Rapport du Projet Programmation Fonctionnelle : *Analyse et Prédiction de la Pollution Urbaine, Traitement Big Data, GraphX et Machine Learning avec Scala*

ING2 GMA

Auteurs :

- ✓ *Wendpêgré Hugues Anselme KABORE*
- ✓ *Athittyan BALAGOWRIYAN*
- ✓ *Abdoul Aziz SOW*
- ✓ *Papa Moussa NDONG*

Année universitaire : 2024 – 2025

Table des matières

Introduction

La pollution atmosphérique constitue aujourd'hui un enjeu majeur de santé publique et d'environnement, notamment dans les zones urbaines fortement densifiées. Les émissions issues du trafic routier, des activités industrielles et du chauffage contribuent à la dégradation de la qualité de l'air, avec des conséquences directes sur la santé des populations et sur les écosystèmes.

L'analyse de ces phénomènes repose sur l'exploitation de données environnementales volumineuses et hétérogènes. Le traitement de telles données nécessite des outils capables de gérer de grands volumes d'informations tout en garantissant des performances élevées. Apache Spark s'impose alors comme une solution adaptée grâce à son modèle de calcul distribué et à son support de la programmation fonctionnelle.

Ainsi donc comment analyser, modéliser et prédire la pollution urbaine à partir de données environnementales massives en exploitant la programmation fonctionnelle et les capacités de traitement distribué d'Apache Spark ?

L'objectif principal de ce projet est de concevoir une application capable de mesurer, analyser et prédire la pollution urbaine. Plus précisément, il s'agit de :

- Préparer et nettoyer des données de pollution à grande échelle ;
- Réaliser une analyse exploratoire afin d'identifier des tendances temporelles et des périodes critiques ;
- Modéliser la pollution sous forme de graphe pour étudier sa propagation ;
- Mettre en œuvre et comparer plusieurs modèles de prédiction ;
- Évaluer les performances des modèles obtenus.

Ce projet repose sur les technologies suivantes :

- **Scala**, favorisant la programmation fonctionnelle ;
- **Apache Spark** pour le traitement distribué des données ;
- **Spark SQL** pour l'analyse exploratoire ;
- **Spark GraphX** pour la modélisation par graphe ;
- **Spark MLlib** pour l'apprentissage automatique ;
- **XChart** et **GraphViz** pour la visualisation des résultats.

I. Présentation et construction du jeu de données

1. Source des données

Le jeu de données utilisé dans ce projet provient de la plateforme **Kaggle**, une plateforme de référence pour le partage de jeux de données utilisés en data science et en apprentissage automatique.
Le dataset sélectionné est un **jeu de données multivarié de pollution atmosphérique**, couramment utilisé pour des problématiques de prévision environnementale et de séries temporelles. Il est accessible via ce lien : https://www.kaggle.com/datasets/rupakroy/lstm-datasets-multivariate-univariate .

Il contient des **mesures horaires de pollution** associées à plusieurs variables météorologiques, couvrant plusieurs années consécutives. Les données sont initialement fournies au format **CSV**, ce qui permet une ingestion directe avec Apache Spark.

Ce choix de données répond aux exigences du sujet du projet, qui recommande l'utilisation de données de pollution, de variables temporelles et d'informations contextuelles telles que la météo.
Le caractère horaire et continu des mesures rend ce dataset particulièrement pertinent pour :

- L'analyse temporelle,
- La détection de périodes critiques,
- Et la mise en œuvre de modèles de prédiction.

2. Description des variables

Chaque observation du jeu de données correspond à une heure donnée et est décrite par plusieurs variables, regroupées en trois catégories : pollution, météorologie et temps.

a. Variable cible

- **Pollution** : niveau de pollution mesuré à l'instant considéré. Cette variable constitue la **variable cible** du projet, utilisée à la fois pour l'analyse exploratoire et pour les modèles de prédiction.

b. Variables météorologiques

Les variables météorologiques fournissent un contexte environnemental essentiel à la compréhension de la pollution :

- **temp** : température de l'air ;

- **dew** : point de rosée, indicateur du taux d'humidité ;
- **press** : pression atmosphérique ;
- **wnd_spd** : vitesse du vent ;
- **wnd_dir** : direction du vent ;
- **rain** : indicateur de pluie ;
- **snow** : indicateur de neige.

Ces variables sont connues pour influencer directement ou indirectement la dispersion et l'accumulation des polluants dans l'atmosphère.

c. Variable temporelle

- **Date** : horodatage précis de la mesure, à l'échelle horaire.

Cette variable est fondamentale car elle permet :

- L'extraction de caractéristiques temporelles (heure, jour, mois, etc.) ;
- L'étude des cycles journaliers, hebdomadaires et saisonniers ;
- La modélisation temporelle de la pollution.

```
spark est bien lance
aper├ºu des donnees
+-------------------+---------+---+----+------+-------+-------+----+----+
|date               |pollution|dew|temp|press |wnd_dir|wnd_spd|snow|rain|
+-------------------+---------+---+----+------+-------+-------+----+----+
|2010-01-02 00:00:00|129.0    |-16|-4.0|1020.0|SE     |1.79   |0   |0   |
|2010-01-02 01:00:00|148.0    |-15|-4.0|1020.0|SE     |2.68   |0   |0   |
|2010-01-02 02:00:00|159.0    |-11|-5.0|1021.0|SE     |3.57   |0   |0   |
|2010-01-02 03:00:00|181.0    |-7 |-5.0|1022.0|SE     |5.36   |1   |0   |
|2010-01-02 04:00:00|138.0    |-7 |-5.0|1022.0|SE     |6.25   |2   |0   |
|2010-01-02 05:00:00|109.0    |-7 |-6.0|1022.0|SE     |7.14   |3   |0   |
|2010-01-02 06:00:00|105.0    |-7 |-6.0|1023.0|SE     |8.93   |4   |0   |
|2010-01-02 07:00:00|124.0    |-7 |-5.0|1024.0|SE     |10.72  |0   |0   |
|2010-01-02 08:00:00|120.0    |-8 |-6.0|1024.0|SE     |12.51  |0   |0   |
|2010-01-02 09:00:00|132.0    |-7 |-5.0|1025.0|SE     |14.3   |0   |0   |
+-------------------+---------+---+----+------+-------+-------+----+----+
only showing top 10 rows
```

3. Volume et structure des données

Le jeu de données contient **43 800 lignes**, correspondant à des observations horaires sur plusieurs années.

Ce volume de données est suffisamment important pour justifier l'utilisation d'un moteur de traitement distribué tel qu'Apache Spark.

Les données présentent les caractéristiques suivantes :

- Structure tabulaire claire ;
- Variables numériques et catégorielles ;
- Dépendance temporelle forte.

Ces propriétés rendent le dataset adapté à :

- L'analyse exploratoire à grande échelle,
- La programmation fonctionnelle via Spark,
- La modélisation par graphe,
- Et l'apprentissage automatique.

