Professeurs référents : 01/12/20 P-A. MASSE et N. LAGA

**Rapport de projet informatique : Sujet 1**

MARCHAND PEYCLIT Clara

MESSIOUX Adrien

Groupe C, binôme 2



Notre projet, le sujet 1, consistait à analyser des données : luminosité, bruit, et autres données récoltées à l'aide de six capteurs placés dans des bureaux et mises dans un tableur csv. Nous devions récupérer ces données, les mettre sous forme de courbe et repérer des potentielles anomalies dans ces données. Tout d’abord, nous devions utiliser GitHub comme outil de travail collaboratif pour mener à bien ce projet.

I Utilisation de Github :

GitHub est, comme précisé auparavant, un outil de travail collaboratif. Cela permet à chaque collaborateur du projet de travailler sur sa ou ses parties, à n’importe quelle heure, de valider le travail d’un autre collaborateur, de commenter son travail, tout ça en évitant les conflits de code. Il permet aussi de travailler hors connexion sur le projet commun. L’utilisation de Git et Github nous a permis d’avancer rapidement sur notre projet, une fois que nous avons compris à la fois son fonctionnement, et son intérêt. Pour ce faire, nous nous sommes créés un dépôt « global » de nos données, c’est-à-dire de nos lignes de codes et nos données du tableur csv, et nous avions chacun notre dépôt « local » de ces données. Lorsqu’on décide de travailler sur le projet, il fallait utiliser la commande « git pull » qui récupérait les potentielles modifications d’un collaborateur sur le dépôt global, pour les mettre sur son propre dépôt local. On effectue en suite ses propres modifications, on les ajoute à git via la commande « git add » et on les « commit », c’est-à-dire qu’on les dépose sur le dépôt global, pour qu’un autre collaborateur puisse en profiter.

II L’affichage des courbes :

Avant de pouvoir utiliser les données regroupées dans le fichier csv, nous avons dû utiliser le module pandas afin de les extraire. Les données étaient alors sous la forme d’un tableau de format DataFrame. Pour manipuler plus aisément les données, nous avons transformé chaque colonne du tableau en liste avec la commande “L[].tolist()”. On pouvait alors faire les opérations courantes sur les listes.

Après avoir obtenue les listes, nous avons choisi de séparer la liste regroupant les six capteurs en six sous-listes : une pour chaque capteur. Puis dans un second temps de séparer les valeurs de chaque catégorie de données en fonction du capteur qui les avait enregistrés.

Ensuite on a travaillé sur les données relatives au temps, la liste “sent\_at”. Les différentes prises étaient classées de la manière suivante “année-mois-jour heure:minute:seconde”. Le premier problème était lié au fait que python comprenne “-”, séparant l’année du mois par exemple, comme un signe moins, cela entrainait un calcul très long et donc un affichage très lent des courbes. Pour y pallier, nous avons créé une liste T qui prend les valeurs de temps sous la forme “ [instant, jour, mois, année] ”. Puis à partir de la liste T, nous avons créé une t qui renvoie le nombre de secondes qui se sont écoulées depuis le premier instant. Comme toutes les mesures ont été prise en aout 2019, on ne tient compte que du jour et le l’instant.

Une fois que l’on a obtenu tous les instants, on peut tracer chaque catégorie de données en fonction du temps. Pour cela, on a donc tracé chaque sous liste correspondant à un capteur en fonction de la sous liste du temps correspondant au même capteur.

III Les valeurs statistiques :

Nous avions pour travail de calculer les valeurs statistiques suivantes pour chaque courbe : la moyenne, les valeurs minimales et maximales, la variance, l’écart-type, l’indice Humidex et le coefficient de corrélation entre deux variables.

Pour le calcul du minimum, le squelette du code est le suivant : on alloue une place de mémoire, on parcourt la liste de donnée et on compare la première valeur avec la deuxième, la plus petite des deux est stockée, puis on compare la valeur stockée avec la suivante, la plus petite des deux est stockée, jusqu’à la fin de la liste. On procède de même pour le calcul du maximum, en stockant cette fois-ci les plus grandes valeurs.

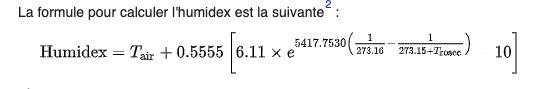
Pour le calcul de la moyenne, on alloue un espace mémoire et on fait la somme de toutes les valeurs présentes dans la liste, puis on divise cette somme par la taille de la liste, c’est-à-dire par le nombre d’éléments qui la composent.

Pour le calcul de la médiane, on fait la division euclidienne du nombre d’éléments de la liste pour connaitre sa parité et on trouve la médiane en fonction.

