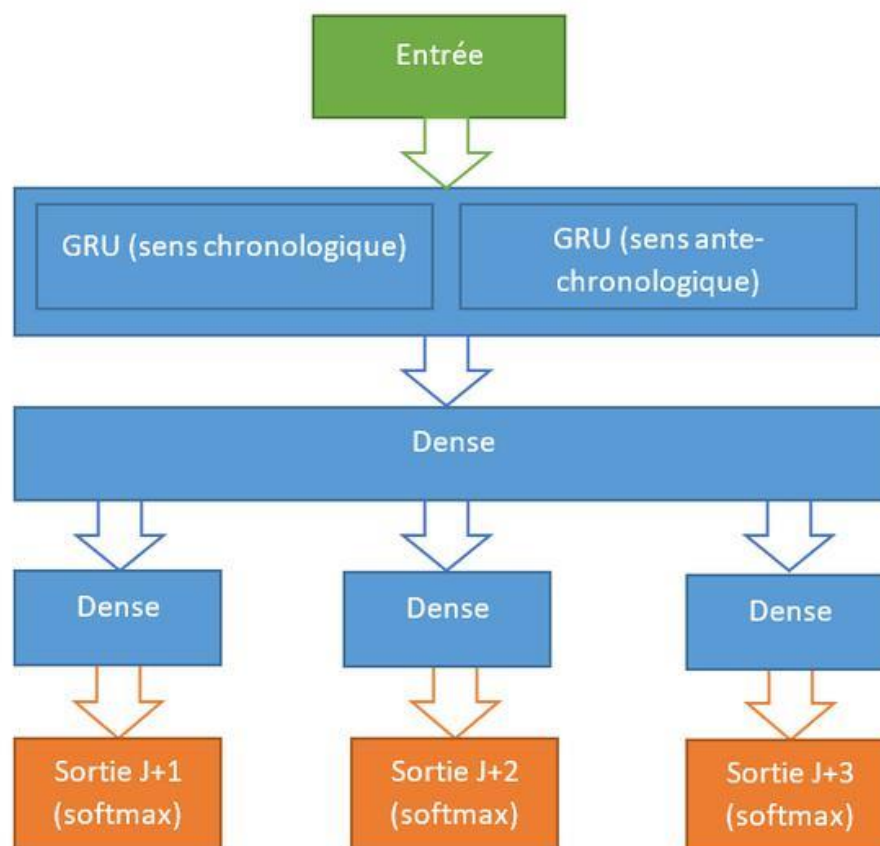


Figure : Structure de notre script de prédiction

Voici la structure exacte créée dans tensorflow pour plus de détails :

Model: "model_3"

Layer (type)	Output Shape	Param #	Connected to
inputLayer (InputLayer)	(None, 32, 54)	0	
GRUchronon (GRU)	(None, 1024)	3314688	inputLayer[0][0]
GRUantechronon (GRU)	(None, 1024)	3314688	inputLayer[0][0]
Concat (Concatenate)	(None, 2048)	0	GRUchronon[0][0] GRUantechronon[0][0]
DensecommonLayer1 (Dense)	(None, 128)	262272	Concat[0][0]
Drop1 (Dropout)	(None, 128)	0	DensecommonLayer1[0][0]
D1 (Dense)	(None, 10)	1290	Drop1[0][0]
D2 (Dense)	(None, 10)	1290	Drop1[0][0]
D3 (Dense)	(None, 10)	1290	Drop1[0][0]

Total params: 6,895,518
 Trainable params: 6,895,518
 Non-trainable params: 0

Figure : Structure détaillée de notre script de prédiction

L'entrée

Nous avons donc en entrée 32 vecteurs correspondants chacun à un intervalle de 3h pour les mesures, avec l'indice de qualité de l'air mesuré, la force du vent, sa direction, la température, l'humidité, un indice des principaux polluants mesurés (O3, NO2, PM10). Ces 32 mesures correspondent à 4 jours, à raison de 8 mesures par jour (durée qui nous a donné les meilleurs résultats expérimentaux).

Avant d'être injectées dans notre modèle, ces données sont toutes normalisées et l'IQ et la direction du vent sont transformés en vecteurs binaires, plus adéquats pour la classification. D'où la forme (32,54) pour notre matrice d'entrée.

Exemple de dataset d'entrée avant normalisation :

	IQ	date	pressure	wind_direction	wind_force	humidity	temperature	NO2	O3	PM10
0	5.0	2015-01-01 00:00:00+00:00	103030	170	2.4	100	272.85	1.0	3.0	1.0
1	5.0	2015-01-01 03:00:00+00:00	102920	160	3.5	95	272.75	1.0	3.0	1.0
2	5.0	2015-01-01 06:00:00+00:00	102990	170	3.4	94	272.65	1.0	3.0	1.0
3	5.0	2015-01-01 09:00:00+00:00	103010	190	4.7	85	274.35	1.0	3.0	1.0
4	5.0	2015-01-01 12:00:00+00:00	102820	190	6.4	72	277.25	1.0	3.0	1.0

Exemple de transformation en vecteur binaire :

Indice de qualité de l'air : 7 -> [0,0,0,0,0,0,1,0,0,0]

La sortie

En sortie de notre script, nous avons 3 vecteurs de taille 10 nous indiquant l'indice prédit pour J+1, J+2 et J+3. Ces 3 vecteurs sont retransformés en entiers compris entre 1 et 10 afin d'obtenir nos résultats finaux.

Chiffres

Nous évaluons nos résultats à l'aide d'un dataset de test. En effet, nous avons environ 3 ans de données complètes, que nous avons découpées en 80% pour l'entraînement et 20% pour le test. Lors de l'entraînement, nous utilisons la méthode du K-fold (avec k=10) et de la cross validation (notions déjà évoquées dans le chapitre précédent). Tensorflow nous permet de mesurer directement les différentes métriques qui nous intéressent ici, à savoir l'accuracy à J+1, J+2 et J+3.

Ainsi, en moyenne, notre modèle obtient les scores suivants en tests (avec des données non utilisées pour l'entraînement) :

	J+1	J+2	J+3
Accuracy moyenne en test :	60%	68%	57%

Et avec notre modèle final nous obtenons les résultats suivants :



Exemple de résultats expérimentaux :

```
--== Training ==--
loss 0.5294313916813629
D1_loss 0.15828224
D2_loss 0.17813154
D3_loss 0.21113715
D1_accuracy 0.9429207
D1_mse 0.008343865
D2_accuracy 0.9377316
D2_mse 0.008867103
D3_accuracy 0.9303188
D3_mse 0.010199547

--== Test ==--
test_loss 3.9238792991638185
test_D1_loss 1.4203745126724243
test_D2_loss 0.9737179279327393
test_D3_loss 1.5470918416976929
test_D1_accuracy 0.6506666541099548
test_D1_mse 0.05516081303358078
test_D2_accuracy 0.7440000176429749
test_D2_mse 0.03899996355175972
test_D3_accuracy 0.6053333282470703
test_D3_mse 0.059596866369247437
```

L'accuracy est donc très élevée en training, mais sans engendrer un overfitting* trop important puisqu'en moyenne les valeurs en tests sont très correctes et proches de nos objectifs initiaux. Par ailleurs notre modèle semble donner une bonne tendance de la qualité de l'air sur nos prédictions en temps réel.

L'objectif principal de notre modèle est de pouvoir prévenir les prochains pics de pollutions. Ainsi avec le modèle que nous avons sélectionné pour la production, nous arrivons à prédire un pic de pollution (c'est-à-dire un indice de qualité de l'air supérieur à 6) à J+1 dans 60%, en J+2 dans 45,5% et J+3 dans 42% des cas (tests effectués sur des données qui n'ont pas été utilisées pour l'entraînement). On peut alors penser à un résultat médiocre mais pour ces mêmes données l'écart moyen est de 1,15 à J+1, 1,79 à J+2 et 2,06 à J+3, (avec des variances inférieures respectives de 0,79, 0,99 et 1,08) ce qui nous permet tout de même d'avoir une tendance plutôt correcte de la qualité de l'air lors de pics de pollution.

***Overfitting:** (ou surapprentissage) *Analyse statistique qui correspond trop étroitement ou exactement à un ensemble particulier de données. Ainsi, cette analyse peut ne pas correspondre à des données supplémentaires ou ne pas prévoir de manière fiable les observations futures.*

II - RÉALISATION TECHNIQUE

A. Architecture

Après avoir travaillé sur le modèle qui est le cœur de notre projet, nous avons également entrepris de réaliser un produit afin de présenter nos résultats de manière automatisée et professionnelle. Ainsi nous avons conçu toute une architecture pour notre projet technique que l'on peut voir ci-dessous.

