Rapport du projet de programmation Informatique

Par Daphné Below et Patrick Mu

Apprentissage du logiciel Github :

Lors de la première séance, nous ne maîtrisions pas suffisamment le logiciel Github. Nous avons donc partagé le code de notre algorithme par mail sous forme de bloc-notes afin de pouvoir avancer mutuellement. Après un premier entretien avec le professeur et grâce à des tutoriels trouvés sur Internet, nous avons appris à transférer les modifications Python sur le dépôt Github et à importer ces modifications dans notre fichier. Cela nous a permis d'importer les modifications sur le dépôt Github au fil des séances et de les consulter facilement. Nous avons utilisé pour cela le terminal Git Bash en tapant les codes suivants :

Transférer les modifications de github à son pc:

.Si le dossier n'est pas encore créé sur le bureau :

-faire en sorte d'être dans son bureau (avec cd Onedrive, cd Desktop)

-écrire "git clone https://github.com/Patrick-Mu/ProjetEIVP.git"

.Si le dossier est déjà dans le bureau :

-faire en sorte d'être dans le dossier "projetEIVP" (avec cd Onedrive, cd Desktop, cd ProjetEIVP)

-écrire "git pull"

Transférer les modifications du pc vers github :

.faire en sorte d'être dans le dossier "projetEIVP" (avec cd Onedrive, cd Desktop, cd ProjetEIVP)

.faire: "git init" puis "git add ." puis "git commit -m "Modification"" puis "git push"

Difficulté rencontrée sur Github :

Nous avons essayé de modifier l'algorithme simultanément, ce qui a engendré une erreur lorsque nous avons essayé d'importer les modifications sur le dépôt. En effet, il est impossible de faire un "git push" avant un "git pull" si des modifications ont été faites auparavant.

Travail sur Python : journal de bord

Pour pouvoir travailler sur Python, nous avons commencé par importer les lignes du fichier Excel grâce à nos connaissances de prépa (*f.readline()*).

Lors de la première séance, nous avons créé une fonction permettant d'obtenir la liste des valeurs d'une colonne donnée. En parcourant la colonne *sent_at* du fichier Excel, nous avons remarqué que les dates des relevés étaient toujours comprises entre le 11 et le 25 du mois d'Août 2019. Nous avons alors découvert que 6 expériences avaient été réalisées et que celles-ci étaient répertoriées dans la colonne *id (nombre entier allant de 1 à 6).* Dans toutes les fonctions écrites, nous avons donc rajouté la variable *id* comme argument.

Afin d'interpréter les données de la colonne sent_at, nous avons créé une fonction *sent_at_date* qui importe une liste dans laquelle chaque élément est une liste de 4 valeurs : le jour, l'heure, les minutes et les secondes du relevé. Chaque expérience se fait au sein du même mois. Nous n'avons donc pas relevé la valeur du mois et de l'année.