4. Justification du choix du dataset

Le choix de ce jeu de données a été motivé par plusieurs raisons :

- Il s'agit de **données réalistes**, proches de celles utilisées dans des systèmes de surveillance environnementale ;
- Le volume et la granularité temporelle sont adaptés à un projet Spark ;
- La présence de variables météorologiques permet une **analyse multivariée riche** ;
- Les données se prêtent naturellement à des approches d'analyse, de modélisation et de prédiction.

De plus, ce dataset permet de couvrir l'ensemble des phases demandées dans le sujet du projet : ingestion, exploration, analyse approfondie, modélisation par graphe et prédiction.

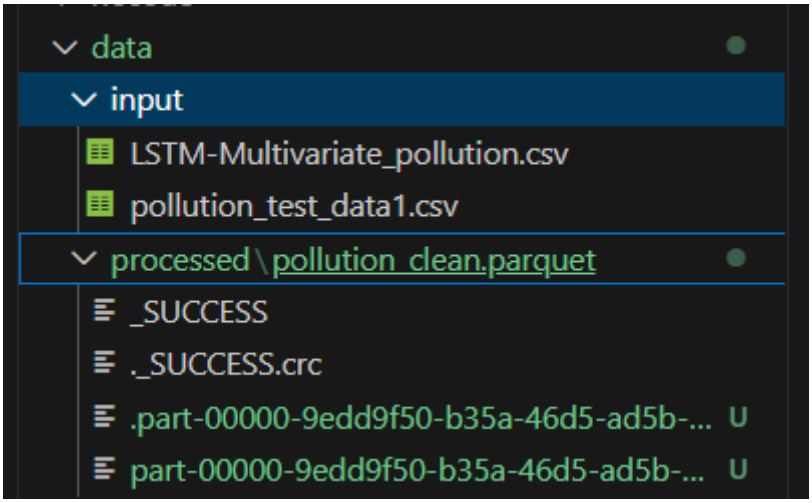
Le recours à un jeu de données issu de Kaggle garantit également une certaine **fiabilité et reproductibilité** du projet, les données étant publiques, documentées et largement utilisées dans la communauté data science.

5. Organisation des données dans le projet

Afin de structurer correctement le projet, les données ont été organisées selon une architecture claire :

- Les données brutes sont stockées dans un dossier **input**, au format CSV ;

- Les données nettoyées et enrichies sont sauvegardées dans un dossier **processed**, au format Parquet.



Cette organisation facilite la reproductibilité des expériences et la séparation entre données sources et données transformées.

II. Ingestion et préparation des données avec Apache Spark

Cette étape constitue une phase clé du projet, car la qualité des analyses et des modèles dépend directement de la qualité des données préparées. L'ensemble du processus d'ingestion et de préparation a été réalisé à l'aide d'**Apache Spark**, en exploitant les **DataFrames** et les opérations issues de la **programmation fonctionnelle**.

1. Initialisation de l'environnement Spark

Le projet débute par l'initialisation d'une **SparkSession**, point d'entrée principal pour l'utilisation de Spark SQL et des DataFrames. Cette session permet de configurer l'application Spark, de gérer les ressources et d'exécuter les transformations distribuées sur les données.

L'utilisation de Spark permet de bénéficier automatiquement :

- de la parallélisation des calculs,
- de la tolérance aux pannes,
- et d'une gestion efficace de grands volumes de données.

2. Lecture des données avec Spark

Les données brutes sont stockées au format **CSV** et sont lues à l'aide de Spark SQL. Lors de la lecture, plusieurs options sont activées afin de faciliter le traitement :

- prise en compte de la ligne d'en-tête ;
- inférence automatique du schéma ;
- conversion directe vers un **DataFrame Spark**.

Cette approche permet de travailler avec une structure tabulaire fortement typée, tout en conservant les avantages du calcul distribué.

Une première visualisation des données est réalisée à l'aide de la méthode show(), afin de vérifier la cohérence des valeurs et la structure globale du dataset.

3. Nettoyage des données

Une phase de nettoyage est ensuite appliquée afin de garantir la qualité des données :

a) Gestion des valeurs manquantes

Les valeurs manquantes sont identifiées à l'aide des fonctions Spark dédiées. Pour les variables numériques, une stratégie d'**imputation par la moyenne** est adoptée. Ce choix permet de conserver l'ensemble des observations tout en limitant l'impact des valeurs manquantes sur les analyses ultérieures.

Cette opération est réalisée de manière fonctionnelle, sans modification directe des données originales, conformément aux principes de la programmation fonctionnelle.

```
//nettoyage des données

//retirons toutes les lignes vides

val dfClean=dfRaw.na.drop()
println(s"nombre de lignes avant nettoyage : ${dfRaw.count()}")
println(s"nombre de lignes apres nettoyage : ${dfClean.count()}")
println("aperçu des donnees nettoyees")
dfClean.show(10,truncate = false)
```

b) Suppression des doublons

Les doublons sont supprimés afin d'éviter toute redondance dans les calculs statistiques et les modèles de prédiction.
Spark permet d'effectuer cette opération efficacement, même sur de grands volumes de données, grâce à ses mécanismes de partitionnement et de distribution.

Après cette étape, le nombre total de lignes est vérifié afin de s'assurer de la cohérence du nettoyage.

```
//suppressions des doublons
val dfNoDupli =dfClean.dropDuplicates()
println(s"nombre de lignes apres suppression des doublons : ${dfNoDupli.count()}")

//remplacer les valeurs manquantes
val dfImpute = dfNoDupli.na.fill(Map(
      "temp" -> 0,
      "dew" -> 0,
      "wnd_spd" -> 0,
      "pollution" -> 0
    ))
println("aperçu des donnees apres imputation")
dfImpute.show(10,truncate = false)
```

4. Enrichissement des données (Feature Engineering)

Une étape d'enrichissement des données est réalisée afin d'exploiter pleinement la dimension temporelle du dataset.

À partir de la variable **date**, plusieurs variables temporelles sont extraites :

- **year** : année ;
- **month** : mois ;

- **day** : jour du mois ;
- **hour** : heure de la journée ;
- **dayofweek** : jour de la semaine.

Ces variables sont obtenues à l'aide de fonctions Spark SQL appliquées de manière distribuée. Cet enrichissement permet :

- D'analyser les cycles journaliers et saisonniers ;
- De détecter des périodes critiques de pollution ;
- D'améliorer la performance des modèles de prédiction.

```
//convertissons la colonne date en timestamp
val dfTS = dfImpute.withColumn("timestamp",to_timestamp(col("date"), "yyyy-MM-dd HH:mm:ss"))
println("aperçu des donnees avec date en timestamp")
dfTS.show(10,truncate = false)

//extraire les variables temporelles : année, mois, jour, heure

val dfTimes= dfTS
  .withColumn("year",year(col("timestamp")))
  .withColumn("month",month(col("timestamp")))
  .withColumn("day",dayofmonth(col("timestamp")))
  .withColumn("hour",hour(col("timestamp")))
  .withColumn("dayofweek",date_format(col("timestamp"),"E"))
println("apperçu des donnees avec variables temporelles")
dfTimes.show(10,truncate = false)

val dfFinal = dfTimes.withColumn(
    "pollution_level",
    when(col("pollution") < 50, "Low")
      .when(col("pollution") < 100, "Medium")
      .otherwise("High")
  )
```

5. Vérification et aperçu des données préparées

Après le nettoyage et l'enrichissement, un nouvel aperçu des données est effectué afin de vérifier :

- la cohérence des nouvelles variables ;
- L'absence de valeurs aberrantes ;
- La bonne transformation des types de données.

