Septembre 2020-Novembre 2020

Rapport du projet programmation informatique

Projet programmation informatique

Sujet 2

Projet réalisé par :

Lucie GEULIN et Marion AURIEL

Groupe 15

Projet encadré par :

Monsieur Nassim LAGA

Table des matières

Introduction

L'objectif du projet de programmation informatique était de **déterminer les similarités entre six capteurs suivant les caractéristiques de bruit, de luminosité, de température, d'humidité et du taux de CO_2 lors d'une campagne de mesure au sein d'un bâtiment de bureau.** Ces données étaient contenues dans un fichier .CSV.

Méthode de travail

Pour réaliser ce projet, nous avons choisi de travailler ensemble sur la première partie et de se séparer les tâches lorsque nous aurions une idée concrète de la finalité du projet. Afin de communiquer de manière instantanée, l'utilisation de Github n'a pas été immédiatement mise en place. En effet, des difficultés ont été rencontrée pour travailler avec la plateforme au départ, nous avons donc utilisé pendant plusieurs séances un Google Drive dans lequel nous déposions les nouvelles versions de notre programme. Deux semaines avant le rendu du projet, Lucie s'est occupée d'appréhender l'utilisation de Github et c'est alors que Github fut l'espace de partage pour l'avancée du projet. Nous avons quand même déposé toutes les versions antérieures de notre programme (que nous avions mises au fur et à mesure sur Google Drive) sur Github afin de vous montrer l'évolution de notre programme.

Pour que notre programme et notre rapport final ne soient pas perdus parmi tous ces fichiers, nous avons créé un dossier : *Rendu final* dans le répertoire : *luciegeulin/projet-informatique* contenant uniquement le rapport, le programme final et les données sous format .txt. Le répertoire est disponible à cette adresse : https://github.com/luciegeulin/Projet-informatique .

Démarche

Dans le but de déterminer les similarités entre les différents capteurs, nous avons tout d'abord cherché à importer le document .CSV sur Python et à récupérer ces données dans une liste. (1)

L'objectif du programme étant de comparer les similarités des capteurs, nous avons séparé les données par capteur. Chaque liste capteur contenant les mesures de toutes les caractéristiques (noise, temperature, humidity, lum, CO_2 et date). (2)

Dans la continuité de l'étape précédente, nous avons séparé les caractéristiques (noise, temperature, humidity, lum, CO_2, date) issues de chaque capteur dans des listes différentes. (3)

Nous avons ensuite implémenté le programme permettant de restreindre les données sur un intervalle de temps spécifique. (4)

Ensuite, nous avons programmé toutes les fonctions demandées par le sujet, c'est-à-dire, le minimum, le maximum, la moyenne, la variance, l'écart-type, l'indice humidex et le coefficient de corrélation. (5)

Puis, nous avons tracé les courbes représentant une caractéristique donnée pour chaque capteur, ainsi que ses valeurs statistiques. (6)

Enfin, nous avons programmé une fonction capable d'afficher si deux capteurs étaient similaires par rapport à une caractéristique (bruit, lumière, température, humidité et CO_2). (7)

Finalement, la dernière étape a été la détermination de l'occupation des bureaux. (8)

Programme

1. Récupération des données contenues dans le fichier .csv

La première difficulté rencontrée a été de récupérer les données contenues dans le fichier csv. Nous avons d'abord transformer le fichier .csv en un fichier .txt. Pour cela, nous avons enregistré le fichier .csv donné dans le sujet en un fichier .txt. Ce choix a été réalisé car nous étions plus à l'aise avec ce type de fichiers.
Après l'importation du fichier.txt sur Python, les données contenues dans ce fichier ont été récupérées dans la liste : *donnees*. Cette dernière a également été subdivisée en listes comportant les différentes lignes du fichier .txt, séparée à l'aide de la méthode .split('\t'). La méthode .split('\t') a été utilisée car '\t' est le séparateur des fichiers .txt. La liste *donnees*, est donc une liste contenant chaque ligne du fichier initial dans une liste.