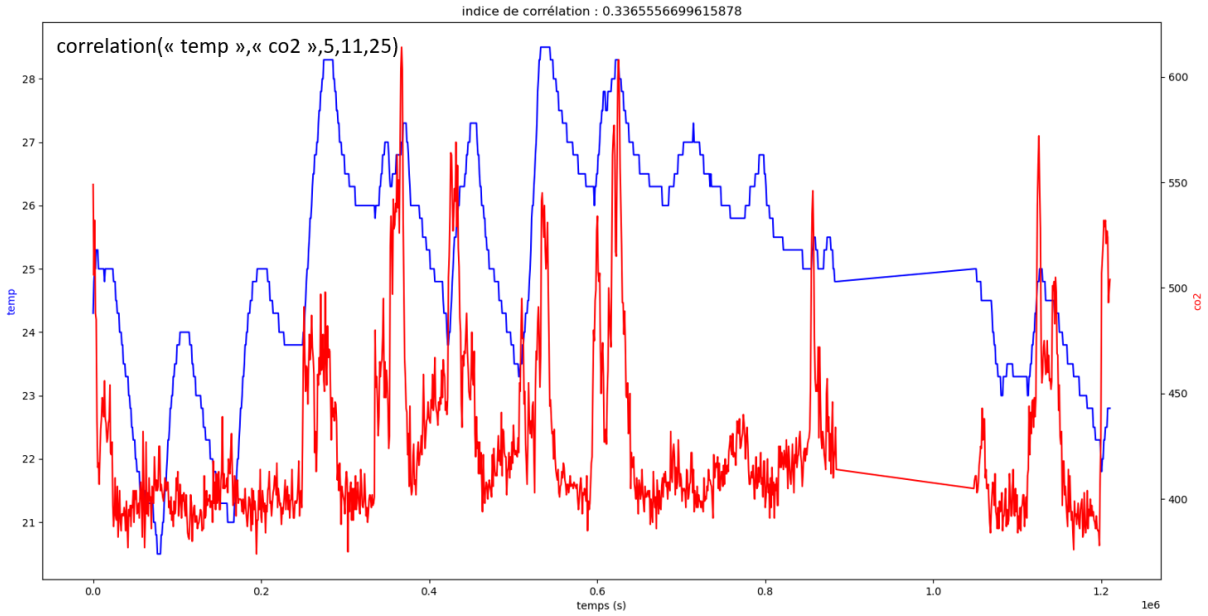
Nous avons ensuite créé la fonction *conv_sec* afin de convertir les éléments de la liste précédente en secondes et de pouvoir tracer les courbes.

Nous avons également décidé de créer deux fonctions distinctes à la place d'une seule fonction. En effet, afin de spécifier un intervalle de temps dans la ligne de commande et donc d'ajouter en argument un jour de début et un jour de fin dans nos fonctions, il est nécessaire de conserver les dates en jour.

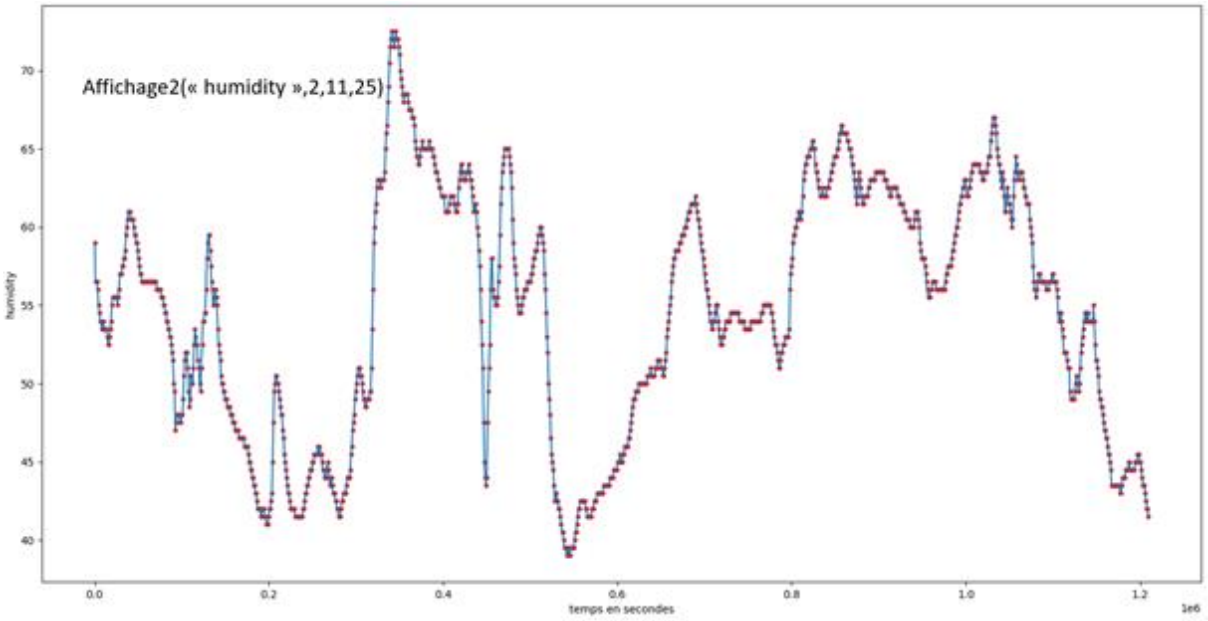
Pour la première fonction *affichage*, la complexité en temps est trop importante (quelques minutes environ). Nous avons en effet construit les listes de valeurs X et Y en parcourant des listes de listes. Afin de résoudre le problème, nous sommes passés par la fonction *courbes*. Cette dernière extrait dans la liste de 4 éléments le jour uniquement. La complexité obtenue est raisonnable car nous ne manipulons plus de listes de listes.