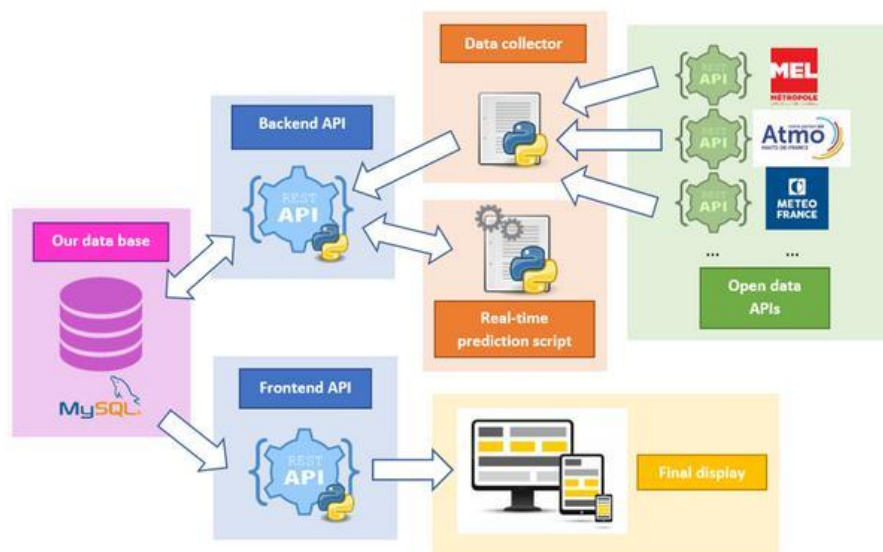


Figure : Architecture de notre projet

● La base de données

Pièce centrale de notre projet, la base de données nous permet de stocker toutes les informations nécessaires pour réaliser nos prédictions, ainsi que pour stocker ces dernières. Nous utilisons une base de données MySQL que nous avons basée sur Google Cloud Platform (GCP). Cela nous a donc permis d'apprendre à utiliser cette plateforme pour l'occasion mais aussi de mieux maîtriser la mise en place d'une base de données sur un serveur distant.



● API frontend et backend

Afin de nous permettre un accès simplifié, rapide et modulaire, nous avons réalisé notre propre API sous python avec le module Flask. Cette API nous permet en backend de remplir la base de données et d'accéder aux données nécessaires pour réaliser la prédiction, et en Frontend d'afficher nos résultats de manière simple.



A l'origine scindée en 2 parties distinctes, nous avons plus tard rassemblé le frontend et le backend pour simplifier la mise en ligne et ne pas perdre trop de temps, l'essentiel du projet étant d'avoir une prédiction fiable. Tout ceci est déployé sur GCP également.

Cette partie du projet représente un investissement conséquent en temps et en ressource car nous n'avions jamais réalisé d'API auparavant. Pour consulter la documentation, elle se trouve un peu plus loin en annexe. A l'intérieur, nous avons réussi à mettre en place différentes requêtes complexes.



● Le data collector

C'est un script python qui est chargé de récupérer toutes les données dont nous avons besoin de manière automatique. Il se connecte aux différentes API pour collecter les données, les nettoie, les met en forme et les envoie à notre API pour être stockées dans la base de données. Les données sont mises à jour une fois par jour.

C'est la partie de la réalisation la plus chronophage, en effet il faut non seulement contacter des API qui ont des fonctionnements différents, mais aussi nous assurer que toutes les données sont bien présentes, avant de les nettoyer, les classer et les agréger et tout cela de manière automatisée avec une gestion d'erreur.

● Le script de prédiction

Cette routine python se lance tous les jours vers 12h afin de réaliser la prédiction de la qualité de l'air à J+1, J+2 et J+3. Pour cela, le script se connecte à notre API afin de demander les données nécessaires et envoyer le résultat à la base de données.



● L'affichage final

L'affichage final sera présenté plus en détails dans la partie sur la présentation des résultats, mais le but de cet élément est d'afficher proprement nos résultats de prédictions afin d'être accessibles à tous. Nous avons fait le choix de réaliser cette page web à l'aide du langage R et du module Shiny qui permet la mise en ligne rapide de datavisualisation.



Ce site nous a permis de réviser nos cours de R et de data visualisation réalisés au cours de l'année, en plus de leur trouver un aspect pratique.

● Temps consacré et intérêt

Au global, cette réalisation technique représente environ 45% du volume horaire du projet, car il était important pour nous d'apprendre à maîtriser de nouvelles technologies comme celles mentionnées tout au long de ce rapport.

La modularité de notre solution nous a également permis de gérer très rapidement des changements effectués sur l'API de la MEL la veille de la présentation de notre projet à ces derniers, sans quoi notre présentation eu été bien moins convaincante. Cet événement nous a conforté dans l'idée qu'il était plus intéressant de prendre du temps lors de la conception pour pouvoir gérer des changements extérieurs sans trop de difficulté.

B. Produit fini

● L'affichage de nos prédictions

Afin de pouvoir être utilisé par le plus grand nombre facilement, notre projet devait avoir un outil de visualisation rapide et simple à utiliser. Nous avons donc réalisé une page web qui permet d'afficher nos prédictions sur les 7 derniers jours (que nous vous invitons à visiter pour plus de détail) :

<https://alexandre-verept.shinyapps.io/App-prediction/>

Page principale :



Résultats récents :

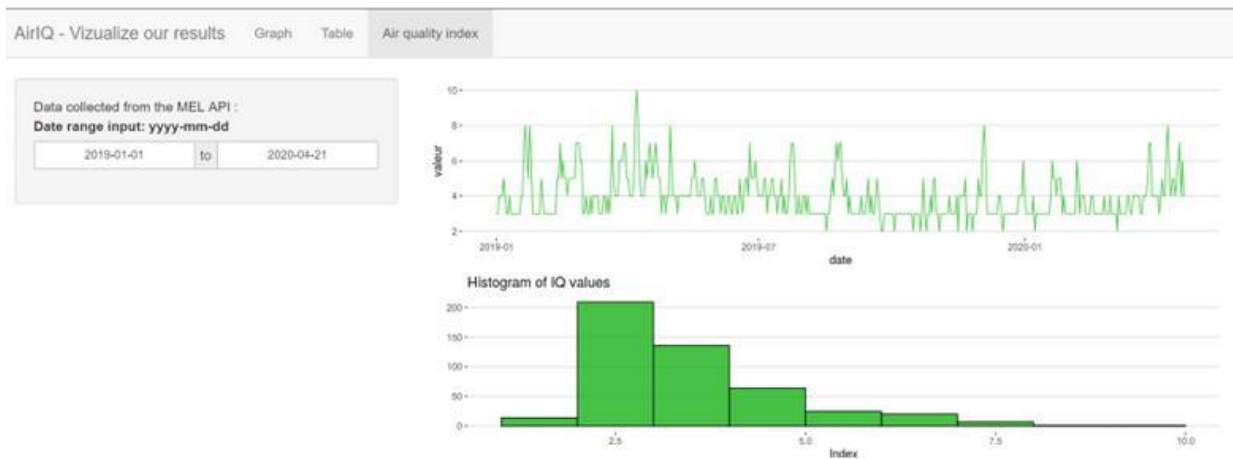
The screenshot shows the 'Table' tab of the 'AirIQ - Visualize our results' application. The table displays the 'date' and 'predicted_value' for the most recent three entries. The 'Show' dropdown is set to '10' and the 'Search' field is empty.

	date	predicted_value
1	2020-04-20	5
2	2020-04-21	6
3	2020-04-22	6

Showing 1 to 3 of 3 entries

Previous 1 Next

Historique de la qualité de l'air ATMO :



