IA School
M2 IA et Management

Projet Big Data

Mehdi KARECH

Creation d'une pipiline de données pour la prévision de la météo

---

Superviseur :

Karim KOUKI

Contents

Chapter 1

Introduction

Dans ce projet nous allons nous intereser à faire la prévision de la météo pour la ville de Paris (savoir s'il va pleuvoir par exemple) en implémentant une pipeline de données qui récupere ces dernieres, en utilisant l'api du site (https://openweathermap.org/), l'api permet de récuperer des données tel que la température maximale, minimale et moyenne, la pression, la vitesse du vent, l'humidité et enfin la meteo du jour, la récuperation est faite sur la derniere année (on peut pas récuperer plus sans faire un abonnement) sur des intervalles de points d'une heure.

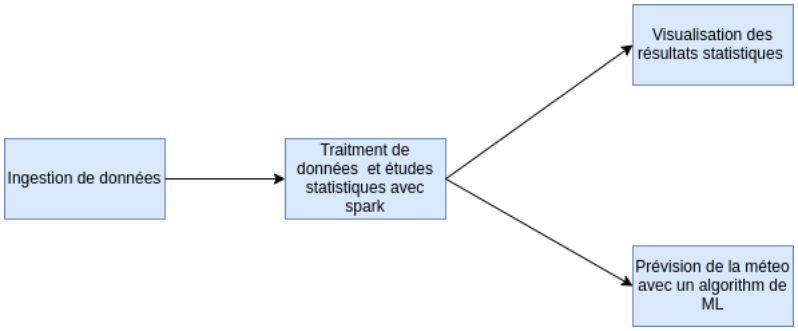

Figure 1.1: Pipeline général.

Ingestion de données :
Pour l'ingéstion de données nous allons utiliser un producer kafka qui va récupérer les données en utilisant l'api (openweather map), puis un kafka consumer va construire un fichier csv à partir des données injécté.
Traitement de données et études statistique avec Spark :
Pour l'étude statistique du data set construits nous allons utilisé un script python avec pyspark
Visualisation des résultats statistiques Un autre script s'occupera de la visualisation des données qui sera dynamique selon les données injéctés.
Partie ML : Une fois le dataset finie et qu'on aura récupérer toutre les données un modéle de Machine learning (une régression logistique) sera lancé sur le dataset finale, pour permettre la prévision de la météo.

Chapter 2

Architécture de la solution

Dans ce chapitre nous allons présenter en détails chaque composants de notre pipeline

2.1 Ingestion de données

Pour l'ingéstion de données nous avons utilisé l'api d'open weather pour récupérer les données, l'api permet seulement de récupérer les données de la derniere année (les douzes derniers mois), la récupération se fait avec des intervalles d'une heure.

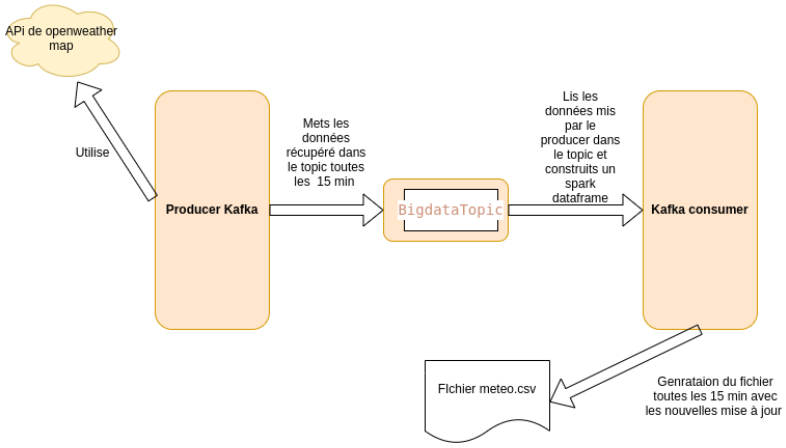

Figure 2.1: Ingestion de données

configuration de kafka : Pour utiliser notre architecture nous devons d'abord configurer kafka, pour commencer nous lançons le serveur zookeeper

```
$ sudo systemctl start zookeeper
```

Figure 2.2: Lancement de zookeeper

Puis Kafka :

```
mehdi@mehdi-G5-5587:~$ sudo systemctl start kafka
```

Figure 2.3: Lancement de kafka

une fois le serveur kafka lancé on crée le topic **BigDataTopic** qui servira de canal de communication entre le produceur kafka et le consumer kafka.

Une fois le topic créé le **Producer kafka est lancé**, ce dernier est implémenter dans le fichier **producer.py**. Dans ce script les données sont récupérés en utilisant l'api cité precedement, le script est implémanter de sorte qu'on récupére les données d'un mois toutes les 15 minutes et on les mets sous format **.json** dans le topic **BigDataTopic** les données acquis sont (time stamp, température maximale, température minimale, température moyenne, pression,humidité, vitesse de vent, et météo)
Aprés le lancement du script **producer.py**, le script **consumer.py** est exécuté, ce dernier récupere les objets **.json** à partir de **BigDataTopic** , construit un dataframe et crée un fichier **meteo.csv** toute les 15 min (l'ancien fichier est écrasé et remplacé par le nouveau fichier et le dataframe est mis à jour à chaque fois)

2.2 Traitement de données

Une fois le premier fichier **meteo.csv** créé, le script **traitement.py** est lancé, dans ce dernier des statisitcs sont efféctué sur chaque colonne du dataset, et le script et réexécuter toute les 15 minutes pour prendre en compte la nouvelle mise à jour des données, le script génere un fichier **stats.csv** qui servira à faire la visualisation dans l'étape suivante