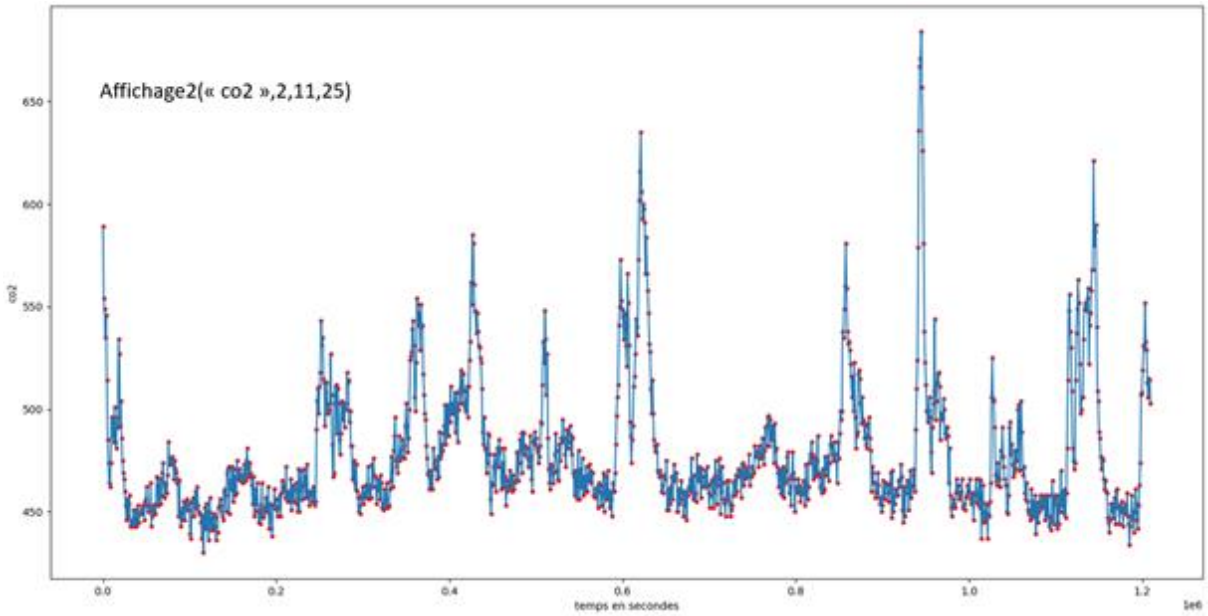
L'apprentissage des différentes fonctionnalités du module *matplotlib* a constitué une partie importante de notre travail. Ce module nous a permis de tracer les différentes courbes demandées et de les afficher.

Nous avons notamment appris à choisir le type de points (nuage de points discrets ou ligne continue), à régler la taille et la couleur des courbes, à ajouter une légende et un titre ou encore à afficher deux courbes simultanément avec des échelles de valeurs différentes en ordonnée. (cf fonction *correlation*).