Cette étape permet de valider que les données sont prêtes pour l'analyse exploratoire et la modélisation.

6. Sauvegarde des données au format Parquet

Les données nettoyées et enrichies sont sauvegardées au format **Parquet** dans le dossier *processed* du projet.

Le choix du format Parquet est motivé par plusieurs avantages :

- format colonne optimisé pour les requêtes analytiques ;
- compression efficace ;
- temps de lecture plus rapides avec Spark ;
- meilleure scalabilité pour des volumes de données importants.

Cette sauvegarde intermédiaire permet également de séparer clairement les phases de **préparation** et d'**analyse**, améliorant ainsi la reproductibilité du projet.

```
val outputPath="data/processed/pollution_clean.parquet"
//sauvegarde au format parquet
dfFinal.write.mode("overwrite").parquet(outputPath)
println(s"donnees nettoyees sauvegardees dans : $outputPath")
```

7. Apport de la programmation fonctionnelle

L'ensemble des opérations de préparation des données repose sur les principes de la programmation fonctionnelle :

- Utilisation de transformations immuables sur les DataFrames ;
- Enchaînement d'opérations telles que select, filter, groupBy et withColumn ;
- Absence de boucles impératives explicites ;
- Parallélisation automatique des traitements.

Cette approche est particulièrement adaptée au traitement de données massives et constitue un des apports majeurs d'Apache Spark dans ce projet.

III. Analyse exploratoire des données (EDA)

L'analyse exploratoire des données (Exploratory Data Analysis – EDA) constitue une étape essentielle dans tout projet de data science. Elle permet de mieux comprendre la structure des données, d'identifier des tendances, des anomalies et des relations entre variables, avant toute phase de modélisation.

Dans ce projet, l'EDA a été réalisée à l'aide de **Spark SQL** et des opérations fonctionnelles proposées par Apache Spark, ce qui permet de traiter efficacement un volume important de données tout en conservant une approche déclarative et distribuée.

1. Statistiques globales de la pollution

Une première analyse consiste à calculer des statistiques globales sur la variable de pollution. Les indicateurs suivants ont été extraits :

- valeur minimale de pollution ;
- valeur maximale de pollution ;
- valeur moyenne de pollution.

Les résultats montrent que la pollution présente une **forte variabilité**, avec des valeurs pouvant aller de 0 jusqu'à des niveaux très élevés. Cette amplitude importante suggère la présence de **pics de pollution extrêmes**, justifiant une analyse approfondie et l'utilisation de modèles capables de capturer des comportements non linéaires.

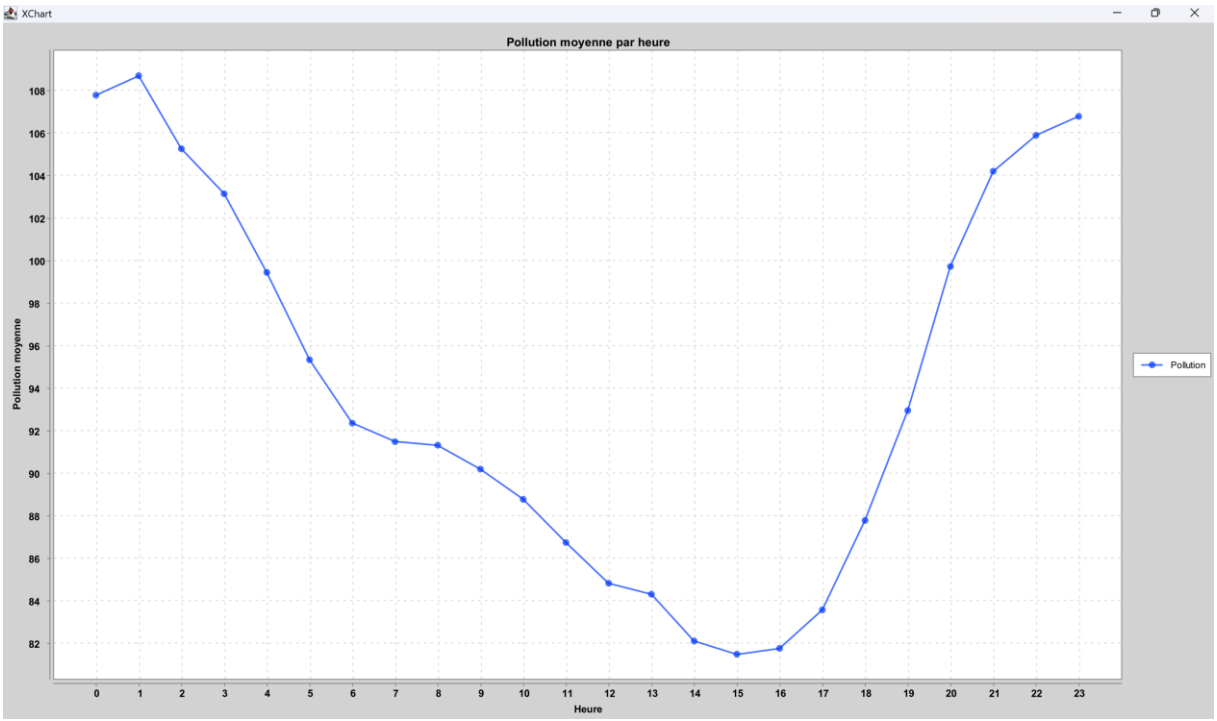
2. Analyse de la pollution moyenne par heure

L'analyse de la pollution moyenne par heure met en évidence une dynamique journalière marquée.

Les résultats montrent que :

- les niveaux de pollution sont plus élevés durant la **nuit** et en **fin de soirée** ;
- un minimum est observé au cours de l'**après-midi**, généralement entre 14h et 16h.

Cette variation horaire peut s'expliquer par une combinaison de facteurs, tels que l'activité humaine, la circulation routière, ainsi que les conditions météorologiques favorisant ou limitant la dispersion des polluants.