2. Séparation des données par capteurs

La seconde étape fut de séparer dans des listes différentes, les données par capteurs. La dénomination du capteur se trouvant toujours au niveau du deuxième élément de la liste (deuxième colonne), il a suffi de tester pour chaque liste de la liste *donnees*, quel était le deuxième élément (cf. figure 1 ci-dessous). Par exemple, le premier élément de la liste *donnees,* illustrée ci-dessous, sera ajouté à la liste du capteur 1 : *c1.*

```
donnees =[['0', '1', '35.5', '25.8', '55.0', '282', '448', '2019-08-11 17:48:06+02:00\n'],
          ['1', '1', '44.5', '25.5', '55.0', '288', '429', '2019-08-11 18:03:03+02:00\n'],
          ['2', '1', '34.5', '25.5', '55.0', '286', '417', '2019-08-11 18:18:03+02:00\n'],
          ['3', '1', '37.5', '25.5', '54.5', '282', '433', '2019-08-11 18:33:03+02:00\n'],
          ['4', '1', '36.0', '25.3', '55.0', '274', '403', '2019-08-11 18:48:03+02:00\n'],
          ['5', '1', '30.0', '25.3', '55.0', '254', '410', '2019-08-11 19:03:03+02:00\n'],
          ['6', '1', '33.0', '25.3', '55.5', '234', '407', '2019-08-11 19:18:03+02:00\n'],
          ['7', '1', '30.0', '25.3', '56.0', '200', '407', '2019-08-11 19:33:03+02:00\n'],
          , [.....],[.....]                                                    .,[.....]]
```

Figure 1: Exemple d'une partie de la liste: donnees

3. Séparation par caractéristiques des données de chaque capteur

De la même façon que précédemment, nous avons récupéré pour chaque capteur les différentes caractéristiques dans des listes différentes à l'aide de leurs indices. Par exemple, le bruit se trouvait en première position de chaque liste capteur (première colonne).

Pour la liste des temps, un « +02 :00\n » était présent après chaque date (cf. figure 1 ci-dessus). Nous n'avions pas besoin de ces caractères pour notre étude donc nous les avons retirés lors de la création des listes comportant les dates et heures à l'aide du slicing.

4. Option : Intervalle de temps spécifiques.

Le sujet proposait de réaliser une option permettant de choisir sur quel intervalle nous souhaitions analyser les mesures. Afin que le commande d'affichage demander l'intervalle spécifique, la commande : *input*() a été utilisée.

Ensuite, il a fallu restreindre chaque liste de caractéristiques de chaque capteur sur cet intervalle. Pour cela, nous avons créé de nouvelles listes par compression de liste. La nouvelle liste des temps fut réalisée par comparaison entre les éléments de la liste de temps initiale et les dates demandées. Dans le but de restreindre les autres listes, nous avons récupéré les indices de début et de fin des listes de temps de chaque capteur, et nous avons fait du slicing entre ces indices dans les listes des caractéristiques initiales. (Cf. figure 2)

```
c1_n=[35.5, 44.5, 34.5, 37.5] # Liste avant restriction

c1_d=['2019-08-14 19:32:51', '2019-08-15 10:32:49',  #Liste avant restriction
      '2019-08-15 12:17:48', '2019-08-15 15:02:48']


c1_date= ['2019-08-14 19:32:51', '2019-08-15 10:32:49']  #Liste après restriction:
                                                          réduite aux deux
                                                          premiers éléments


c1_noise=c1_n[0:2]   #Liste restreinte à l'aide du slicing
```

Figure 2: Exemple de restriction d'une liste de caractéristique

Afin de vérifier la cohérence entre les dates demandées et celles des mesures réalisées, nous avons utilisé la commande *assert*. L'utilisateur doit rentrer des dates comprises entre le 11 août 2019 et le 25 août 2019 pour que les dates soient valides.