Exemples de courbes affichées:



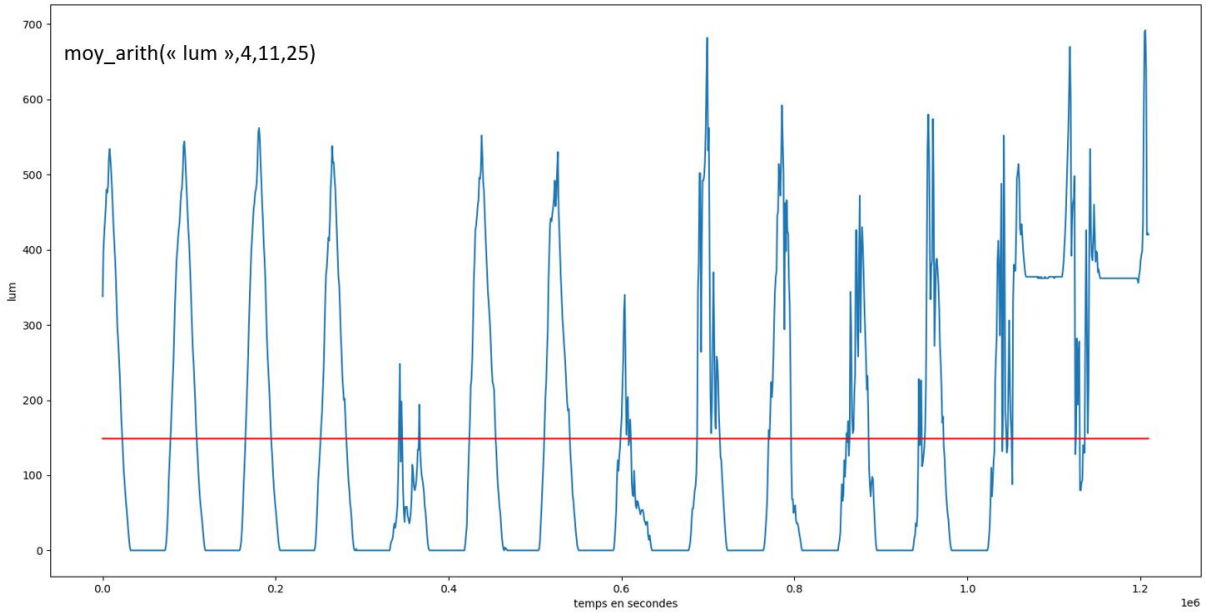


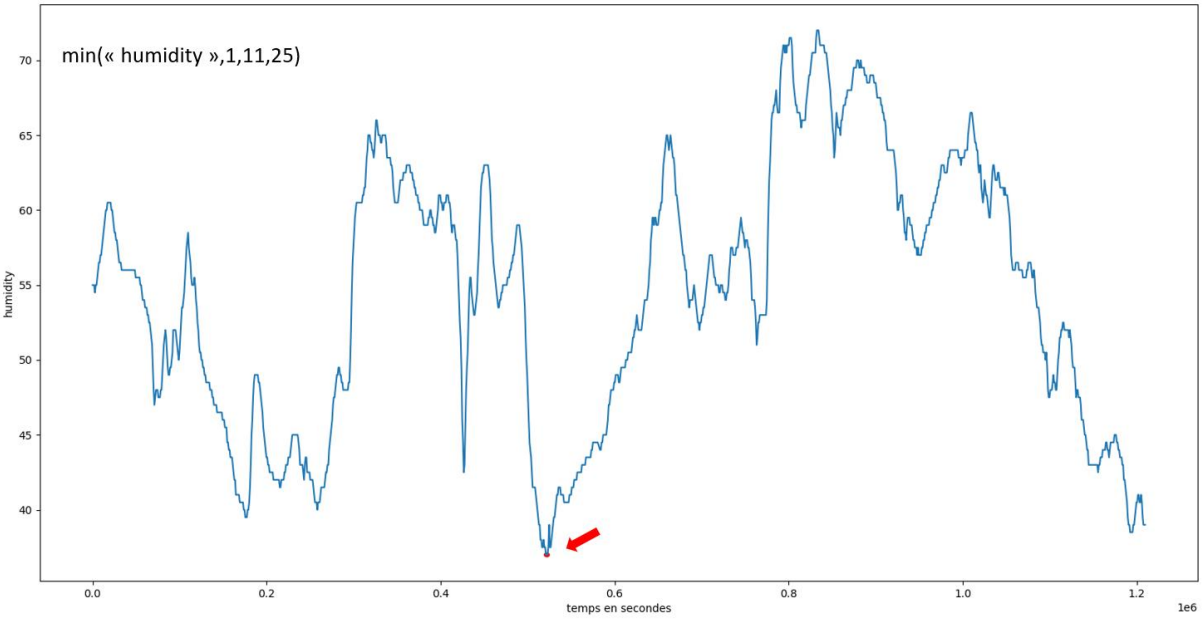
Remarque pour l'expérience numéro 5 :

