Rapport du projet de programmation informatique

Hugo Baldo et Agathe Catenot

Le projet avait pour but de nous faire travailler le design d'algorithmes, leur implémentation et l'utilisation d'outil de gestion de code informatique. Ce sujet nous propose alors d'explorer et d'analyser des données issues d'une campagne de mesure au sein d'un bâtiment de bureau. Les données sont proposées par Kandu et sont composées entre autres de la mesure de la température ambiante (°C), de l'humidité relative (%), du niveau sonore (dBA), du niveau lumineux (lux), de la quantité de CO2 (ppm).

1) Utilisation de Github

Github est une plateforme en ligne qui permet aux utilisateurs de stocker et de partager, publiquement ou non, le code qu'ils créent.

Ainsi, on commence par créer un dossier contenant le projet sur notre ordinateur à l'aide de la commande mkdir sur GitCMD. On définit ensuite l'identité de l'utilisateur à l'aide de la commande git config. Puis on ajoute le fichier python dans le dossier et on enregistre les modifications faites sur le fichier à l'aide des commandes git add puis git commit. De cette manière chaque membre du groupe peut modifier le projet et enregistrer ses modifications depuis son ordinateur dans le fichier qu'il a créé c'est-à-dire le dépôt local.

Pour se partager les fichiers, il faut ensuite créer un dépôt distant en créant un repository sur Github dont on autorise l'accès à l'ensemble des collaborateurs. Pour que chaque membre du groupe dépose ses fichiers sur le dépôt distant, il faut créer un lien entre le dépôt local et le dépôt distant grâce aux commandes déjà données sur Github. Enfin, chaque membre du groupe envoie du dépôt local au dépôt distant ses modifications du fichier en utilisant la commande git push.

En somme Github nous permet d'échanger simplement les fichiers du projet et de visualiser rapidement leur évolution puisque la commande git log donne dans l'ordre chronologique l'ensemble des commit qui ont été effectués par les différents utilisateurs.

2) Analyse des données

La première étape de l'analyse des données consiste à ouvrir le fichier csv et d'en lire les données. Pour cela on importe les bibliothèques csv et panda de python.

```
>import pandas as pd
>import csv
```

L'analyse de ces données peut se faire grâce aux outils statistiques. La fonction describe() de panda permet alors de calculer les statistiques du tableau dataframe qui nous était donné.

```
df = pd.DataFrame(data)
stat_df = df.describe()
print(stat_df)
```

	id	noise	...	lum	co2
count	7880.000000	7880.000000	...	7880.000000	7880.000000
mean	3.468528	32.508185	...	164.291371	449.255330
std	1.711084	8.648065	...	202.650402	41.691629
min	1.000000	27.000000	...	0.000000	343.000000
25%	2.000000	27.000000	...	0.000000	417.000000
50%	3.000000	27.000000	...	84.000000	444.000000
75%	5.000000	36.500000	...	284.000000	470.000000
max	6.000000	67.000000	...	1418.000000	705.000000

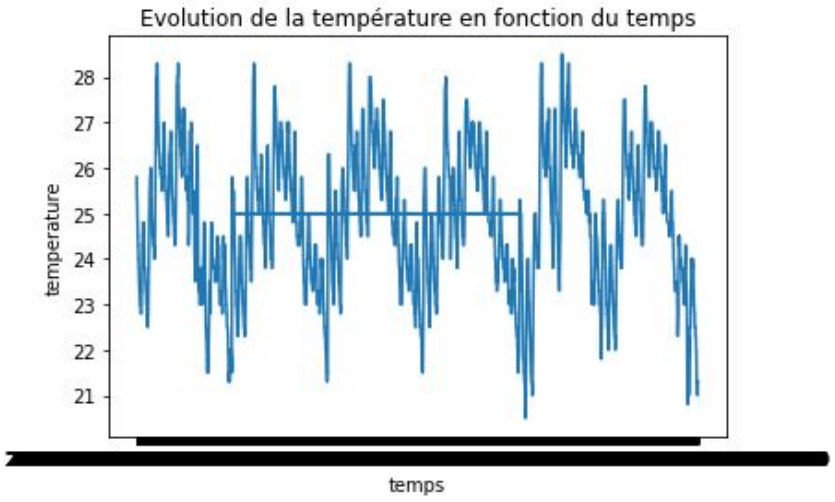
On obtient ainsi la moyenne, l'écart-type, le nombre d'observations, le minimum, le maximum et les quartiles.

L'algorithme devait ensuite comprendre des courbes montrant l'évolution d'une variable en fonction du temps. Pour cela on importe la bibliothèque Matplotlib à l'aide de la commande suivante.

```
> import matplotlib.pyplot as plt
```

On trace ensuite les courbes demandées.