5. Programmation des différentes fonctions demandées

a. Valeurs statistiques

Nous avons tout d'abord écrit de façon classique les fonctions calculant les différentes valeurs statistiques d'une liste (maximum, minimum, écart-type, moyenne, variance et médiane). Pour le calcul de la moyenne, nous avons décidé de choisir la moyenne arithmétique classique car elle nous semblait être la plus parlante et la plus pertinente. Pour le calcul de la médiane, nous avons dû au préalable trier la liste par ordre croissant. Nous avons utilisé pour cela un algorithme de tri par insertion

car cet algorithme est efficace sur des listes de la taille de celles traitées dans notre projet (complexité en *O(n²)* avec n la longueur de la liste).
Nous avons ensuite écrit des programmes renvoyant chacun une courbe ou un point légendé(e) correspondant à la valeur statistique associée.

b. Indice humidex

Nous avons trouvé après des recherches les formules suivantes pour calculer l'indice humidex :

$$Humidex = T + 0.5555[6.11 * e^{5417,7530\left(\frac{1}{273,16} - \frac{1}{273,15 - Trosée}\right)} - 10]$$

Avec : T la température mesurée en degré
$T_{rosée}$ le point de rosée, calculé comme suit :

$$Trosée = \frac{b * \alpha(T,\varphi)}{a - \alpha(T,\varphi)} \text{ avec } \alpha(T,\varphi) = \frac{a * T}{b + T} + ln\,\varphi$$

Avec : φ l'humidité relative
T la température en degré
a=17,27
b=237,7

Ces formules ont été transcrites sur Python pour accéder à l'indice humidex d'un couple humidité relative/température.

c. Coefficient de corrélation

Nous avons trouvé que l'indice de corrélation pouvait être calculé de la façon suivante :

$$cor(X,Y) = \frac{cov(X,Y)}{\sigma(X) * \sigma(Y)}$$

Avec : $\sigma(X)$ et $\sigma(Y)$ les écarts-types de X et Y
$cov(X,Y)$ la covariance de X et de Y

Cette dernière se calcule de la façon suivante :

$$cov(X,Y) = \frac{1}{N} * \sum_{i=1}^{N} \left(Xi - \underline{X}\right) * (Yi - \underline{Y})$$

avec N le nombre de valeurs des listes X et Y (supposées de même longueur), et $\underline{X}$ et $\underline{Y}$ les moyennes des listes X et Y.
Cette formule a été transcrite sur Python afin d'accéder au coefficient de corrélation de deux listes. Le programme renvoyant la covariance de deux listes a une complexité en *O(n²)* et celui renvoyant l'indice de corrélation a aussi une complexité en *O(n²)*, avec n la longueur des deux listes.

6. Tracé des courbes en fonction du temps.

La première difficulté rencontrée pour le tracé des courbes fut de choisir une base de temps adaptée. Après observation de la liste des dates, nous avons remarqué que les mesures étaient réalisées à intervalle de temps régulier (toutes les 15 minutes). Nous avons donc au départ choisi de

ne pas nous soucier du temps, et de prendre en abscisse une liste contenant le nombre de mesures réalisées.
Cependant, cette technique ne nous permettait pas de comparer sur un même graphique les données de deux capteurs différents, car les mesures n'étaient pas toujours réalisées au même instant.
C'est pourquoi nous avons donc décidé de convertir chaque date en seconde. La référence prise serait 0 pour la première date c'est-à-dire le 11 août 2019 à 0h00. Les mesures étant réalisées entre le 11 août 2019 et le 25 août 2019, (mesures réalisées durant un même mois), il a été ajouté 24h converties en secondes, pour chaque nouvelle journée passée par rapport à la référence (le 11 août). La complexité de cet algorithme est en *O(n)*.
Ensuite, nous avons utilisé la commande *matplotlib.pyplot.plot(temps, données)*, afin de tracer pour tous les capteurs une même caractéristique en fonction du temps.
Nous avons ensuite modifié l'axe des abscisses afin qu'il affiche les jours correspondant, donnée plus pertinente que le « nombre » de secondes qui était automatiquement affiché.

7. Affichage des similarités

L'affichage des similarités a évolué suite aux échanges avec les professeurs. Au départ, nous souhaitions que le programme indique visuellement les similarités. Ce premier programme comparait les similarités à 95% de deux capteurs vis-à-vis de la moyenne, la médiane, l'écart-type, la variance, le coefficient de corrélation ou encore l'indice humidex.
Nous avions utilisé la méthode *.format* afin d'afficher les différentes similarités. Cependant, ces similarités n'étaient pas représentatives de la réalité, car deux capteurs pouvaient par exemple obtenir une même moyenne avec des valeurs totalement disparates.