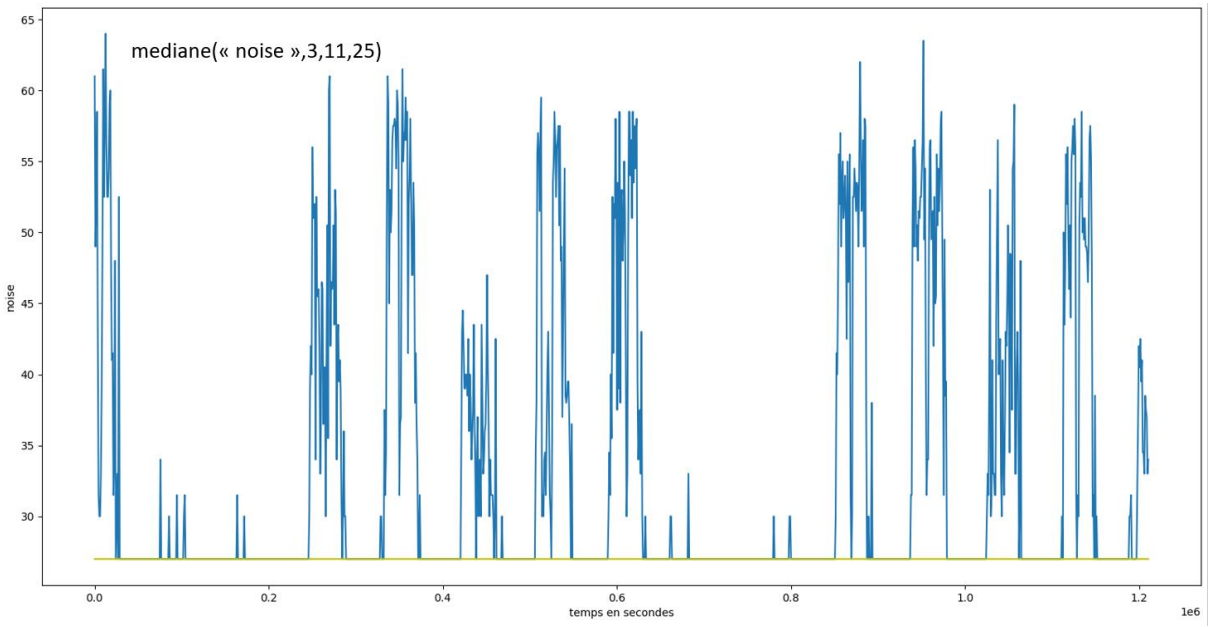
On constate sur les différentes courbes obtenues pour l'expérience 5 une absence de données le 22/08/2019. Le relevé n'a donc visiblement pas eu lieu ce jour ci. Cette absence révèle sans doute un dysfonctionnement ponctuel de l'ensemble des capteurs. C'est pourquoi nous avons modifié la fonction *courbes*, affichant un message d'erreur dans le cas où ce jour serait le jour initial ou final.