● Notre API

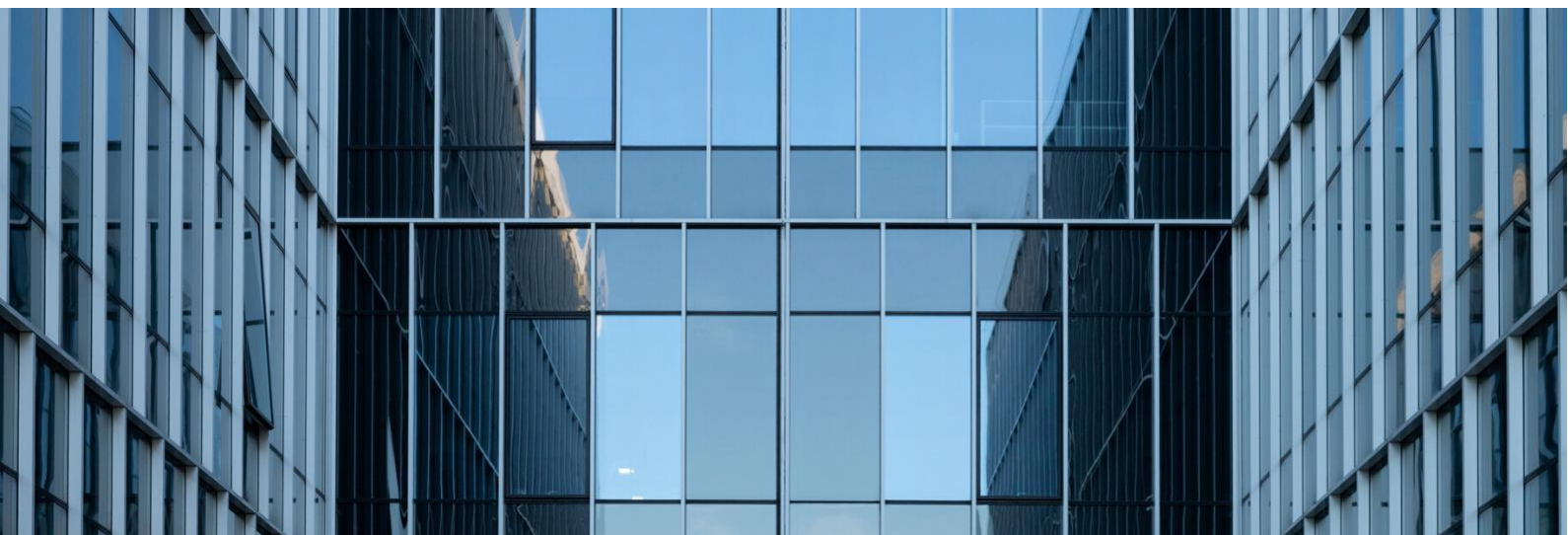
Nous avons également réalisé une API afin d'accéder à nos résultats de n'importe où et de manière automatique.

Le lien vers l'API :

<https://airiq-271312.appspot.com/>

La documentation complète est disponible en annexe et ici :

https://github.com/AlexandreVerept/AirIQ2/blob/master/7_GoogleAPI/README.md



C. Améliorations

Nous avons pensé à plusieurs idées qui pourraient mener à une amélioration de notre modèle. Le temps nous a manqué pour tester correctement toutes ces hypothèses, mais peuvent être envisagées pour une poursuite du projet :

- Une piste évidente est de rajouter encore **plus de données** prises en compte pour une prédiction. Par exemple, nous pourrions inclure une notion de trafic routier, de vacances scolaires, jours fériés... tout ce qui pourrait s'avérer pertinent pour une prédiction de l'indice.
- L'un des axes d'amélioration principale concerne ce qu'on appelle la "**Data augmentation**" : cette méthode consiste à augmenter la taille de notre dataset en créant des exemples manuellement, qui sont proches de ceux déjà existant. Pour prendre une analogie simple, si nous voulions créer un réseau capable de reconnaître des images de chats, nous pourrions augmenter la taille du dataset en changeant légèrement les images, en les inversant, les décalant un peu etc. Cette méthode aurait pour mérite d'encore augmenter la taille du dataset, mais surtout de corriger le nombre de données plus faibles représentant des pics de pollution.
- Il serait également bénéfique d'avoir une **période encore plus longue de données d'entraînement**, en croisant avec d'anciens datasets pour compléter les archives, ou encore de ré-entraîner le modèle régulièrement pour y inclure les dernières données.
- Nous pourrions réaliser un **modèle de prédiction différent pour chaque jour** (J+1, J+2, J+3) au lieu d'un modèle avec un tronc commun et des sorties différentes.
- Comme évoqué précédemment, nous pourrions utiliser la méthode de « **fine tuning** » (qui signifie peaufinage en français) afin de ne mettre à jour les poids que dans certaines parties du réseau.
- Si nous avons accès à un autre réseau de prédiction de qualité de l'air que le nôtre, nous pourrions utiliser une méthode nommée "**Apprentissage par transfert**" (ou transfer learning) qui consiste à inclure une partie d'un réseau déjà entraîné dans un autre modèle.
- Enfin nous pourrions continuer à tester de nombreuses **combinaisons de paramètres** tels que le type, le nombre et la disposition des couches de neurones.

III - GESTION DE PROJET

A. Phase de démarrage

Identification des besoins

A travers ce projet, nous pouvons répondre à différents besoins qui sont les suivants :

- Améliorer notre santé. Lorsque l'on détermine un indice de qualité de l'air, nous tenons la population informée de la pollution qui nous entoure. Et en effet, une étude de l'organisation mondiale de la santé (OMS) dénombre 2 millions de décès prématurés par an à cause de maladies dues à la pollution atmosphérique. Selon le niveau de concentrations des différents polluants, ces composés peuvent être toxiques, irritants pour les yeux et les voies respiratoires et même causer des maladies cardiovasculaires ou des cancers.
- Prévenir la population qu'il est préférable par exemple d'utiliser les transports en commun, le vélo, etc... Savoir prédire à J+2 permet d'anticiper les pics de pollution.
- Appliquer une surveillance continue en réalisant des analyses et des mesures de l'air pour déterminer un indice fiable.
- Compléter les données météorologiques.
- Comprendre les variations de l'atmosphère.

Identification des acteurs

A présent, nous allons lister l'ensemble des acteurs qui ont contribué à ce projet :

- La Métropole Européenne de Lille (MEL) avec qui nous avons eu un rendez-vous à la fin du premier semestre et 2 semaines avant le début du projet du deuxième semestre. Nous avons donc eu affaire à :
 - M. Van Oost, chargé de mission open data
 - M. Martin, chef de projet service public métropolitain de la donnée
 - Mme. Billion-Prunier, chargé de mission qualité de l'air et adaptation au changement climatique
 - M. Ait Sidi Hammou, chargé de mission intelligence artificielle
- ATMO Hauts de France avec qui nous avons pu échanger par mail et avec qui nous aurions dû avoir un rendez-vous le vendredi avant le début du projet mais pour des raisons sanitaires, nous n'avons pas pu les rencontrer. Les personnes contactées sont les suivantes :
 - M. Lecarpentier, responsable d'AIREKA, le laboratoire d'innovations pour la qualité de l'air
 - M. Bourel, responsable société à l'institut catholique de Lille
 - Mme. Pujol Sohne, responsable du pôle modélisation
- L'ISEN :
 - M. Hérissé, doctorant
 - M. Verept, étudiant en Big Data en M1
 - M. Boulanouar, étudiant en Cybersécurité en M1
 - M. Thoor, étudiant en Big Data en M1

Enfin, les personnes impactées par cette prédiction de la qualité de l'air sont les habitants de Lille.



Phase d'analyse du projet

Est-ce rentable ?

N'étant pas l'objectif du projet, notre projet n'est ni rentable ni en déficit.

N'existe-t-il pas d'autres alternatives plus crédibles ?

Il existe d'autres manières de déterminer l'indice de qualité de l'air mais le déterminer grâce à l'intelligence artificielle, nous pensons qu'il n'y a pas d'autres alternatives plus crédibles. C'est même pour cela que nous avons eu un intérêt de la part de la ville et d'ATMO.

Est-il techniquement réalisable ? Accessible ?

Oui, il est techniquement réalisable et accessible puisque nous l'avons déjà fait lors du premier semestre ; il s'agit maintenant de l'améliorer.

Basé sur des hypothèses réalistes ?

Ces hypothèses sont réalistes puisque ce type de modèle a déjà été réalisé dans des villes tels que New York. Nous savons donc, grâce à nos recherches scientifiques, qu'il est basé sur des bases solides.

Avons-nous les moyens ? Les ressources ?

Oui, tout est à notre disposition pour mener ce projet à son terme grâce à tous les acteurs de ce projet.

B. Planification



Découpage du projet

Pour rappel, notre principal et unique produit livrable correspond à un affichage des valeurs de prédiction de notre algorithme.

Dans le but de réaliser ce projet, nous avons choisi un Organigramme Technique Projet qui traite des différents composants d'un produit : le **PBS** (Product Breakdown Structure). En effet, l'affichage de nos données sera un élément composé de différentes parties. Pour notre cas, nous devons par exemple nous occuper d'assembler tous les constitutifs de notre projet afin de pouvoir atteindre notre but. S'il manque un composant comme par exemple la frontend API (voir à la figure 3 située deux pages au-dessus), la base de données ne pourra pas communiquer avec le final display et donc notre affichage sera impossible. Chaque élément est donc vital.

Pour faire part de l'avancement du projet, nous avons choisi de nous fixer des objectifs semaine après semaine afin de ne pas perdre la motivation et d'avoir toujours une base de repères pour nous permettre d'avancer sur le projet.