Nous avons donc cherché à calculer la distance entre la moyenne des mesures de deux capteurs pour une même caractéristique sur des intervalles de temps d'une heure. En effet, nous avons remarqué que la valeur des mesures changeait de manière significative au bout d'une heure.
Pour cela nous avons procédé en plusieurs étapes.

Tout d'abord, une liste : *base_temps* a été créée afin de récupérer les indices des points dans un même intervalle pour tous les capteurs. Ces indices ont été utilisés pour découper les listes de caractéristiques de chaque capteur dans ces intervalles. Enfin, nous avons déterminé la moyenne des données sur ces intervalles afin de pouvoir calculer la distance moyenne entre deux courbes sur un même intervalle de temps.
Ceci est illustrée par la figure 3 suivante.

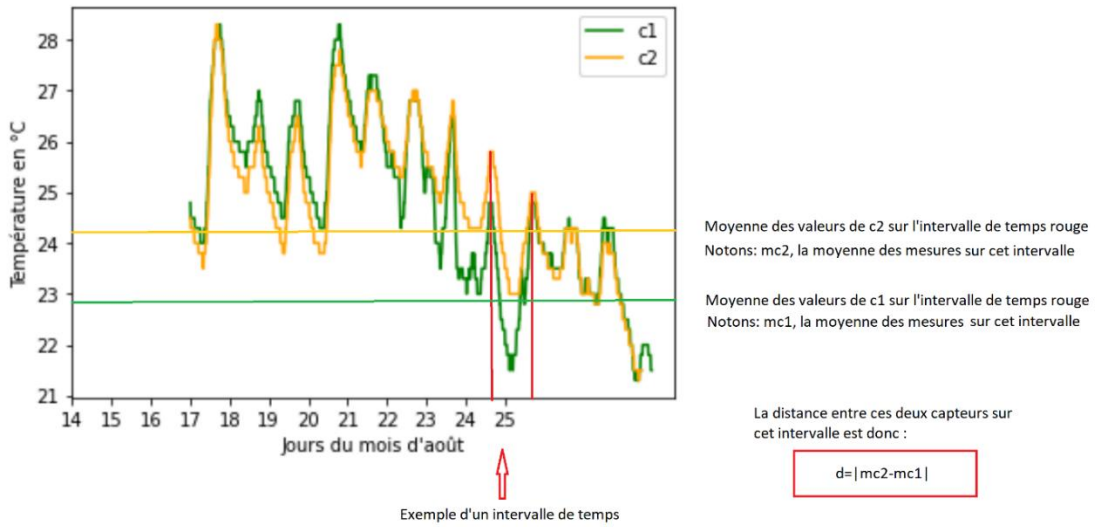

Figure 3: Calcul de la distance entre les capteurs 1 et 2 vis-à-vis de la température sur un intervalle de temps

Nous avons ensuite déterminé la distance moyenne de la totalité de la liste.
Cependant, une fois encore, la moyenne n'était pas l'outil le plus adéquat pour comparer les similarités. Nous nous sommes donc orientés vers un calcul de variance autour de la moyenne, mais nous ne savions pas quelle valeur de référence paraissait pertinente étant donné la variance.

Finalement, nous avons choisi de travailler avec un calcul de métrique relatif aux distances. Pour cela nous avons cherché à normaliser les distances obtenues pour chaque capteur, d'après la formule suivante :

Soit X une liste
i∈[|0,len(X)|]

$$X[i] = \frac{\mathrm{X[i]} - \min(\mathrm{X})}{max(X) - \min(X)}$$

De plus, nous avons créé une liste *D_capteurs* qui répertorie pour chaque caractéristique la distance des capteurs deux à deux. L'exemple suivant illustre la liste *D_capteurs* pour une seule caractéristique.