L'écriture des fonctions permettant de calculer les valeurs statistiques (min, max, moyenne, écart-type, etc…) n'ont pas causé de difficultés. Il en est de même pour les fonctions liées à l'indice humidex et à l'indice de corrélation. Pour calculer la médiane des valeurs, il a cependant été nécessaire de trier les valeurs de la liste en amont par l'intermédiaire d'un tri à bulles. Ce tri n'est pas le plus efficace en termes de complexité mais il est relativement court et facilement compréhensible. Nous avons alors utilisé les fonctions du module matplotlib pour afficher ces valeurs sur la courbe et pour les mettre en relief.

Affichage des valeurs statistiques sur les courbes :


moy_arith(« lum »,4,11,25)
lum
temps en secondes
1e6


min(« humidity »,1,11,25)
humidity
temps en secondes
1e6

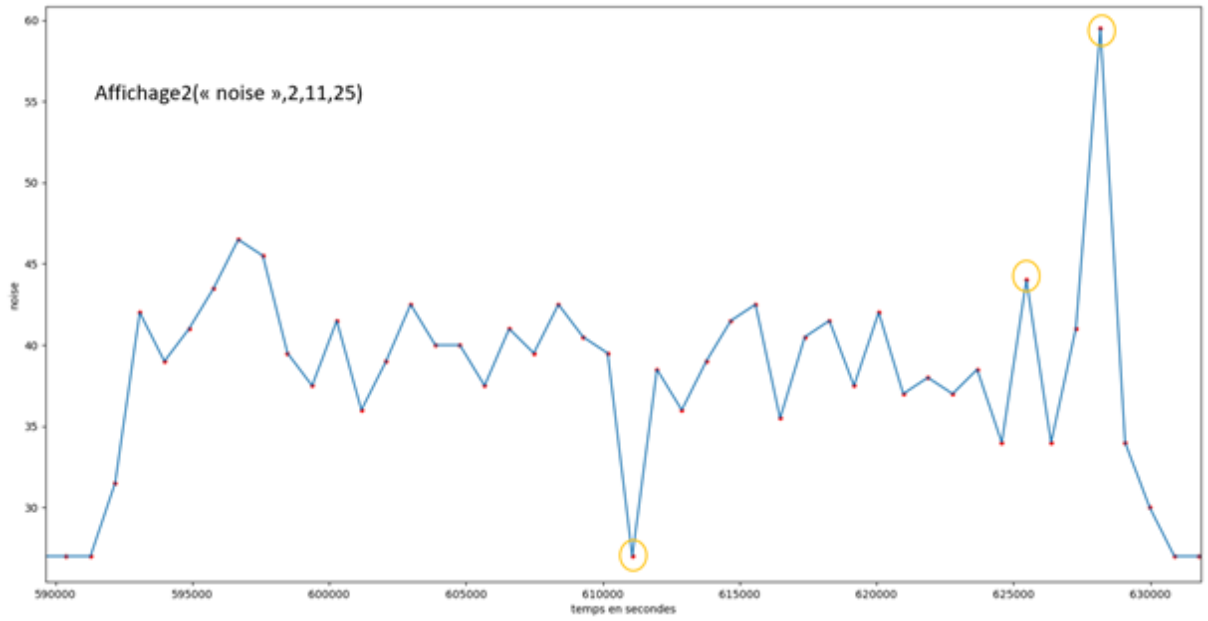


Sur la dernière courbe, la valeur de la médiane correspond à la valeur minimale de la variable noise. Cela montre que plus de la moitié des valeurs de la variable *noise* valent 27,0 dB qui est la valeur minimale de la variable.

Anomalies : Définition / solution proposée pour les détecter automatiquement et les afficher sur la courbe.