Voici le planning que nous avons réalisé au début du projet :

Lundi 16	Mardi 17	<u>Mercredi 18</u> ● Finir les recherches scientifiques	Jeudi 19	<u>Vendredi 20</u> ● Base de données avec l'état de l'art rempli automatiquement tous les jours
<u>Lundi 23</u> ● Voir quel autre réseau de neurones il est possible d'utiliser	Mardi 24	Mercredi 25	Jeudi 26	<u>Vendredi 27</u> ● Etablir un modèle de machine Learning
Lundi 30	Mardi 31	Mercredi 1	Jeudi 2	<u>Vendredi 3</u> ● Modèle fonctionnel mais non optimal
<u>Lundi 6</u> ● Commencer la partie frontend ● Améliorer la précision	Mardi 7	Mercredi 8	Jeudi 9	Vendredi 10
Lundi 13	Mardi 14	<u>Mercredi 15</u> ● Commencer la partie documentation	Jeudi 16	<u>Vendredi 17</u> ● Finir la partie basse du schéma ci-dessus (frontend)
<u>Lundi 20</u> ● Préparation de la réunion avec la MEL	Mardi 21	<u>Mercredi 22</u> ● Finir définitivement le modèle prédictif	<u>Jeudi 23</u> ● Réunion avec la MEL	<u>Vendredi 24</u> ● Commencer le rapport
Lundi 27	Mardi 28	<u>Mercredi 29</u> ● Finir le rapport	Jeudi 30	Vendredi 31

Dans notre façon de fonctionner, nous cherchons à mettre l'accent sur le cœur de notre projet tout au long des 7 semaines, c'est-à-dire le script de prédiction de l'indice de qualité de l'air.

Cependant, nous avons travaillé en parallèle sur une partie du schéma de la figure 8 chaque semaine. Ainsi, chacun n'a pas la responsabilité entière d'un composant de notre projet mais au contraire, a pu amener ses compétences techniques au sein du travail réalisé. On peut donc parler de responsabilité technique et non de responsabilité par phase.

Pour l'organisation, afin que nous ne soyons pas tous sur la même tâche, nous avons mis en place un document où l'on inscrit toutes nos avancées dans la journée afin de savoir ce sur quoi tout le monde travail. De plus, avant de démarrer d'un coup les travaux sur un composant, nous faisons la phase de démarrage ensemble en commençant par une discussion pour savoir où nous voulons aller et comment le réaliser. Cela permet à chacun d'être en accord avec ce qui sera réalisé. Cette façon de raisonner nous a donc amené vers une approche **Bottom Up**. Le groupe fait le tour de toutes les tâches imaginables sans se soucier véritablement de l'organisation puisque c'est en faisant le brainstorming que cela se précise.

● Score Card

Pour représenter notre diagramme de Gantt, nous avons choisi d'établir une Score Card car elle nous permet à la fois d'avoir un schéma fonctionnel illustrant l'ordre dans lequel les activités sont réalisées mais aussi parce qu'elle définit l'ensemble des tâches faites par chacun des membres du groupe.

Nom de tâche	Nom de la personne	Date début	Date fin prévue	Date de fin effective	Avancement	Notes
Recherches initiales	Maxime, Mohamed	16-mars	17-mars	17-mars	100	Recherches scientifiques
Mettre en ligne la BDD	Alexandre	16-mars	17-mars	18-mars	100	-Comprendre GCP -Mettre la BDD en ligne -Accéder à la BDD depuis l'extérieur Problème : héberger une API sur un serveur google et accéder à la base de données sur ce même serveur
Se renseigner sur les modèles mathématiques existants	Maxime, Mohamed	17-mars	18-mars	18-mars	100	Assimiler la notion de Time Series Forecasting et test compilations avec notre dataset
Recherche de Dataset intéressante pour le modèle	Mohamed	17-mars	17-mars	17-mars	100	Cf readme GitHub
Créer le dataset collector complet	Alexandre, Mohamed	19-mars	20-mars	20-mars	100	Agrège tous les datasets

Figure : Extrait de la Score Card

C. Exécution

● Contrôle du projet

Dans l'objectif d'avoir un contrôle constant du projet, nous avons mis en place différentes méthodes de suivi, notamment la Score Card ou l'avancée quotidienne. Cette façon de faire qui est similaire à un diagramme de Gantt nous a permis de superviser et d'évaluer l'avancée de chacun dans les tâches réalisées. Enfin, les réunions sont prévues en début de semaine afin d'établir un bilan de la semaine précédente.

● Suivi des dépenses

Nous avons fait un suivi régulier des dépenses effectuées dans Google Cloud Platform. Pour accéder aux facturations et pour voir leur évolution, elles ont été mises en annexe.

● Communication

En plus de notre tableau de bord avec la ScoreCard, nous avons réalisé un document pour renseigner l'avancée quotidienne du travail, un peu comme un journal de bord. Chaque membre peut décrire plus précisément ce qu'il a fait durant sa journée.

Avancée quotidienne

lundi 16 mars 2020 18:36

Lundi 16 mars 2020 :

- Réunion de 10h15 à 11h15 afin de définir le planning avec les deadlines et les objectifs à atteindre. Démarrage du projet. Compte-rendu de la réunion.
- Envoi du mail à la MEL pour avoir accès à de nouveaux jeux de données pour travailler.
- Création du fichier Excel Organisation pour connaître l'avancement de chacun dans ses tâches.
- Création d'un dossier avec toutes les recherches scientifiques effectués ce jour (Mohamed et Maxime).
- Création d'un dossier pour effectuer tous les tests nécessaires. Ici, il a été créé pour voir si chacun avait accès à la base de données créée sur GCP (Alexandre).

Mardi 17 mars 2020 :

- Synthèse des recherches scientifiques effectuées la veille pour la documentation.
- Envoi d'un mail à ATMO pour essayer d'avoir un jeu de données plus conséquent sur l'indice de qualité de l'air.
- Recherche de jeux de données complémentaires pour notre modèle prédictif
- Recherche sur les modèles mathématiques envisageables pour les comparer à notre modèle prédictif (Time Series Forecasting)
- Compilation du programme sur Jupyter Notebook de la Vector AutoRegression.

Mercredi 18 mars 2020 :

- Mise en ligne de l'API frontend
- Création du dataset collector complet
- Réalisation de notebook avec des prévisions basées sur un modèle mathématique
- Nouvelle recherche sur différents modèles mathématiques en adéquations avec notre dataset

Figure : Extrait de l'avancée quotidienne

● Leadership

Durant toutes les phases du projet, les choix et décisions sont partagés entre les différents membres du groupe. Chacun a son mot à dire sur les solutions à apporter. Cependant Alexandre a le rôle de surveiller l'avancement et la bonne réalisation de chaque étape.

Ce projet est la suite logique du projet du premier semestre car nous avons décidé de retravailler ensemble pour aller plus loin dans notre démarche. Par conséquent tous les membres du groupe ont des opinions très proches concernant la stratégie et la vision.

La récompense de ce projet est l'acquisition de nouvelles connaissances qui nous seront bénéfiques à l'avenir pour de futurs stages ou emplois. En outre, obtenir une excellente note finale est évidemment une grosse motivation.

Au cours du projet et durant chaque étape, chacun des membres avait une mission précise qui pouvait être réalisée seul ou à deux. Afin d'optimiser l'avancée, nous nous sommes arrangés pour travailler en parallèle et d'être le moins souvent possible à trois sur une même mission. Le défi et l'envie de se dépasser afin de rendre un travail d'une qualité qui nous convienne sont ce qui nous fait avancer au quotidien.

De plus, à la fin de notre projet nous avons une réunion programmée avec la MEL et ATMO afin de présenter nos résultats, ce qui nous apporte une certaine satisfaction personnelle et peut nous aider à créer un début de réseau professionnel. Pour finir, c'est une fierté de voir son projet abouti et de devoir le présenter à différentes organisations dans la perspective de futures collaborations.



CONCLUSION

L'objectif initial de 70% de précision à J+2 est proche, même si cela a entraîné une légère perte de précision en J+1 par rapport au premier semestre. Nos tests et méthodes d'évaluations sont devenues plus rigoureuses, grâce à l'utilisation de validation et de test, en plus d'avoir un dataset presque 3 fois plus important. L'overfitting a été grandement diminué. Les résultats sont satisfaisants et nous sommes heureux d'avoir pu créer un modèle à la hauteur de nos objectifs initiaux.

Il existe également quelques pistes d'améliorations que nous n'avons pas pu tester faute de temps en fin de projet. Par exemple, nous pouvons jouer sur les mises à jour des différents poids lors de l'entraînement ("fine tuning"), réaliser 3 modèles complètement différents pour chaque jour (J+1, J+2 et J+3) ou encore ajouter de nouveaux paramètres au dataset. Il y a donc encore quelques idées à explorer pour améliorer le modèle.