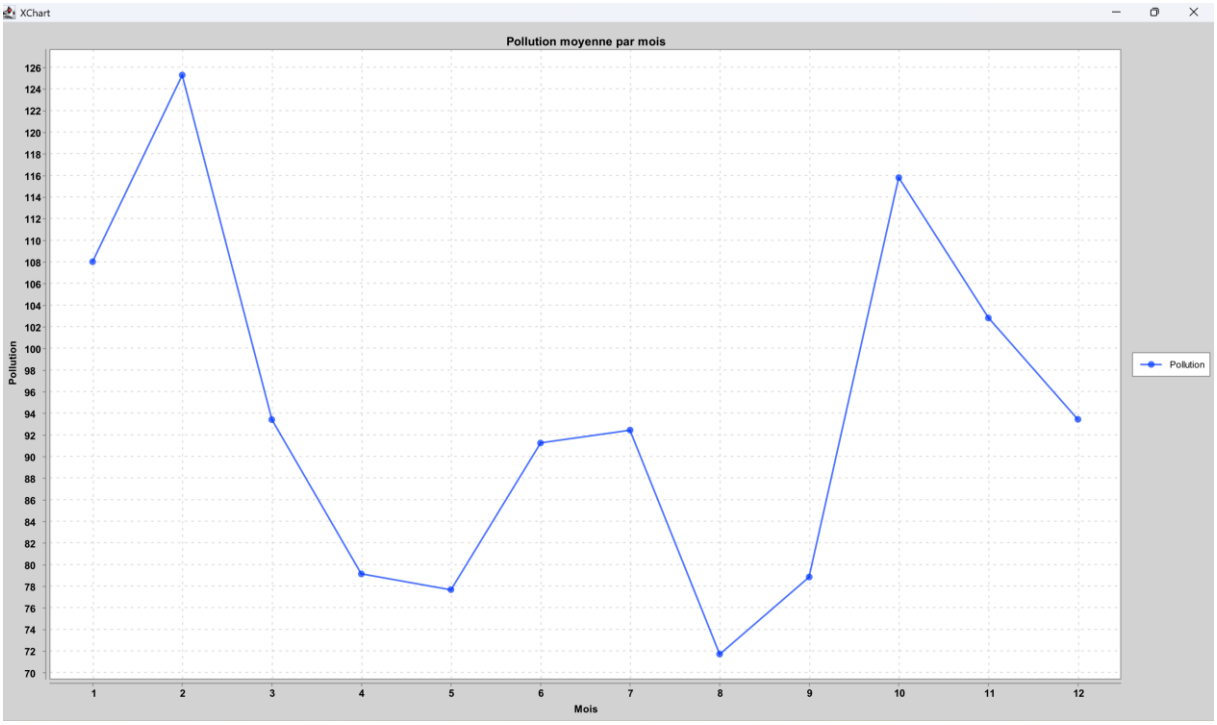


3. Analyse mensuelle et saisonnalité

L'analyse de la pollution moyenne par mois révèle une **forte saisonnalité** :

- les mois d'hiver présentent des niveaux de pollution plus élevés ;
- un minimum est observé durant les mois d'été, notamment en août ;
- une remontée progressive est constatée à l'automne.

Cette saisonnalité est cohérente avec des phénomènes connus, tels que l'augmentation du chauffage en hiver et une meilleure dispersion atmosphérique en été.



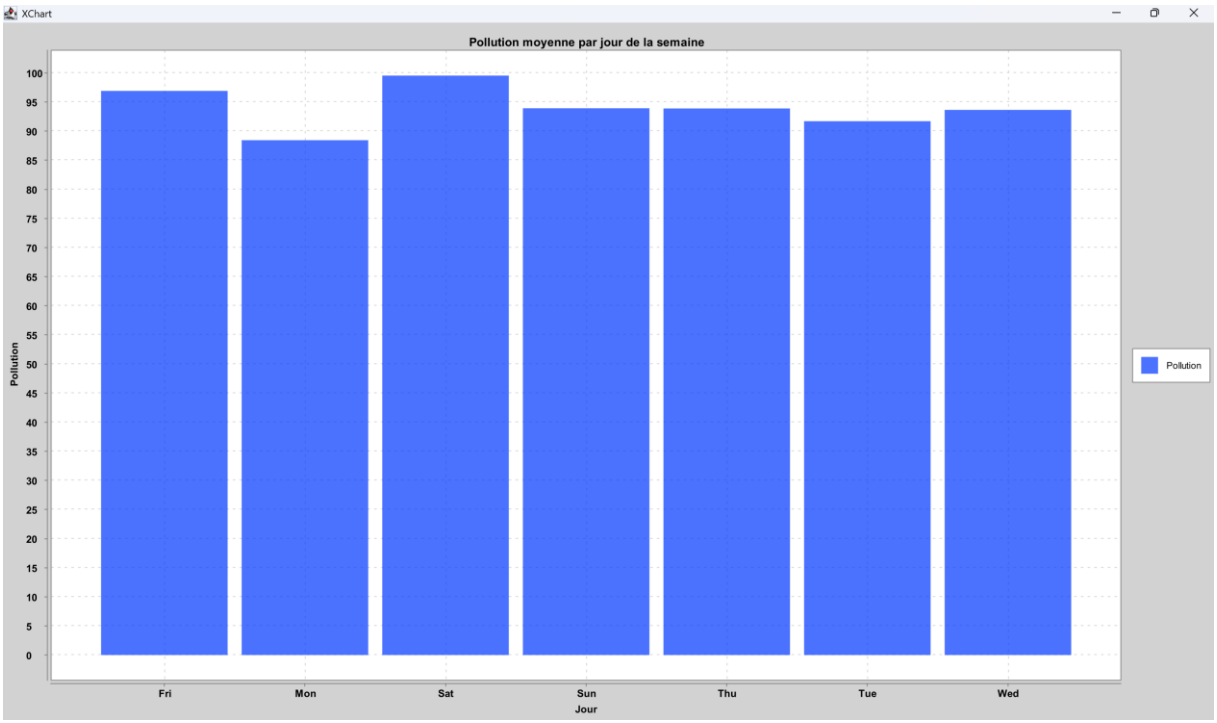
4. Analyse par jour de la semaine

Une analyse complémentaire a été réalisée en calculant la pollution moyenne par jour de la semaine.

Les résultats indiquent que :

- les différences entre les jours ouvrés sont relativement faibles ;
- les week-ends présentent en moyenne des niveaux légèrement plus élevés.

Ces variations peuvent être liées à des changements de comportements et d'activités humaines selon les jours de la semaine.



5. Corrélations entre pollution et variables météorologiques

Afin d'évaluer l'influence directe des variables météorologiques sur la pollution, des corrélations ont été calculées entre la pollution et certaines variables numériques, notamment la température.

Les résultats montrent une **corrélation négative faible** entre la pollution et la température, ce qui indique que la température seule ne permet pas d'expliquer les variations de pollution.