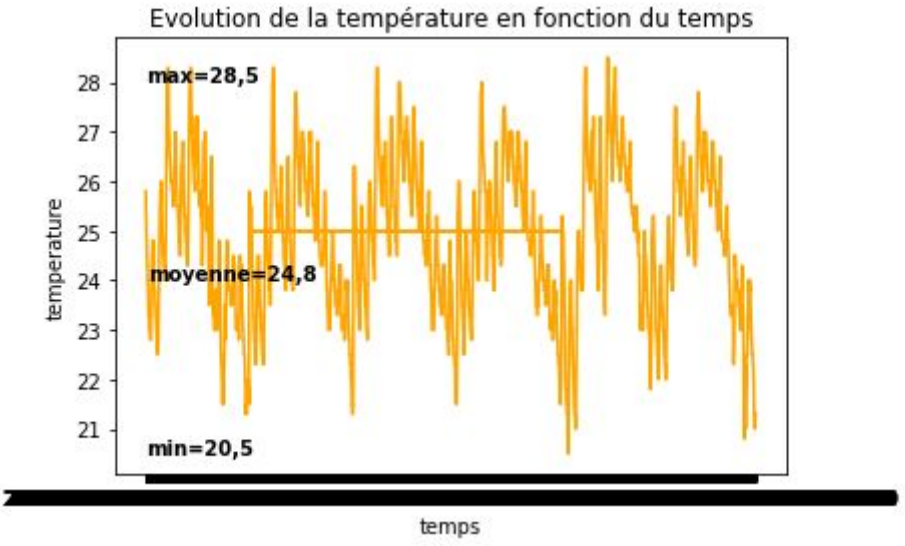
```
x=df['sent_at']
y=df['temp']
plt.plot(x,y)
plt.title('Evolution de la température en fonction du temps')
plt.xlabel('temps')
plt.ylabel('temperature')
plt.show()
```



Pour les autres colonnes il suffit de changer ‘temp’ dans la deuxième ligne du script par le nom de la colonne que l’on souhaite voir apparaître sur la courbe.

Il faut ensuite afficher les valeurs statistiques sur la courbe. Pour cela on utilise la fonction text de la bibliothèque matplotlib. On peut ainsi choisir la position du texte sur la courbe, sa couleur ou encore le style de la police.

```
#affichage des valeurs statistiques sur la courbe précédente
x=df['sent_at']
y=df['temp']
plt.title('Evolution de la température en fonction du temps')
plt.xlabel('temps')
plt.ylabel('temperature')
plt.text(2,20.5,'min=20,5',fontweight='bold')
plt.text(2, 28,'max=28,5',fontweight='bold')
plt.text(20, 24,'moyenne=24,8',fontweight='bold')
plt.plot(x,y,'orange')
plt.show()
```



Il est par ailleurs demandé de calculer l’indice humidex. Il s’agit d’une formule utilisée par les météorologues pour intégrer les effets combinés de la chaleur et de l’humidité. Il se calcule grâce à la formule suivante :

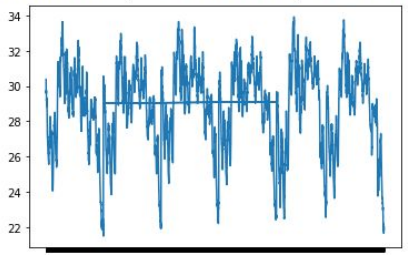
$$\text{Humidex} = \text{Air temperature} + \frac{5}{9} \times (6.112 * 10^{7.5 \times \frac{T}{237.7+T}} \times \frac{H}{100} - 10)$$

Avec T la température et H l’humidité relative.
On écrit ainsi un algorithme qui prend en entrée le dataframe, qui calcule pour chaque ligne l’indice humidex et renvoie une liste qui contient l’indice humidex de chaque ligne du dataframe. On trace également la courbe montrant l’évolution de l’indice humidex en fonction du temps.

```
humidex=[]
colonne_temp=data["temp"]
colonne_humidity=data["humidity"]
L=len(data)

for i in range(L):
  h= colonne_temp[i] + (5/9)*(6.112*(colonne_humidity[i]/100)*10**(7.5*(colonne_temp[i]/(237.7+colonne_temp[i])))-10)
  humidex.append(h)
print(humidex)
x=humidex
y=df['sent_at']
plt.plot(y,x)
plt.show()
```

```
[30.374664534947826, 29.89637343499578, 29.89637343499578, 29.805901353263497, 29.5790380816241, 29.5790380816241, 29.668443478325734,
```



Enfin, on cherche à déterminer l'indice de corrélation entre un couple de variables.

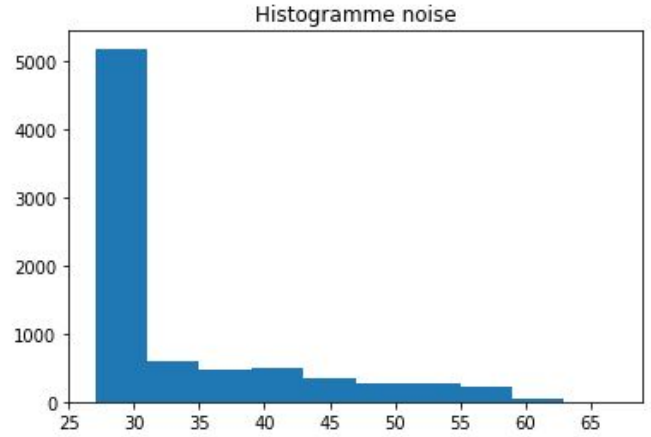
3) <u>Réponse au sujet</u>

L'objectif du sujet était de détecter d'éventuelles anomalies dans les données puis de proposer un algorithme permettant de les relever et de les montrer sur une courbe.