Pour ce qui concerne la variance, on alloue un espace mémoire et on fait la somme des différences au carré de toutes les valeurs de la liste avec sa moyenne (ce qui est la définition de la variance).

L’écart type est trouvé en faisant la racine carrée de la variance.

Pour calculer l’indice Humidex, nous avons utilisé la formule provenant de Wikipédia. En calculant dans un premier temps le point de rosée puis en l’intégrant à la formule en utilisant comme température : la liste des données de température obtenues grâce aux capteurs. Nous avons tracé cet indice en fonction des données de température relatives aux six capteurs.



Formule pour calculer la température du point de rosée :

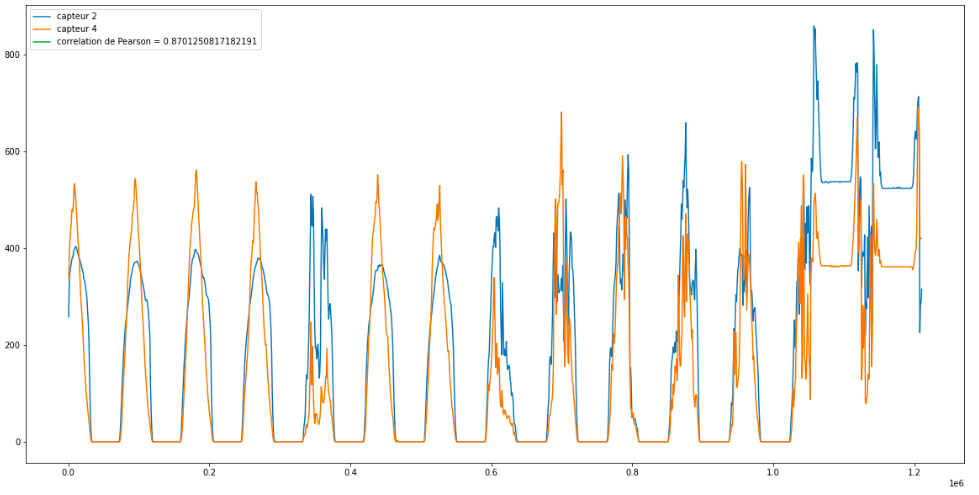
  avec phi l’humidité relative.

Pour calculer l’indice de corrélation (de Pearson) entre deux variables, nous avons utilisé le module scipy : “ scipy.stats.pearsonr(v1,v2)[0]”, avec v1 et v2 les variables.

**BONUS :**

Nous avons réussi à superposer sur le même graphe la courbe représentant l’évolution de la luminosité provenant du capteur numéro 2, en fonction du temps, et la courbe représentant l’évolution de la luminosité provenant du capteur numéro 4, en fonction du temps. Les données ont été prises sur une même portion de temps.

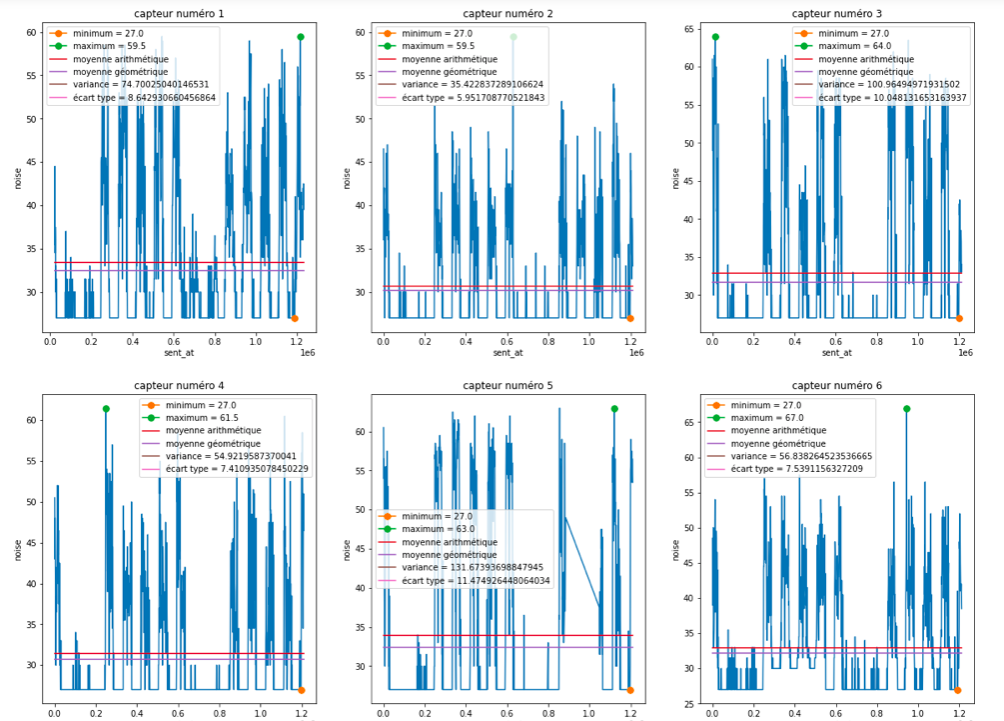
On peut lire l’indice de corrélation de Person sur la légende du graphique.