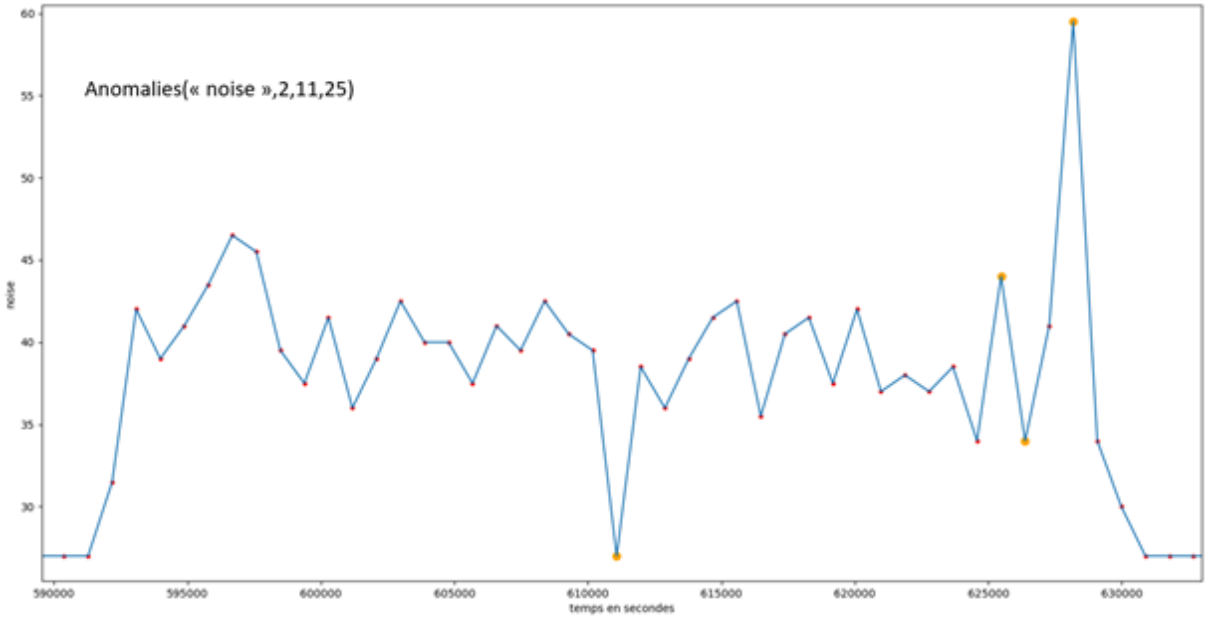
Définition d'une anomalie :

Lors de l'analyse des différentes courbes obtenues pour une expérience donnée, nous avons remarqué la présence de pics anormaux (entourés en orange sur la courbe) qui semblent se dénoter par rapport aux autres valeurs.



Nous appelons donc anomalie tout extremum local pour lequel l'écart avec la valeur précédente ainsi que celui avec la valeur suivante sont supérieurs à une valeur seuil. La valeur seuil choisie correspond à l'écart-type de la courbe, calculé grâce à la fonction définie précédemment.

Nous avons ainsi implémenté un algorithme qui permet de détecter ces pics et de les afficher sur la courbe (en orange) pour une expérience donnée.



On remarque que les anomalies observées sur la courbe d'origine ont bien été prises en compte par la fonction *Anomalies*.

Néanmoins, il y a un quatrième point détecté comme une anomalie par l'algorithme. En effet, lorsqu'un point est une anomalie, il est possible qu'un point frontalier à ce point devienne une anomalie « par erreur » à cause de l'écart entre les deux points. Ce problème était assez récurrent et nous n'avons pas vraiment réussi à trouver de solutions pour le résoudre.

De plus, nous avons remarqué que notre fonction ne détecte aucune anomalie pour les courbes de température, d'humidité et de luminosité. Cela peut s'expliquer par leur faible variation dans le temps.