Cette observation justifie l'utilisation de **modèles multivariés**, capables de prendre en compte simultanément plusieurs facteurs environnementaux.

```
Correlation entre pollution et temperature : -0.09079795152516064
Statistiques journalieres de la pollution
+----------+------------+------------+------------------+
|date_only |max_pollution|min_pollution|avg_pollution     |
+----------+------------+------------+------------------+
|2010-01-02|181.0       |105.0       |145.95833333333334|
|2010-01-03|107.0       |53.0        |78.83333333333333 |
|2010-01-04|79.0        |20.0        |31.333333333333332|
|2010-01-05|106.0       |25.0        |42.458333333333336|
|2010-01-06|132.0       |20.0        |56.416666666666664|
|2010-01-07|198.0       |23.0        |69.0              |
|2010-01-08|275.0       |66.0        |176.20833333333334|
|2010-01-09|196.0       |37.0        |88.5              |
|2010-01-10|88.0        |22.0        |57.25             |
|2010-01-11|28.0        |13.0        |20.0              |
+----------+------------+------------+------------------+
only showing top 10 rows

 Top 10 journees les plus polluees (moyenne)
+----------+------------+------------+------------------+
|date_only |max_pollution|min_pollution|avg_pollution     |
+----------+------------+------------+------------------+
|2013-01-12|886.0       |0.0         |529.4583333333334 |
|2011-02-21|595.0       |407.0       |493.9166666666667 |
|2014-01-16|671.0       |161.0       |457.5             |
|2014-02-25|577.0       |369.0       |442.6666666666667 |
|2010-12-21|615.0       |206.0       |434.8333333333333 |
|2012-01-19|558.0       |348.0       |426.7916666666667 |
|2013-01-29|526.0       |0.0         |419.875           |
|2013-01-13|744.0       |0.0         |417.0416666666667 |
|2011-02-23|470.0       |373.0       |416.6666666666667 |
|2014-02-26|562.0       |27.0        |407.6666666666667 |
+----------+------------+------------+------------------+
```

6. Analyse des statistiques journalières et détection des pics

Une analyse journalière a été menée afin d'identifier les journées les plus polluées. Pour chaque jour, les statistiques suivantes ont été calculées :

- pollution maximale ;
- pollution minimale ;
- pollution moyenne.

Cette analyse permet de détecter des **journées critiques**, caractérisées par des niveaux de pollution anormalement élevés. Ces événements extrêmes sont particulièrement importants à identifier, car ils peuvent avoir un impact significatif sur la santé publique.

```
Statistiques globales de la pollution :
+-------------+-------------+-----------------+
|max_pollution|min_pollution|avg_pollution    |
+-------------+-------------+-----------------+
|994.0        |0.0          |94.01351598173515|
+-------------+-------------+-----------------+
```

7. Répartition des niveaux de pollution

Afin de simplifier l'analyse, les niveaux de pollution ont été regroupés en trois catégories :

- **Low** (faible),
- **Medium** (moyen),
- **High** (élevé).

La répartition de ces catégories montre que :

- une part importante des observations correspond à un niveau faible ;
- une proportion significative correspond à un niveau élevé ;
- le reste se situe à un niveau intermédiaire.

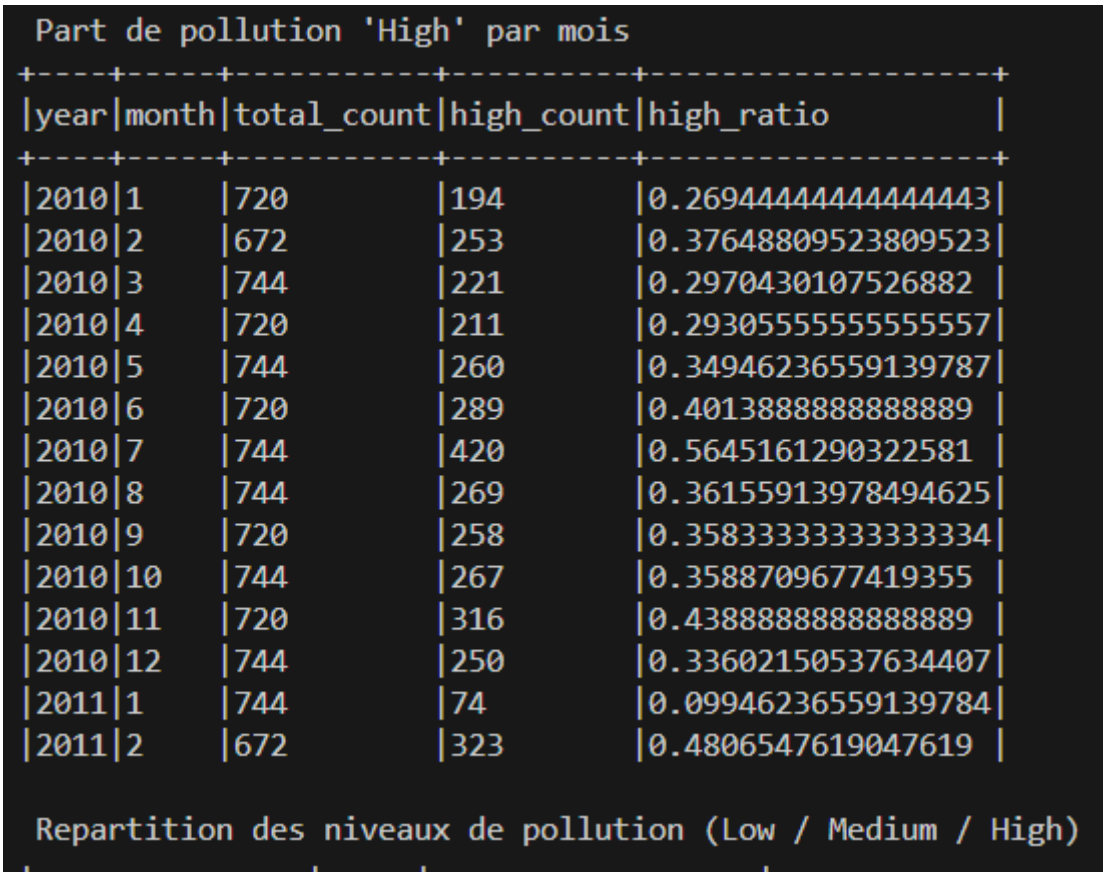
Cette classification met en évidence la fréquence non négligeable d'épisodes de forte pollution et souligne l'intérêt d'un système de prédiction fiable.

```
 Repartition des niveaux de pollution (Low / Medium / High)
+---------------+-----+------------------+
|pollution_level|count|percentage        |
+---------------+-----+------------------+
|High           |15562|35.529680365296805|
|Low            |17727|40.47260273972603 |
|Medium         |10511|23.99771689497717 |
+---------------+-----+------------------+
```

8. Analyse de la pollution élevée par mois

Une analyse plus fine a été réalisée en étudiant la proportion de pollution classée comme **High** pour chaque mois.

Les résultats montrent que certains mois présentent une **probabilité plus élevée d'épisodes fortement pollués**, confirmant les observations issues de l'analyse saisonnière. Cette information est particulièrement utile pour la prise de décision et la mise en place de mesures préventives.