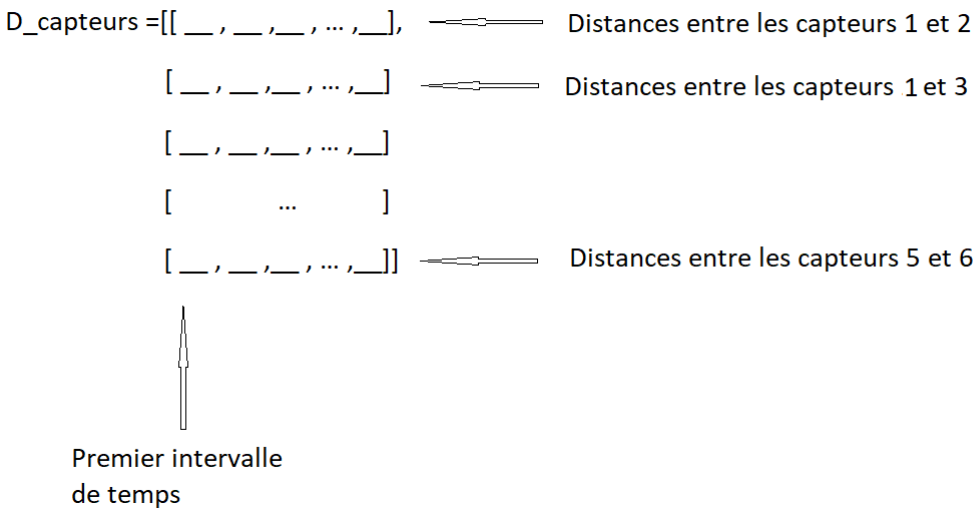

Figure 4: Schéma représentatif de la normalisation sur la liste D_capteurs

La normalisation des distances se fait donc sur chaque intervalle de temps afin que les distances soient comparables. D'après la formule utilisée pour la normalisation, le maximum et le minimum sont différents pour chaque intervalle de temps, c'est-à-dire pour chaque colonne. Les valeurs obtenues sont donc comprises entre 0 et 1 et sont d'autant plus proche de 0 si leur distance au minimum est faible.

Après avoir normalisé toutes nos listes, nous avons pu déterminer sur l'intervalle entier la distance moyenne normalisée en pourcentage entre tous les capteurs pour une même caractéristique. Par exemple, la distance moyenne de 14 % entre les capteurs 5 et 6 vis-à-vis du CO_2 signifie que l'écart par rapport à la distance minimale entre deux capteurs est égal à 14% du plus grand écart. Nous avons pu en déduire que les capteurs avec les distances moyennes les plus faibles étaient les plus similaires. La suite de programme ainsi créée pour déterminer les similarités est en $O(n^3)$.

Pour déterminer les seuils de similarités, nous nous sommes aidées du tracé des courbes pour chacune des caractéristiques. Nous avons choisi les seuils tels que les courbes évoluant de la même façon et étant proches entre-elles soient considérées comme similaires. Par exemple, un seuil de 20% a été choisi pour la caractéristique du bruit.

8. Bonus : Programme de détermination de l'occupation des bureaux

Nous avons séparé ce programme en deux parties. En effet, afin de déterminer les moments où les bureaux étaient occupés, nous avions besoin de connaître les horaires des bureaux et de connaître les jours correspondant aux week-ends.

a. Horaires d'occupation des bureaux

Pour trouver les horaires des bureaux, nous avons choisi d'étudier la luminosité de ceux-ci car cette donnée nous semblait la plus pertinente. En effet, le bruit, la température, le taux d'humidité et le CO_2 ne dépendent pas uniquement de la présence de personnes dans les bureaux alors que la lumière est nécessaire au travail.
Pour comparer la luminosité à différents moments de la journée, nous devions trouver un seuil permettant de trancher sur la possibilité de travailler ou non avec cette quantité de lumière. D'après différentes sources, la luminosité dans les lieux de travail est généralement supérieure à 150 lux. Nous avons donc pris cette valeur comme référence : lorsque la luminosité est supérieure à cette valeur, nous considérons que les bureaux sont occupés.