De plus nous avons su tirer profit de notre travail en gagnant de l'expérience sur les différents modèles et méthodes d'évaluation des résultats en machine Learning.

Concernant notre solution Cloud, nous sommes heureux du résultat accompli et de ce que nous avons appris à réaliser avec notre base de données, notre API et toutes nos routines. Ceci nous a permis de créer un résultat plus professionnel qu'au premier semestre. En effet il était important pour nous de travailler l'intégration d'un modèle de machine Learning et de superviser un pipeline de données. C'est pourquoi nous avons décidé de prendre du temps pour apprendre ou améliorer nos compétences sur toutes ces technologies qui pourraient nous servir plus tard. Certes le résultat du modèle était important, mais la priorité pour nous était d'en tirer un bénéfice à plus long terme.

Avec le contexte du Covid 19 qu'a traversé ce projet, notre groupe s'est débrouillé pour travailler en autonomie et à distance, tout en essayant de garder le contact avec la MEL et ATMO qui nous ont aidés à obtenir les datasets.

Ainsi, les résultats encourageants de notre projet peuvent servir de base d'amélioration pour quiconque souhaite prédire l'indice de qualité de l'air dans l'optique de prévenir avec un temps d'avance les futurs épisodes de pollution.

REMERCIEMENTS

En premier lieu, nous voudrions remercier Kévin Hérissé, notre tuteur de projet pour son temps et son implication au cours du projet. En effet, suite aux bons résultats du premier semestre, il nous a fait à nouveau confiance pour retravailler sur le sujet. Il nous a donné l'envie de nous investir dans notre projet et a aussi amené à travers sa bonne humeur une bonne ambiance de travail durant les réunions.

Nous souhaiterions également remercier la MEL avec qui nous avons pu échanger régulièrement par mail ou directement lors d'une réunion. Ils nous ont suivi dans notre projet et nous ont permis d'avoir des datasets plus anciens afin d'avoir un jeu de données plus important pour faire tourner notre modèle prédictif. Enfin, leur intérêt pour ce projet s'est notamment manifesté par la rencontre du jeudi 30 avril. Nous avons eu en effet la chance de présenter notre projet devant eux par visio-conférence et celle-ci a été un réel succès puisque la MEL souhaite aujourd'hui pérenniser ce travail.


BIBLIOGRAPHIE

<https://keras.io/optimizers/>
<https://keras.io/metrics/>
<https://keras.io/losses/>
<https://keras.io/models/model/>
https://keras.io/examples/imdb_bidirectional_lstm/
https://keras.io/examples/lstm_stateful/
https://www.tensorflow.org/api_docs/python/tf/keras/layers/GRU
https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Precision
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/BinaryCrossentropy
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/BinaryAccuracy
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/CategoricalCrossentropy
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/MeanAbsoluteError
https://www.tensorflow.org/api_docs/python/tf/concat
<https://datascience.stackexchange.com/questions/14581/when-to-use-gru-over-lstm>
<http://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema/>
<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
<https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/>
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
<https://www.lokad.com/fr/pr%C3%A9vision-de-s%C3%A9ries-chronologiques>
<https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322>
<https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/>
<https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/>

ANNEXE

15/04/2020

Mon compte...cturation – Transactions – Facturation – AirIQ – Google Cloud Platform

 <p>ID du profil de paiement 7643-8000-7441</p> <p>Nom Alexandre Verept</p> <p>Pseudonyme du compte de paiement Google Cloud 01B4DC-0FB8BF-ACA773</p> <p>Date de création du récapitulatif : 15 avr. 2020</p>		
1-31 mars 2020		
		Solde de clôture : 0,00 €
Date	Description	Montant (EUR)
1-31 mars 2020	Credit FreeTrial-Credit-01B4DC-0FB8BF-ACA773	-14,37 €
1-31 mars 2020	Compute Engine Static Ip Charge: 192.341 heures [Conversion de devises : USD en EUR avec le taux 0.921]	1,76 €
1-31 mars 2020	Compute Engine N1 Predefined Instance Ram running in EMEA: 3.557 gibioctets/heure [Conversion de devises : USD en EUR avec le taux 0.921]	0,02 €
1-31 mars 2020	Compute Engine N1 Predefined Instance Core running in EMEA: 0.948 heures [Conversion de devises : USD en EUR avec le taux 0.921]	0,03 €
1-31 mars 2020	Cloud Storage Standard Storage Europe Multi-region: 1.06 gibioctets/mois [Conversion de devises : USD en EUR avec le taux 0.921]	0,03 €
1-31 mars 2020	Cloud SQL Storage PD SSD for DB in Americas: 4.522 gibioctets/mois [Conversion de devises : USD en EUR avec le taux 0.921]	0,71 €
1-31 mars 2020	Cloud SQL IP address idling in seconds for DB in Americas: 285.528 heures [Conversion de devises : USD en EUR avec le taux 0.921]	2,63 €
1-31 mars 2020	Cloud SQL DB standard Intel N1 1 VCPU running in Americas (with 30% promotional discount): 50.472 heures [Conversion de devises : USD en EUR avec le taux 0.921]	3,14 €
1-31 mars 2020	App Engine Flex Instance RAM EMEA: 99.894 gibioctets/heure [Conversion de devises : USD en EUR avec le taux 0.921]	0,72 €
1-31 mars 2020	App Engine Flex Instance Core Hours EMEA: 99.894 heures [Conversion de devises : USD en EUR avec le taux 0.921]	5,33 €
		Solde initial : 0,00 €


<https://console.cloud.google.com/billing/01B4DC-0FB8BF-ACA773/history?project=airiq-271312>

1/2

Position de compte google cloud en mars

27/04/2020

Mon compte...cturation – Transactions – Facturation – AirIQ – Google Cloud Platform



ID du profil de paiement
7643-8000-7441

Nom
Alexandre Verept

Pseudonyme du compte de paiement
Google Cloud 01B4DC-0FB8BF-ACA773

Date de création du récapitulatif :
27 avr. 2020

1–27 avr. 2020

Solde de clôture : -0,07 €

Date	Description	Montant (EUR)
1–30 avr. 2020	Credit FreeTrial:Credit-01B4DC-0FB8BF-ACA773	-95,64 €
1–30 avr. 2020	Compute Engine Storage PD Capacity in London: 1.281 gibioctets/mois [Conversion de devises : USD en EUR avec le taux 0.909]	0,06 €
1–30 avr. 2020	Compute Engine Static Ip Charge: 623.504 heures [Conversion de devises : USD en EUR avec le taux 0.909]	5,66 €
1–30 avr. 2020	Compute Engine N1 Predefined Instance Ram running in London: 333.837 gibioctets/heure [Conversion de devises : USD en EUR avec le taux 0.909]	1,66 €
1–30 avr. 2020	Compute Engine N1 Predefined Instance Ram running in EMEA: 339.837 gibioctets/heure [Conversion de devises : USD en EUR avec le taux 0.909]	1,44 €
1–30 avr. 2020	Compute Engine N1 Predefined Instance Core running in London: 89.023 heures [Conversion de devises : USD en EUR avec le taux 0.909]	3,30 €
1–30 avr. 2020	Compute Engine N1 Predefined Instance Core running in EMEA: 90.623 heures [Conversion de devises : USD en EUR avec le taux 0.909]	2,86 €
1–30 avr. 2020	Cloud Storage Standard Storage Europe Multi-region: 2.405 gibioctets/mois [Conversion de devises : USD en EUR avec le taux 0.909]	0,06 €
1–30 avr. 2020	Cloud SQL Storage PD SSD for DB in Americas: 8.667 gibioctets/mois [Conversion de devises : USD en EUR avec le taux 0.909]	1,34 €
1–30 avr. 2020	Cloud SQL IP address idling in seconds for DB in Americas: 218.526 heures [Conversion de devises : USD en EUR avec le taux 0.909]	1,99 €
1–30 avr. 2020	Cloud SQL DB standard Intel N1 1 VCPU running in Americas (with 30% promotional discount): 405.474 heures [Conversion de devises : USD en EUR avec le taux 0.909]	24,91 €
1–30 avr. 2020	App Engine Flex Instance RAM London: 7.105 gibioctets/heure [Conversion de devises : USD en EUR avec le taux 0.909]	0,06 €

<https://console.cloud.google.com/billing/01B4DC-0FB8BF-ACA773/history?project=airiq-271312>

1/2

Position de compte google cloud en avril partie1

27/04/2020

Mon compte...cturation – Transactions – Facturation – AirIQ – Google Cloud Platform

1-30 avr. 2020	App Engine Flex Instance RAM EMEA: 867.878 gbiocets/heure [Conversion de devises : USD en EUR avec le taux 0.909]	6,15 €
1-30 avr. 2020	App Engine Flex Instance Core Hours London: 7.105 heures [Conversion de devises : USD en EUR avec le taux 0.909]	0,41 €
1-30 avr. 2020	App Engine Flex Instance Core Hours EMEA: 868.871 heures [Conversion de devises : USD en EUR avec le taux 0.909]	45,73 €
1 avr. 2020	TVA	-0,06 €
		Solde initial : 0,00 €

Nos recherches sur le Time Series Forecasting

La prévision de séries chronologiques est un domaine important de l'apprentissage automatique. Des prévisions sont faites pour de nouvelles données lorsque le résultat réel peut ne pas être connu avant une date ultérieure. La prévision consiste à prendre des modèles mathématiques adaptés aux données historiques dont on dispose et à les utiliser pour prédire les observations futures.