Part de pollution 'High' par mois

year	month	total_count	high_count	high_ratio
2010	1	720	194	0.26944444444444443
2010	2	672	253	0.37648809523809523
2010	3	744	221	0.2970430107526882
2010	4	720	211	0.29305555555555557
2010	5	744	260	0.34946236559139787
2010	6	720	289	0.4013888888888889
2010	7	744	420	0.5645161290322581
2010	8	744	269	0.36155913978494625
2010	9	720	258	0.35833333333333334
2010	10	744	267	0.3588709677419355
2010	11	720	316	0.4388888888888889
2010	12	744	250	0.33602150537634407
2011	1	744	74	0.09946236559139784
2011	2	672	323	0.4806547619047619

Repartition des niveaux de pollution (Low / Medium / High)

9. Apports de l'analyse exploratoire

L'analyse exploratoire a permis de :

- comprendre la structure et la variabilité des données ;
- identifier des cycles journaliers et saisonniers ;
- détecter des périodes et des journées critiques ;
- guider les choix de modélisation par graphe et de prédiction.

Les conclusions issues de cette phase constituent une base solide pour les étapes suivantes du projet, notamment la modélisation des relations entre stations et la prédiction de la pollution.

IV. Modélisation par graphe avec Spark GraphX

L'analyse exploratoire permet de comprendre les tendances globales de la pollution, mais elle reste limitée à une lecture statistique et temporelle. Afin d'adopter une vision plus structurelle et dynamique du phénomène, une modélisation par graphe a été mise en œuvre à l'aide de **Spark GraphX**.

Cette approche permet de représenter la pollution non plus uniquement comme une suite de valeurs dans le temps, mais comme un **réseau de relations** entre différentes périodes, facilitant ainsi l'étude de la **propagation** et de la diffusion de la pollution.

1. Motivation de l'approche par graphe

La pollution atmosphérique est un phénomène qui évolue de manière continue dans le temps. Les niveaux observés à un instant donné sont souvent influencés par les périodes précédentes et suivantes.

La modélisation par graphe permet :

- de représenter explicitement ces relations ;
- d'étudier la continuité et la dépendance temporelle ;
- de simuler la diffusion de la pollution entre différentes périodes.

Spark GraphX s'impose comme un outil adapté à cette problématique, car il permet de manipuler des graphes distribués tout en restant intégré à l'écosystème Spark.

2. Définition des stations temporelles

Le jeu de données utilisé ne fournit pas de stations géographiques explicites. Afin de répondre aux exigences du sujet et de construire un graphe pertinent, des **stations temporelles** ont été définies.

La journée a été découpée en cinq périodes distinctes :

- **Nuit** : de 0h à 4h
- **Matin** : de 5h à 8h
- **Milieu de journée** : de 9h à 12h
- **Après-midi** : de 13h à 18h
- **Soir** : de 19h à 23h

Chaque station représente un sommet du graphe et est associée à la **pollution moyenne** observée sur la plage horaire correspondante.

Ce choix permet de conserver une cohérence temporelle tout en simplifiant la représentation du phénomène.

3. Construction du graphe

Le graphe est construit à partir de deux éléments principaux :

i. Sommets (vertices)

Chaque sommet du graphe correspond à une station temporelle.
À chaque sommet est associée une valeur représentant la pollution moyenne calculée lors de l'analyse exploratoire.

Cette information constitue l'attribut principal des sommets et sert de base à la simulation de propagation.

ii. Arêtes (edges)

Les arêtes du graphe représentent les **relations temporelles** entre les stations.
Des connexions sont établies entre les périodes successives de la journée afin de modéliser la continuité naturelle du temps.

Des arêtes secondaires, avec des poids plus faibles, peuvent également être ajoutées pour représenter des influences indirectes entre périodes non consécutives.

Les poids des arêtes traduisent l'intensité de l'influence entre deux stations.

4. Graphe initial

Le graphe initial correspond à l'état de la pollution avant toute propagation.
Chaque station possède une valeur propre, issue directement des moyennes calculées à partir des données réelles.

Ce graphe constitue une photographie statique de la pollution moyenne par période de la journée.
Il est exporté au format **GraphViz (.dot)**, puis converti en image afin de faciliter l'interprétation visuelle.

5. Simulation de la propagation de la pollution

Une étape de propagation est ensuite appliquée sur le graphe.
L'objectif est de simuler la diffusion de la pollution entre stations connectées, en tenant compte des poids des arêtes.

Lors de cette propagation :

- chaque station est influencée par les stations voisines ;
- les valeurs de pollution tendent à se rapprocher ;
- un effet de lissage est observé.

Cette simulation permet de modéliser l'idée selon laquelle la pollution observée à une période donnée est partiellement héritée des périodes précédentes et suivantes.

6. Graphe après propagation

Après la propagation, un nouveau graphe est généré.
Les valeurs de pollution associées aux stations sont modifiées en fonction des influences reçues.

La comparaison entre le graphe initial et le graphe propagé met en évidence :

- une réduction des écarts entre stations ;
- une redistribution plus homogène des niveaux de pollution ;
- une meilleure représentation de la dynamique temporelle.

Ce graphe est également exporté et visualisé à l'aide de GraphViz.

7. Apports de la modélisation par graphe

La modélisation par graphe apporte plusieurs bénéfices au projet :