Cependant, lors du premier essai, le nombre de valeurs des deux listes n’étaient pas les même, un message d’erreur était renvoyé. Or pour pouvoir calculer le coefficient de corrélation, il faut avoir le même nombre de données pour les deux listes de valeurs à comparer.

IV Tracé des courbes et affichages des valeurs statistiques :

Exemple de tracé : évolution du bruit en fonction du temps (“sent\_at”)



On a choisi de répartir les graphiques de la manière suivante : deux à la verticale et trois à l’horizontale, en utilisant la fonction : “plt.subplot(2,3, k)”. Cela qui fait un total de six graphiques correspondants chacun à un capteur en particulier.

On a espacé les graphiques entre eux pour que cela fasse un rendu plus propre, grâce à la fonction : “plt.gcf().subplots\_adjust(wspace=0.3, hspace=0.7)”.

De plus, pour ne pas qu’ils soient les uns sur les autres, on a utilisé la fonction : “plt.figure(figsize=(20,15))”

On a affiché sur chaque graphique, les valeurs statistiques qui lui correspondaient, en les ayant calculées pour chaque capteur au préalable.

Puis on a ajouté un titre et des légendes en fonction des données traitées. : “ plt.title('capteur numéro ' + str(k)” et “plt.xlabel()” et “plt.ylabel()”

On a rencontré un problème cependant, dans les valeurs statistiques affichées, on retrouve la moyenne géométrique. Or pour la calculer, on utilise la fonction log, mais dans le cas des données de la luminosité, le minimum est 0 la courbe y passe régulièrement, or log(0) n’est pas calculable. Cependant, les graphiques s’affichent tout de même ainsi que la légende, mais le message d’erreur aussi...

V Recherche des anomalies :

Notre première idée était de calculer une moyenne logarithmique (pour gommer les pics) sur les différentes données des différents capteurs, mais seulement sur une période de trois jours. Puis de faire défiler les trois jours sur la durée totale de quinze jours. Une fois les moyennes calculées, on les aurait comparées en utilisant un intervalle de confiance. Si elles sortaient de cet intervalle de confiance, fixé en fonction des valeurs des moyennes obtenues, alors il aurait s’agit d’anomalies.

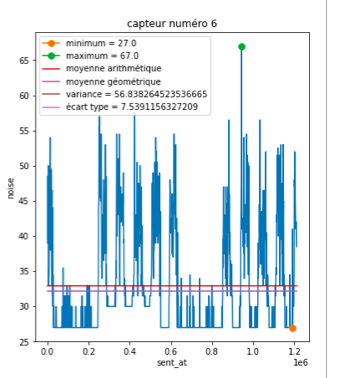
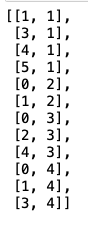
Cependant, nous avons procédé d’une autre manière. Nous avons créé une fonction qui prenait les valeurs de N (liste du nombre de valeurs par capteurs) et qui les divisait (par le nombre que l’on souhaite) mais en portion de temps égales les unes entre les autres, bien que le nombre de valeurs ne soit pas le même.

Ensuite on a séparé les différentes listes en sous-listes, dont on avait besoin pour l’algorithme, en fonction du nombre de division que l’on souhaitait réaliser. On a choisi de faire 5 divisions dans l’algorithme mais cela peut être facilement modifiable.

Après avoir obtenue les sous-listes des listes de données, on a calculé toutes les moyennes logarithmiques de chacune des sous-listes.

Une fois les moyennes obtenues, on les compare entre elles à l’aide d’une valeur de seuil que l’on choisit de prendre à 0,04. Dès lors, si une des moyennes se trouve en dehors de ce seuil, alors elle est considérée comme anormale. Alors le programme renverra une liste de listes. Les sous-listes sont à lire de la manière suivante : [x, y] avec x le capteur (de 0 à 5, donc faire x+1 pour avoir le véritable numéro du capteur) et y la portion de temps ou il y a l’anomalie (également de 0 à 5, car 5 divisions, faire y+1 pour avoir la véritable portion de temps.)

Exemple :

Visuellement, on repère une anomalie allant de 0,25 à 0,5 1e6 seconde sur le graphique, il s’agit de la deuxième portion de temps. Quand on lance le programme, on remarque sur la quatrième ligne qu’il y a bien une anomalie pour le capteur 6 sur la deuxième portion de temps.