2.3 Visualisation

Notre solution a été développé sous linux, et on voulait conncter **Tableau** avec **spark**, On a eu des erreurs d'intégration qu'on a pas pu malheureusement régler à temps, on a décider donc de faire nos visualisations en utilisant **matplotlib** et **seaborn** en l'intégrant dans un script **visualisation.py**, le script de visualisation est également éxecuter toutes les 15 min une fois le premier fichier **stats.csv** généré, on peut trouver quelques visualisations ci-dessous

2.4 Partie ML

Une fois toute les données obtenue (un dataset finale est obtenue avec toute les données sur une année), le script **ML.py** est lancé manuellement, le script applique une régression logistique sur le dataset finale obtenue, pour donner la météo à une heure donnée (4 classes possibles, tout les attributs sont numrique donc pas de one hot encoding à effectuer), nous avons utilisé mllib pour entrainer notre modele sur 80% des données et le tester sur les 20% restantes, et on a obtenue une accuracy de 67%

2.5 Orchestration

Pour l'orchestration des données, et vue qu'on a utilisé linux comme environnement de developpement, on a utilisé **cron** pour lancer les scripts touts les 15 minutes, seuls les

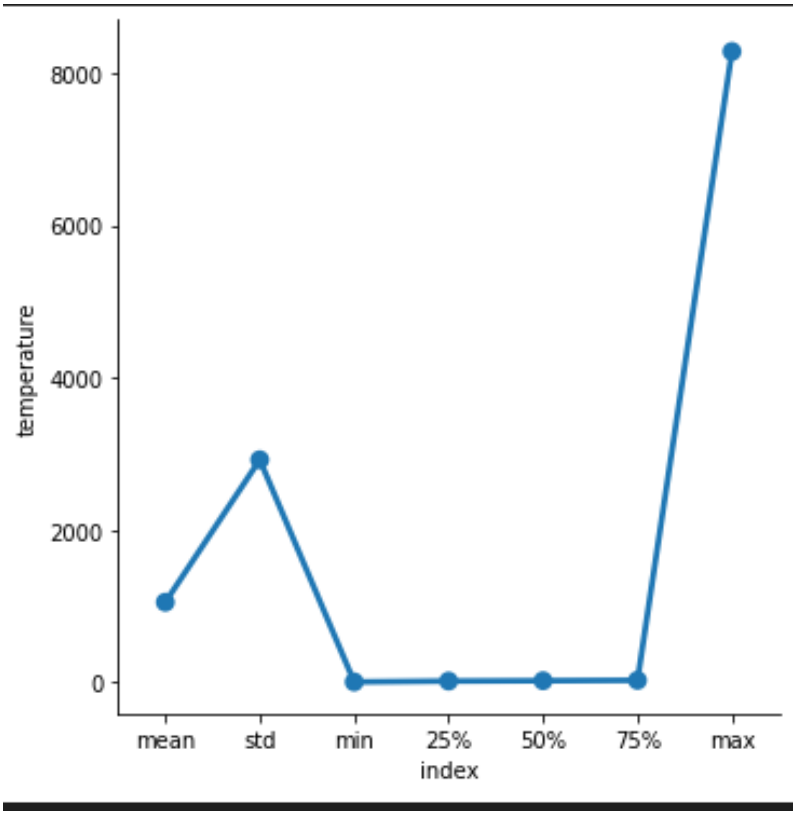

Figure 2.4: Distribution de la population température

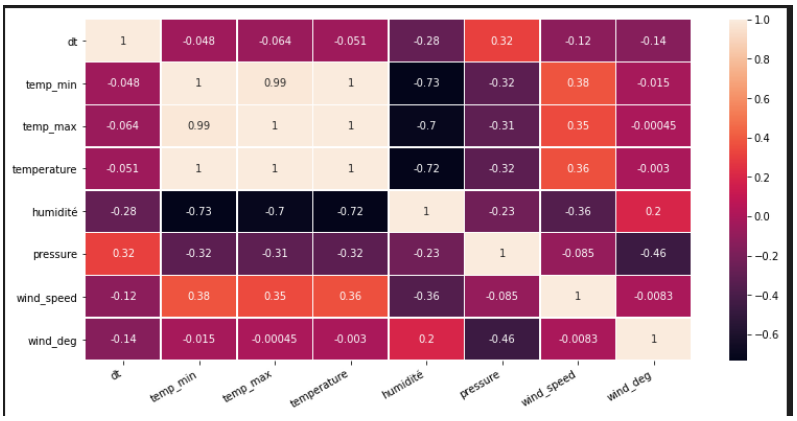

Figure 2.5: Coorelation des données

scripts **producer.py**, **consumer.py** et **ML.py** son exécuter manuelement.
Une meilleure maniére de faire serait d'intégrer Airflow ou Luigi pour orchestrer tout le processsus.(j'ai pas eu le temps de chercher plus par rapport à cet aspect vu que j'ai travaillé tout seul)

Chapter 3

Conclusion

Ce projet a été trés benifique pour moi et j'ai vraiment appris beaucoup de choses, j'aurais aimé pouvoir intégrer tableau et Airflow mais malheureursement j'ai pas eu le temps pour le faire, je vous remercie pour vos effortx tout au long du cours et je vous souhaite une trés bonne continuation pour la suite.