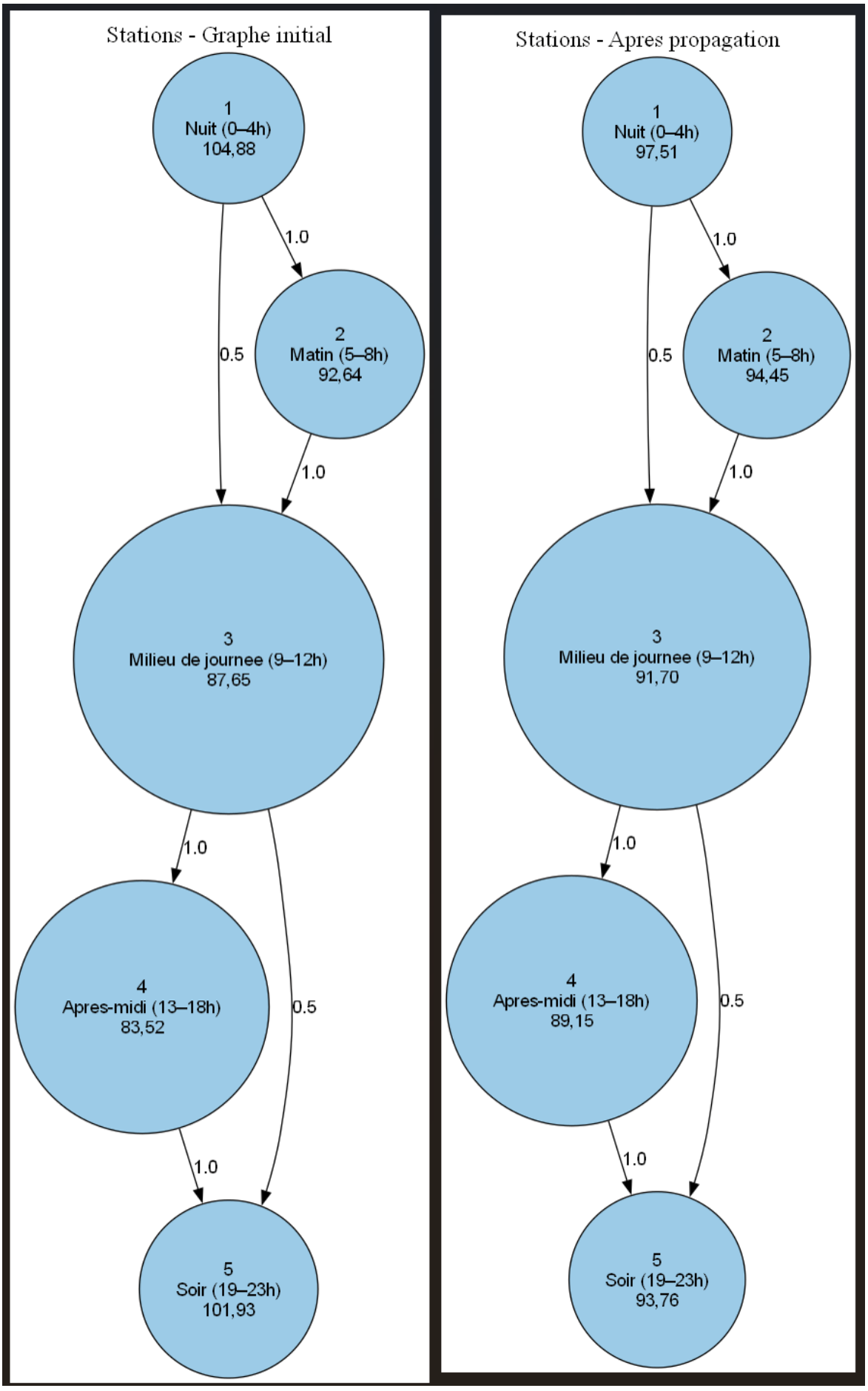
- elle offre une **vision réseau** de la pollution ;
- elle complète l'analyse purement statistique ;
- elle illustre la capacité de Spark à traiter des structures complexes ;
- elle répond directement aux objectifs du sujet concernant l'étude de la propagation.

Cette approche constitue une étape intermédiaire pertinente entre l'analyse exploratoire et la prédiction par machine learning.

```
Lancement de l'analyse graphe (GraphX)
Graphe initial des stations
Stations :
   4 : Apres-midi (13ÔÇô18h)           pollution moyenne =  83,52
   1 : Nuit (0ÔÇô4h)                   pollution moyenne = 104,88
   3 : Milieu de journee (9ÔÇô12h)     pollution moyenne =  87,65
   5 : Soir (19ÔÇô23h)                 pollution moyenne = 101,93
   2 : Matin (5ÔÇô8h)                  pollution moyenne =  92,64
Connexions :
  1 -> 2 (poids = 1.0)
  2 -> 3 (poids = 1.0)
  3 -> 4 (poids = 1.0)
  4 -> 5 (poids = 1.0)
  1 -> 3 (poids = 0.5)
  3 -> 5 (poids = 0.5)

Fichier DOT exporte dans : output/stations_initial.dot
[OK] GraphViz execute avec : dot
[OK] PNG genere : C:\Users\wendp\spark-pollution\output\stations_initial.png
[OK] Image ouverte automatiquement.
Graphe apres propagation
Stations :
   4 : Apres-midi (13ÔÇô18h)           pollution moyenne =  89,15
   1 : Nuit (0ÔÇô4h)                   pollution moyenne =  97,51
   3 : Milieu de journee (9ÔÇô12h)     pollution moyenne =  91,70
   5 : Soir (19ÔÇô23h)                 pollution moyenne =  93,76
   2 : Matin (5ÔÇô8h)                  pollution moyenne =  94,45
Connexions :
  1 -> 2 (poids = 1.0)
  2 -> 3 (poids = 1.0)
  3 -> 4 (poids = 1.0)
  4 -> 5 (poids = 1.0)
  1 -> 3 (poids = 0.5)
  3 -> 5 (poids = 0.5)

Fichier DOT exporte dans : output/stations_propagated.dot
[OK] GraphViz execute avec : dot
[OK] PNG genere : C:\Users\wendp\spark-pollution\output\stations_propagated.png
[OK] Image ouverte automatiquement.
```


Stations - Graphe initial
1
Nuit (0–4h)
104,88
2
Matin (5–8h)
92,64
3
Milieu de journee (9–12h)
87,65
4
Apres-midi (13–18h)
83,52
5
Soir (19–23h)
101,93
1.0
0.5
1.0
1.0
0.5
1.0
Stations - Apres propagation
1
Nuit (0–4h)
97,51
2
Matin (5–8h)
94,45
3
Milieu de journee (9–12h)
91,70
4
Apres-midi (13–18h)
89,15
5
Soir (19–23h)
93,76
1.0
0.5
1.0
1.0
0.5
1.0

V. Modélisation prédictive avec Spark MLlib

Après l'analyse exploratoire et la modélisation par graphe, l'objectif principal du projet est de **prédire le niveau de pollution** à partir des variables météorologiques et temporelles disponibles.
Cette phase repose sur l'utilisation de **Spark MLlib**, la bibliothèque de machine learning intégrée à Apache Spark.

L'approche adoptée consiste à construire plusieurs modèles de régression, à les entraîner sur les mêmes données, puis à comparer leurs performances afin d'identifier le modèle le plus adapté.

1. Objectif de la prédiction

L'objectif de la modélisation prédictive est d'estimer la valeur de la pollution à un instant donné en fonction :

- des conditions météorologiques ;
- des variables temporelles (heure, mois) ;
- des relations mises en évidence lors de l'analyse exploratoire.

La pollution étant une variable numérique continue, le problème est formulé comme un **problème de régression**.

2. Préparation des données pour le machine learning

Avant l'entraînement des modèles, les données doivent être préparées selon les standards de Spark MLlib.

➢ Sélection des variables explicatives

Les variables retenues pour la prédiction sont :

- température (*temp*) ;
- point de rosée (*dew*) ;
- pression atmosphérique (*press*) ;
- vitesse du vent (*wnd_spd*) ;
- indicateurs de pluie (*rain*) et de neige (*snow*) ;
- heure de la journée (*hour*) ;
- mois (*month*).

Ces variables sont sélectionnées car elles ont montré une influence directe ou indirecte sur la pollution lors de l'analyse exploratoire.

- **Assemblage des features**

Les variables explicatives sont regroupées dans un vecteur de caractéristiques à l'aide de l'outil **VectorAssembler**.
Cette étape est indispensable pour rendre les données compatibles avec les modèles de Spark MLlib.