L'objectif principal de l'analyse des séries chronologiques est de développer des modèles mathématiques qui fournissent des descriptions plausibles à partir d'échantillons de données. Il existe plusieurs modèles mathématiques, à partir de notre jeu de données, nous avons utilisé le VAR (Vector Autoregression), ARIMA (Autoregressive Integrated Moving Average), VARMA (Vector Autoregression Moving-Average), et HWES (Holt Winter's Exponential Smoothing). Nous allons détailler chacune des méthodes utilisées ci-dessous.

VAR

La méthode d'auto-régression vectorielle (VAR) modélise la prochaine étape de chaque série chronologique à l'aide d'un modèle d'auto-régression (la méthode d'auto-régression modélise la prochaine étape de la séquence en fonction linéaire des observations aux pas de temps précédents). Il s'agit de la généralisation de l'auto-régression à plusieurs séries chronologiques parallèles, son utilisation peut être étendue par exemple aux séries temporelles multivariées. La méthode convient aux séries chronologiques multivariées sans composantes de tendance et saisonnières.

ARIMA

La méthode ARIMA modélise la prochaine étape de la séquence en fonction linéaire des observations différenciées et des erreurs résiduelles aux pas de temps précédents. Il combine à la fois les modèles d'auto-régression (AR) et de moyenne mobile (La méthode de la moyenne mobile modélise la prochaine étape de la séquence en fonction linéaire des erreurs résiduelles d'un processus moyen à des pas de temps antérieurs) ainsi qu'une étape de prétraitement de différenciation de la séquence pour rendre la séquence stationnaire, appelée intégration. La méthode convient aux séries chronologiques univariées avec tendance et sans composantes saisonnières.

VARMA

La méthode de moyenne mobile à auto-régression vectorielle (VARMA) modélise la prochaine étape de chaque série chronologique à l'aide d'un modèle ARMA (La méthode de la moyenne mobile auto-régressive ARMA modélise la prochaine étape de la séquence en fonction linéaire des observations et des erreurs résiduelles aux pas de temps précédents). C'est la généralisation de l'ARMA à plusieurs séries chronologiques parallèles, par exemple les séries temporelles multivariées. La méthode convient aux séries chronologiques multivariées sans composantes de tendance et saisonnières.

HWES

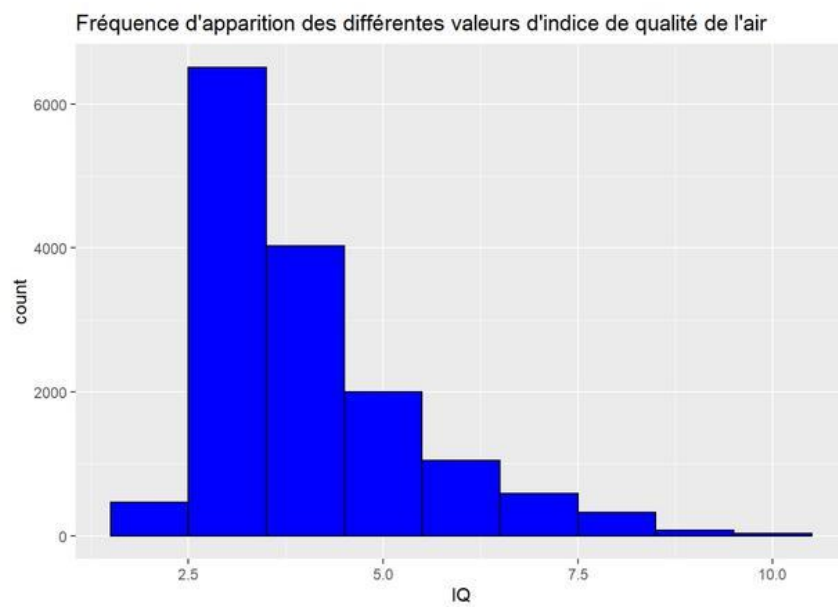
Le lissage exponentiel de Holt Winter (HWES), également appelé méthode de lissage exponentiel triple, modélise le pas de temps suivant en tant que fonction linéaire pondérée exponentiellement des observations des pas de temps précédents, en tenant compte des tendances et de la saisonnalité.

La méthode convient aux séries chronologiques univariées avec des composantes de tendance et/ou saisonnières.

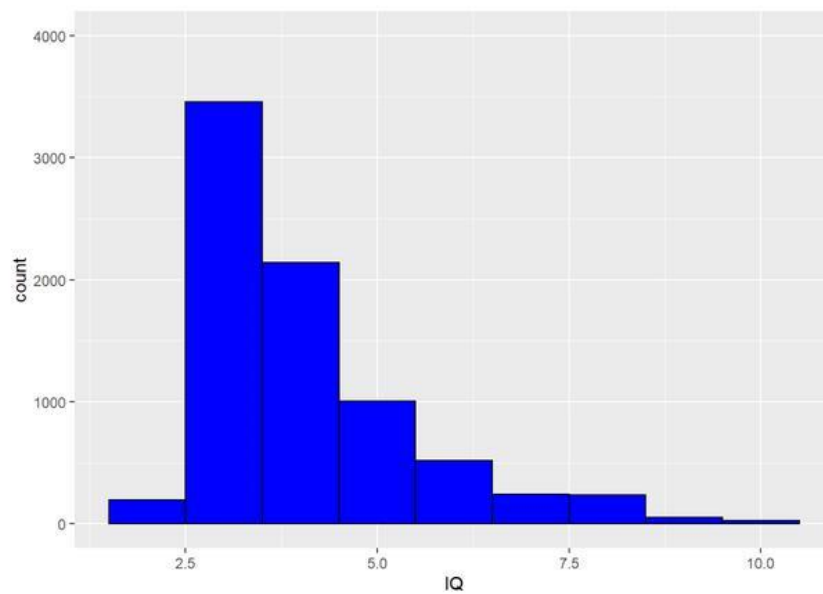
Conclusion

Après avoir testé les différents modèles cités ci-dessus avec notre dataset, nous en avons conclu qu'aucun modèles mathématiques ne correspondaient à notre jeu de données. Puisque pour que le modèle soit le plus optimale possible afin que la prédiction soit acceptable. Il faut que ce modèle prenne en compte un certain type de donnée, il doit convenir aux données de types séries chronologiques variées avec des composantes de tendance et/ou saisonnières tel que l'est notre dataset.

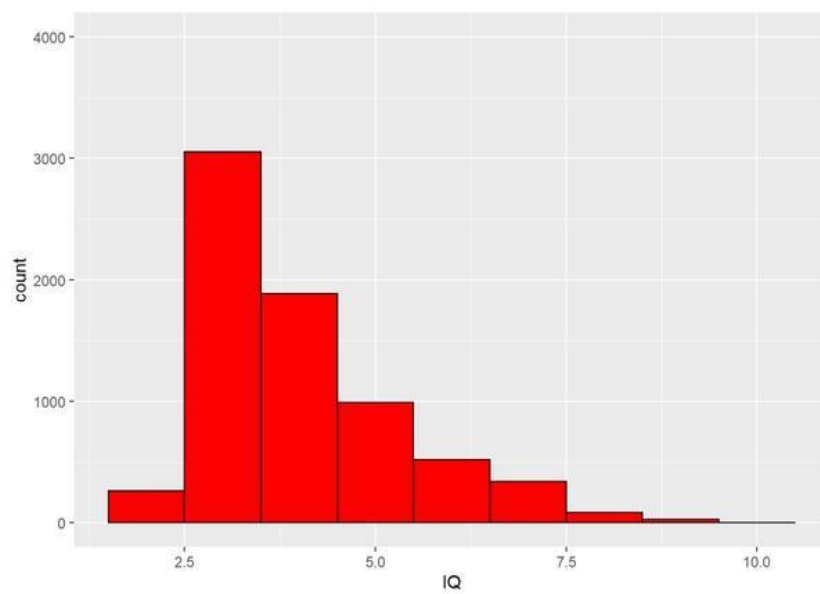
Quelques observations réalisées sur le dataset :