Nous avons écrit dans un premier temps un programme *occupation_bu* qui renvoie la liste des horaires pour lesquels la luminosité est supérieure à ce seuil de 150 lux. Cette liste correspond donc aux horaires d'occupation des bureaux pour un capteur. Nous avons ensuite fait évoluer ce programme pour qu'il renvoie uniquement l'heure de début et de fin d'occupation des bureaux chaque journée. Cet algorithme est de complexité quadratique.
Le programme renvoie des listes cohérentes pour les capteurs 1 et 3. Cependant, on remarque que pour les autres capteurs (2,4,5 et 6), la luminosité ne diminue pas lors des deux dernières nuits. Elles sont donc comptabilisées par notre programme comme des horaires de travail. Cette incohérence peut être due à un oubli d'éteindre la lumière deux nuits de suite, ou à un disfonctionnement des capteurs. Cette deuxième option nous semble plus probable car les valeurs de luminosité pour ces mêmes capteurs lors des deux derniers jours sont beaucoup plus importantes que celles au début des mesures (par exemple avec le capteur 6, le 14 août à 10h30 la luminosité était de 272 lux alors qu'elle est de 728 lux le 25 août à la même heure).

b. Détermination des jours de la semaine

Afin de déterminer les jours d'occupation des bureaux, nous avons utilisé la donnée du bruit car elle variait de façon significative deux jours sur sept, ce que nous avons interprété comme la variation due aux week-ends.

Nous avons tout d'abord écrit un programme : *bruit_jour*, qui sépare les mesures du bruit par journée pour un capteur donné : il renvoie une liste de sous-listes contenant chacune les mesures du bruit pour une journée. Il a une complexité en *O(n)*. Ce programme a été utilisé par *moye_bruit* qui renvoie une liste contenant les moyennes du bruit pour chaque jour pour un capteur donné.

Enfin, ces deux programmes ont été utilisés dans notre programme final : *weekend*, renvoyant une liste contenant les jours de la semaine et une autre les jours du week-end. A l'aide des deux programmes précédents, nous avons pu implémenter une condition spécifiant que si la moyenne du bruit pendant une journée était supérieure à 29,5 décibels, alors cette journée était en semaine sinon elle tombait un week-end. Ce programme a une complexité en *O(n)*.

Enfin, nous avons rassemblé ces deux programmes : *occupation_bu* et *weekend* dans un programme : *horaire_sem*. Ce dernier permet de retirer de la liste renvoyée par *occupation_bu* les

jours retenus comme étant les samedis et dimanche grâce au programme weekend. Il renvoie donc une liste présentant pour chaque jour de la semaine l'heure de début et de fin de travail.

Conclusion

Pour conclure, d'après le programme implémenté, les deux seuls capteurs similaires vis-à-vis du bruit sont les capteurs 2 et 4. Cependant, par rapport à la caractéristique de température des similarités sont notables entre les capteurs 1 et 3, 2 et 4, 2 et 6, 3 et 4 ainsi qu'entre les capteurs 4 et 5. Les capteurs 2 et 3, 2 et 4 ainsi que 3 et 4 sont similaires vis-à-vis de l'humidité. De même, la lumière rassemble les capteurs 1 et 5, 1 et 6 ainsi que 2 et 6, tandis que le $C0_2$ montre des similitudes entre les capteurs 1 et 3, 2 et 4 et enfin 5 et 6.

Finalement, les capteurs 2 et 4 sont les deux capteurs les plus similaires vis-à-vis de toutes les caractéristiques.

Ce projet nous a permis d'apprendre à travailler avec Git et Github. Même si nous avons eu du mal à les maîtriser au départ, ils ont été très utiles à la fin de notre projet. Ils nous ont permis de travailler en groupe sans avoir de problème de transmission des informations. Cette nouvelle connaissance nous sera certainement utile pour le projet Data par exemple. Ce projet de programmation nous a aussi permis de progresser en informatique grâce à des échanges de connaissances entre nous ou grâce à des recherches. Enfin, il a représenté pour nous une nouvelle occasion d'apprendre à travailler en groupe, à communiquer et à utiliser nos complémentarités. Nous avons réussi tout le long du projet à discuter régulièrement de nos avancées et des problèmes que nous rencontrions. Nous nous sommes organisées selon nos domaines de compétences afin de travailler en autonomie et de mener à bien ce projet.