Pour cela on commence par tracer un histogramme des valeurs prises par une colonne donnée ici on choisit la colonne noise.

```
#Histogramme de la colonne noise
plt.hist(data.noise)
plt.title("Histogramme noise")
```

```
Text(0.5, 1.0, 'Histogramme noise')
```



Cette méthode permet de s'assurer que les valeurs sont regroupées autour de la moyenne et ainsi de considérer que les valeurs "éloignées" de la moyenne sont des anomalies.

On écrit alors l'algorithme qui détecte les valeurs de la colonne noise qui sont supérieures à 1,8 fois la moyenne de la colonne et qui renvoie ses valeurs dans une liste appelée anomalie_noise.

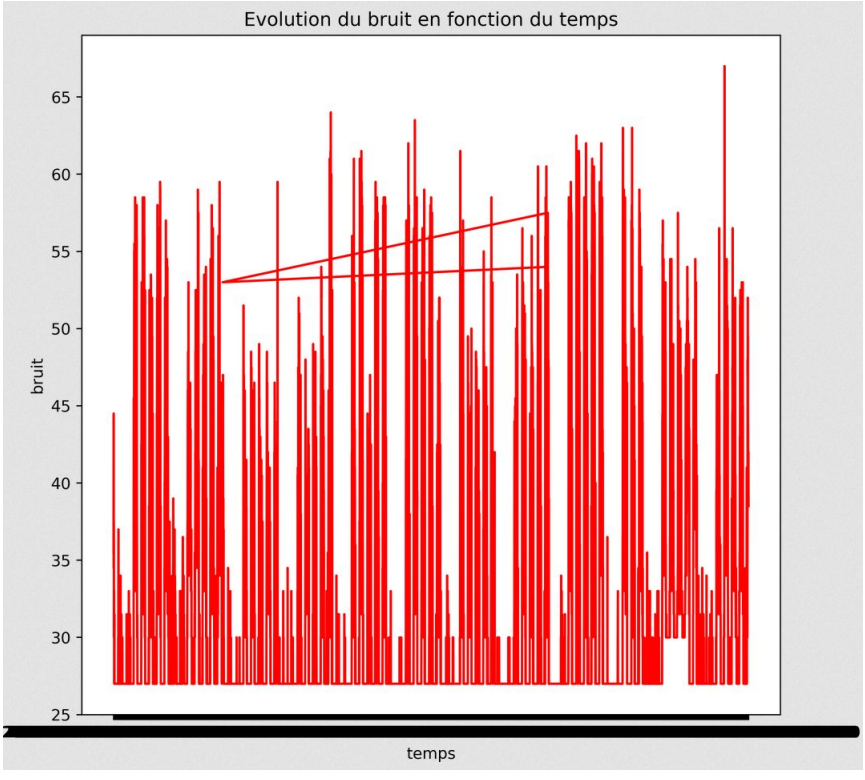
```
#Détection des anomalies pour la colonne "noise"
colonne_noise=data["noise"]
print(colonne_noise)
anomalie_noise=[]
for i in range(len(colonne_noise)):
  if colonne_noise[i]>1.8*mean_df["noise"]:
    anomalie_noise.append(colonne_noise[i])

print(anomalie_noise)
len(anomalie_noise)
```

On souhaite montrer ses anomalies sur une courbe. Pour cela on utilise la bibliothèque matplotlib de python comme suit:

```
plt.figure(0)
plt.figure(figsize=(8,8), dpi=500)
plt.plot(data["sent_at"],data["noise"], 'r')
plt.title('Evolution du bruit en fonction du temps')
plt.xlabel('temps')
plt.ylabel('bruit')
plt.show(0)
```

On obtient ainsi la courbe suivante.



A partir de la colonne lum du fichier csv nous pouvons déterminer les horaires d'occupation des bureaux. En effet, lorsque la lumière s'allume pour la première fois, lum est alors différent de 0, cela signifie que le bureau est occupé et nous pouvons récupérer la date de cet événement grâce au capteur. De plus, lorsque la lumière est éteinte, lum vaut 0 et le bureau est inoccupé. On écrit alors l'algorithme suivant :

```
def horaire2(data):
  for col in data :
    if col=="lum":
      continue
      for i in df.itertuples():
        if data.iloc[i][col]==0:
          debut=data[sent_at, i]
        elif data.iloc[i][col]!=0 :
          fin=data[sent_at,i]
      return(debut,fin)

print(horaire2(data))
```

4) <u>Conclusion :</u>

En somme, ce projet nous a permis d'imaginer des algorithmes permettant d'analyser les données d'un fichier csv. L'utilisation d'un logiciel tel que github nous permet également de travailler sur un même projet à distance tout en gardant l'historique de nos modifications.