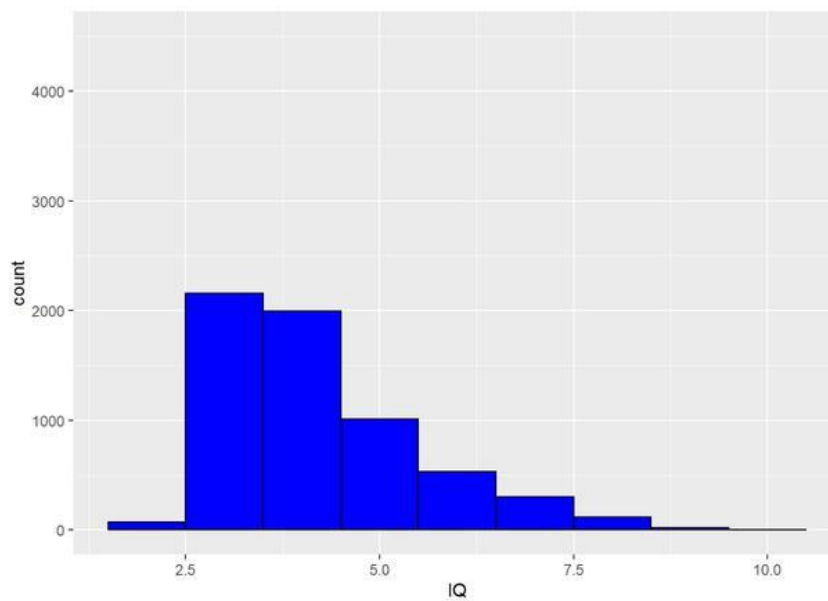
Fréquence d'apparition des différents indices de la qualité de l'air pour une température inférieures à la moyenne



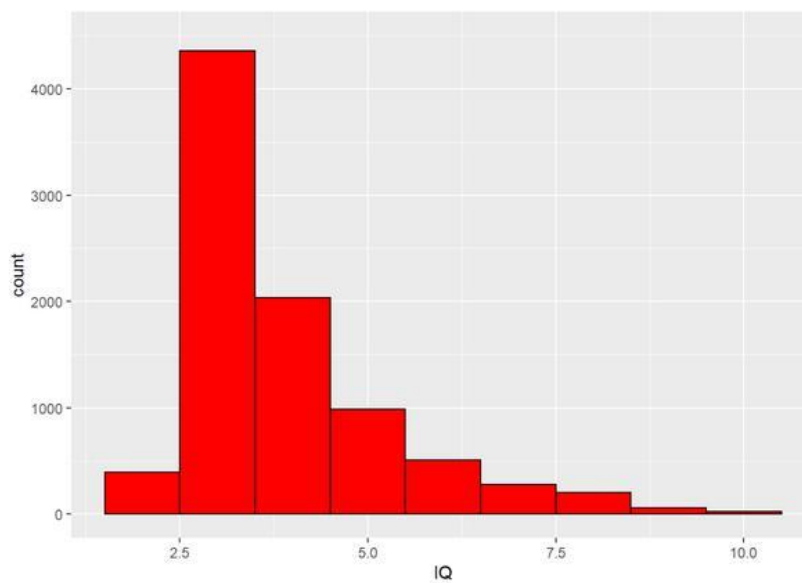
Fréquence d'apparition des différents indices de la qualité de l'air pour les températures supérieures à la moyenne



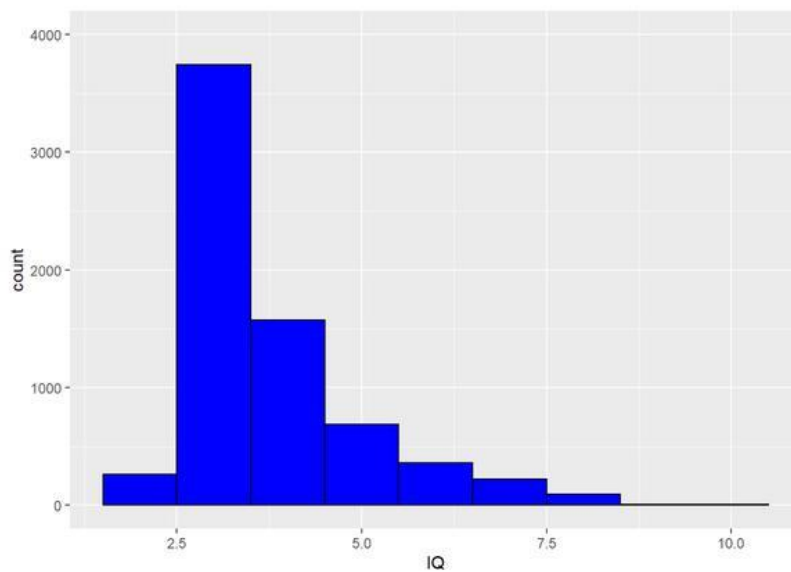
Fréquence d'apparition des différents indices de la qualité de l'air pour une humidité inférieure à la moyenne



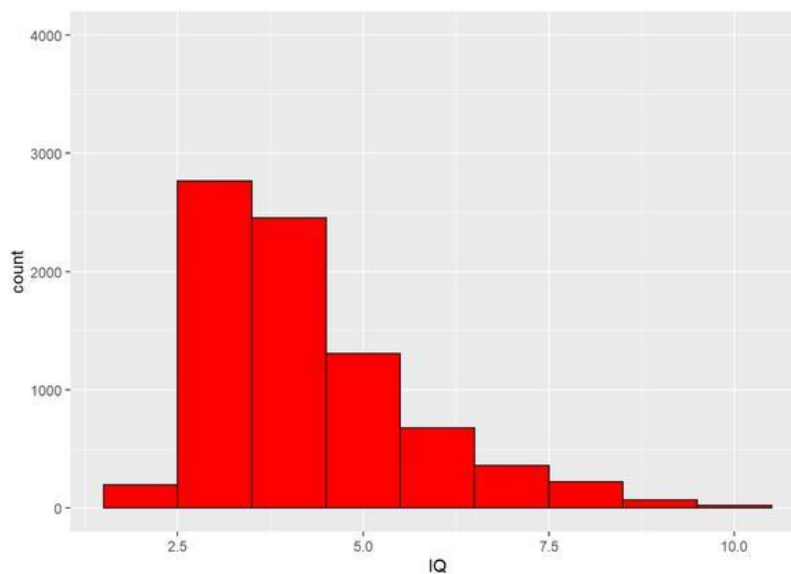
Fréquence d'apparition des différents indices de la qualité de l'air pour une humidité supérieure à la moyenne



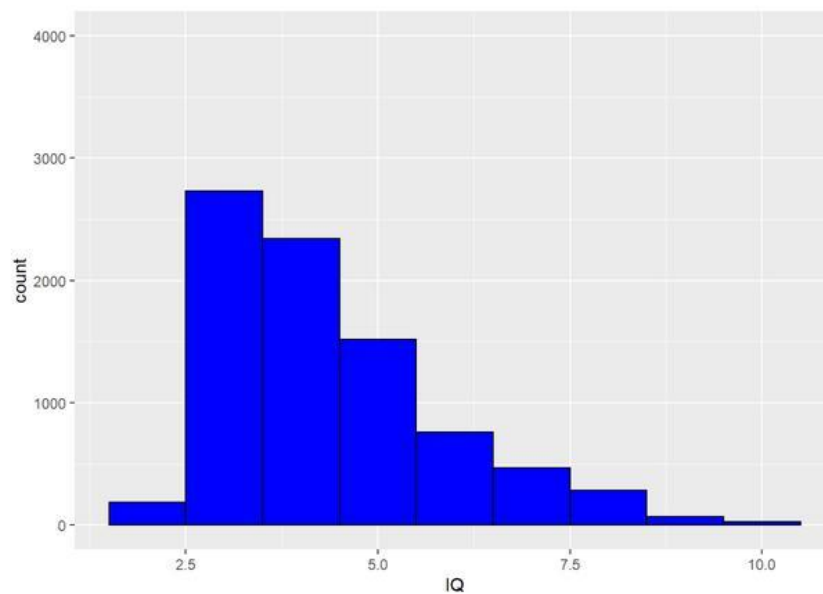
Fréquence d'apparition des différents indices de la qualité de l'air pour une pression inférieure à la moyenne



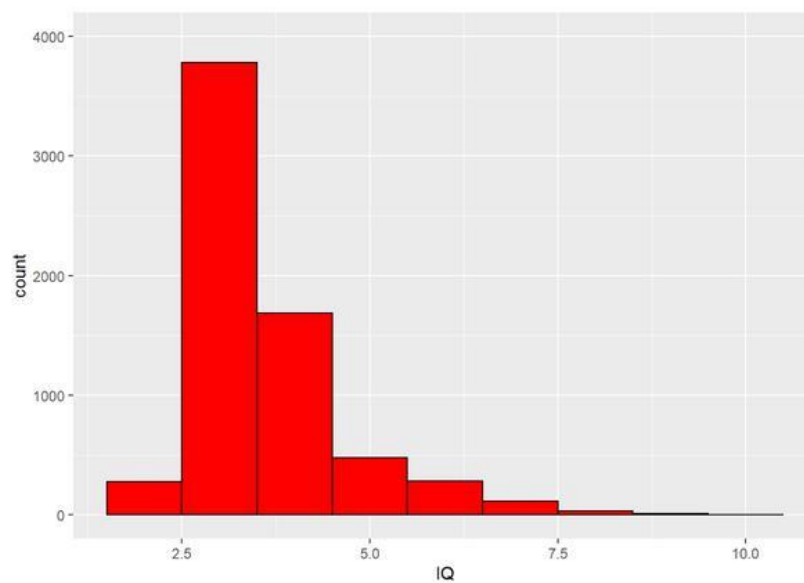
Fréquence d'apparition des différents indices de la qualité de l'air pour une pression supérieure à la moyenne



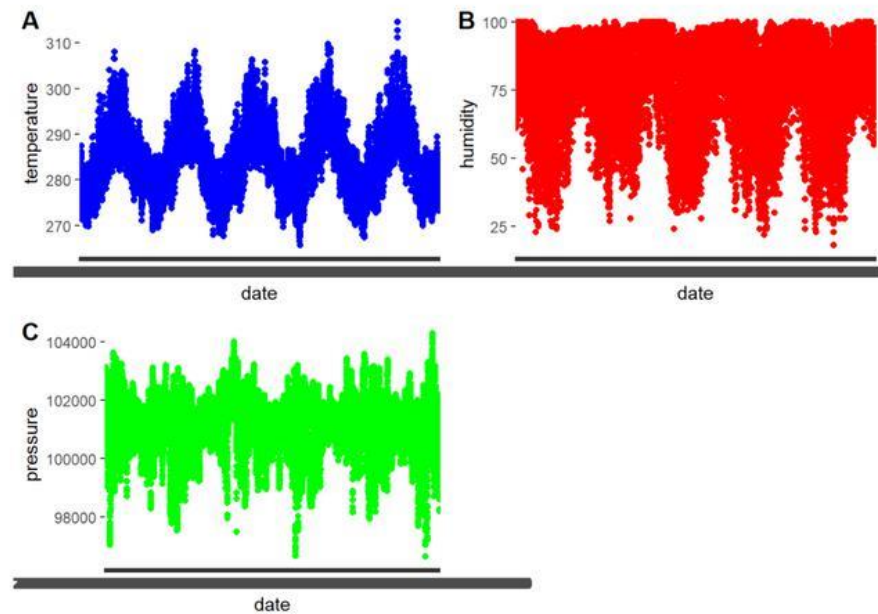
Fréquence d'apparition des différents indices de la qualité de l'air pour une force du vent inférieure à la moyenne



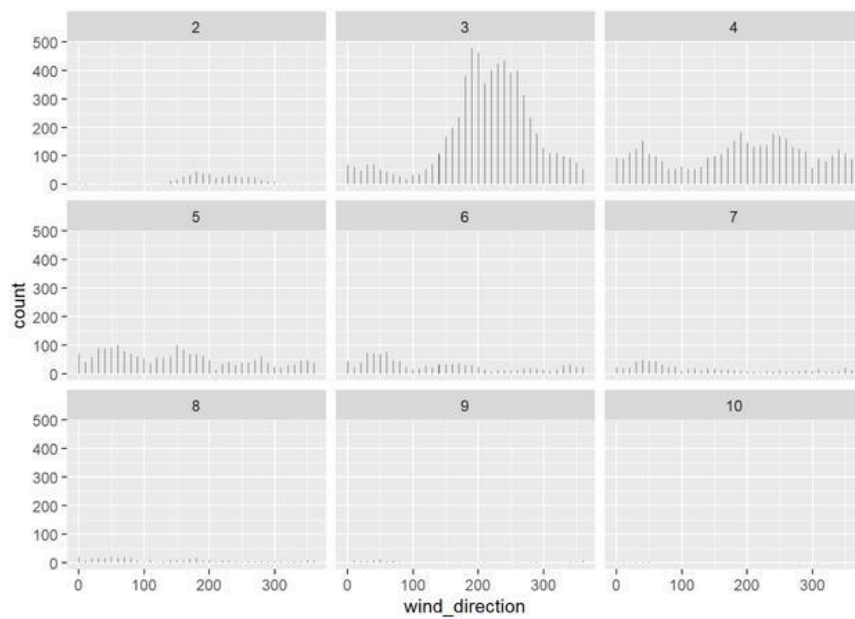
Fréquence d'apparition des différents indices de la qualité de l'air pour une force du vent supérieur à la moyenne



Evolution de différents paramètres en fonction du temps



Afficher un graphe pour chaque IQ en fonction de la direction du vent (entre 0 et 360°)



Documentation de notre API

AIRIQ 2

7_GoogleAPI

Here you can find the API that receive various data from the `DataCollector` or our `PredictionScript` in a way to store them in our `Database`.

Here is the [link to the API](#)

Command	Type of request	Description	Return example
<code>/</code>	GET	Root of the API	<code>"Welcome to our API"</code>
<code>/test</code>	POST	This command is just here to test our POST methods in other scripts. It receive a JSON and send it back.	<code>{"test": "123"}</code>
<code>/prediction</code> or <code>/prediction/<nbOfDays></code>	GET	This method send all the information needed to make one prediction. It send a list of JSON. The optional <code><nbOfDays></code> must be an int (2 by default). Note that this number means the number of days you want, in addition to the today's data.	<code>[{"date": {"0": "Thu, 02 Apr 2020 00:00:00 GMT"}, "value": {"0": 5}}, {"date": {"0": "2020-04-02 06:00:00", "1": "2020-04-02 03:00:00", "2": "2020-04-02 00:00:00"}, "humidity": {"0": 84.0, "1": 85.0, "2": 82.0}, "pressure": {"0": 101110, "1": 101170, "2": 101270}, "temperature": {"0": 276.05, "1": 275.25, "2": 274.55}, "wind_direction": {"0": 220, "1": 160, "2": 290}, "wind_force": {"0": 1.0, "1": 0.5, "2": 1.4}}]</code>
<code>/realtimepredictions</code>	POST	Receive a JSON file with a prediction with: <code>Iq+1</code> , <code>Iq+2</code> , <code>Iq+3</code> , <code>date</code> .	<code>200</code>
<code>/infodatacollector/<type></code>	POST	Receive a JSON file with information coming from the <code>dataCollector</code> . The parameter <code><type></code> must be <code>'iq'</code> , <code>'pollutant'</code> or <code>'synop'</code> .	<code>200</code>
<code>/allpredictions</code>	GET	Get a dictionary with all the predictions made to this day with their <code>dateofprediction</code> , <code>typeofprediction</code> , <code>value</code> .	<code>{ "dateofprediction": {"0": "2020-04-10", "1": "2020-04-09", "2": "2020-04-09", "3": "2020-04-08", "4": "2020-04-08", "5": "2020-04-07", "6": "2020-04-04", "7": "2020-04-03", "8": "2020-04-02"}, "typeofprediction": {"0": "J+3", "1": "J+3", "2": "J+2", "3": "J+2", "4": "J+1", "5": "J+1", "6": "J+3", "7": "J+2", "8": "J+1"}, "value": {"0": 4.9194, "1": 4.59621, "2": 5.85185, "3": 4.46729, "4": 5.47996, "5": 3.7226, "6": 2.70894, "7": 3.14022, "8": 4.22177}} </code>
<code>/getprediction</code> or <code>/getprediction/<nbOfDays></code>	GET	This method send all the predictions for the last X days in a JSON. The optional <code><nbOfDays></code> must be an int (0 by default). Please take note that 0 means that there is only the value of today.	<code>{ "dateofprediction": {"0": "2020-04-10", "1": "2020-04-09", "2": "2020-04-08"}, "insertdate": {"0": "Tue, 07 Apr 2020 00:00:00 GMT", "1": "Tue, 07 Apr 2020 00:00:00 GMT", "2": "Tue, 07 Apr 2020 00:00:00 GMT"}, "typeofprediction": {"0": "J+3", "1": "J+2", "2": "J+1"}, "value": {"0": 4.9194, "1": 5.85185, "2": 5.47996}} </code>