L'approche retenue respecte les principes de la programmation fonctionnelle, en appliquant des transformations successives sur les DataFrames sans modification mutable des données.

- **Séparation des jeux d'entraînement et de test**

Le dataset est ensuite séparé en deux sous-ensembles :

- un **jeu d'entraînement**, utilisé pour apprendre les modèles ;
- un **jeu de test**, utilisé pour évaluer leurs performances.

Cette séparation permet de mesurer la capacité de généralisation des modèles et d'éviter le surapprentissage.

3. Modèles de régression étudiés

Trois modèles de complexité croissante ont été implémentés et comparés.

- **Régression linéaire**

La régression linéaire constitue un modèle de référence. Elle suppose une relation linéaire entre la pollution et les variables explicatives.

Bien que simple à interpréter, ce modèle présente des limites importantes lorsqu'il s'agit de capturer des relations non linéaires ou des interactions complexes entre variables.

- **Random Forest Regressor**

Le modèle **Random Forest** repose sur un ensemble d'arbres de décision entraînés sur des sous-échantillons des données.
Il permet de modéliser des relations non linéaires et est généralement plus robuste que la régression linéaire.

Ce modèle est particulièrement adapté aux données environnementales, où les interactions entre variables peuvent être complexes.

- **Gradient Boosted Trees Regressor**

Le modèle **Gradient Boosted Trees (GBT)** utilise une approche de boosting, où les arbres sont entraînés de manière séquentielle afin de corriger les erreurs des modèles précédents.

Ce modèle offre un bon compromis entre biais et variance et est reconnu pour ses performances élevées sur des problèmes de régression complexes.

4. Entraînement des modèles

Chaque modèle est entraîné sur le même jeu de données afin de garantir une comparaison équitable.

Les paramètres principaux (nombre d'arbres, profondeur maximale, nombre d'itérations) sont choisis de manière à obtenir un bon équilibre entre performance et coût de calcul.

L'entraînement est réalisé de manière distribuée grâce à Spark, ce qui permet de traiter efficacement le volume de données disponible.

5. Apports de Spark MLlib

L'utilisation de Spark MLlib présente plusieurs avantages :

- intégration directe avec les DataFrames Spark ;
- scalabilité et parallélisation automatique ;
- API cohérente pour l'entraînement et l'évaluation des modèles ;
- facilité de comparaison entre plusieurs algorithmes.

Cette bibliothèque permet ainsi de construire une chaîne complète de machine learning dans un environnement Big Data.

6. Lien avec les étapes précédentes

Les modèles de prédiction s'appuient directement sur :

- les tendances identifiées lors de l'analyse exploratoire ;
- les variables temporelles enrichies lors du prétraitement ;
- la compréhension de la dynamique de la pollution apportée par la modélisation par graphe.

Ainsi, la phase de modélisation prédictive s’inscrit dans une démarche cohérente et progressive.

7. Résultats

```
 Etape ML 1 : Regression lineaire seule
[Etape 1] Modele de regression lineaire
Entranement du modele de regression linieaire
26/01/04 22:39:13 WARN Instrumentation: [dde65b2f] regParam is zero, which might cause numerical instability and overfitting.
26/01/04 22:39:14 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
26/01/04 22:39:14 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.lapack.JNILAPACK
=== Modele lineaire entraine ===
Intercept : 1777.1992873576073
Coefficients (dans l'ordre des features) :
  (temp, dew, press, wnd_spd, snow, rain, hour, month)
  [-6.356682241282506,4.496690423047641,-1.5883301096112616,-0.241280783478105,-2.231031594560825,-6.970800200527483,1.6083614738200762,-1.4554954385841472]

=== Evaluation modele Regression lineaire ===
RMSE = 80,764
R▒   = 0,233
=== Exemple de predictions (lineaire) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|136.1090337007281 |
|124.0|129.3204588570195 |
|405.0|130.89061095019701|
|178.0|100.39375342280641|
|9.0  |10.639768129911317|
|10.0 |42.06586665330701 |
|97.0 |91.46735118736592 |
|175.0|105.52116391162758|
|240.0|114.29006136448061|
|55.0 |101.64084782602208|
+-----+------------------+
only showing top 10 rows


Etape ML 2 : Random Forest seul
=== [Etape 2] Modele Random Forest ===
=== Entrainement du Random Forest Regressor ===
26/01/04 22:39:28 WARN DAGScheduler: Broadcasting large task binary with size 1224.6 KiB
26/01/04 22:39:30 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
Modele Random Forest entraine
Nombre d'arbres : 60
Profondeur max  : 8
```

```
=== Evaluation modele Random Forest ===
RMSE = 69,252
R²   = 0,436
=== Exemple de predictions (Random Forest) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|179.09402480418535|
|124.0|179.66705224062952|
|405.0|152.91891895263885|
|178.0|114.19514898840858|
|9.0  |18.83395340647753 |
|10.0 |25.54819764756526 |
|97.0 |110.35021898506425|
|175.0|125.51836394210409|
|240.0|216.97495575527367|
|55.0 |85.8591408244843  |
+-----+------------------+
only showing top 10 rows


Etape ML 3 : GBT seul
=== [Etape 3] Modele Gradient Boosted Trees ===
 Entrainement du GBT Regressor (Gradient Boosted Trees)
Modele GBT entraine
Nombre d'iterations (arbres) : 60
Profondeur max               : 5

=== Evaluation modele GBT ===
RMSE = 64,570
R²   = 0,510
=== Exemple de predictions (GBT) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|237.08861111301735|
|124.0|221.4099155310929 |
|405.0|163.66643068930912|
|178.0|131.89933971499758|
|9.0  |47.66029102578206 |
|10.0 |18.671234000399654|
|97.0 |130.06898426883902|
|175.0|64.98024841517895 |
|240.0|249.49361126358443|
|55.0 |72.52911361759323 |
+-----+------------------+
only showing top 10 rows


Etape ML 4 : Comparaison globale sur meme split
=== Preparation des donnees pour les modeles ML ===
Nombre d'exemples train : 35036
Nombre d'exemples test  : 8764
```

```
Entranement du modele de regression linieaire
26/01/04 22:40:21 WARN Instrumentation: [275ec898] regParam is zero, which might cause numerical instability and overfitting.
=== Modele lineaire entraine ===
Intercept : 1777.1992873576073
Coefficients (dans l'ordre des features) :
  (temp, dew, press, wnd_spd, snow, rain, hour, month)
  [-6.356682241282506,4.496690423047641,-1.5883301096112616,-0.241280783478105,-2.231031594560825,-6.970800200527483,1.6083614738200762,-1.4554954385841472]

=== Evaluation modele Regression lineaire ===
RMSE = 80,764
R▒▒   = 0,233
=== Exemple de predictions (lineaire) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|136.1090337007281 |
|124.0|129.3204588570195 |
|405.0|130.89061095019701|
|178.0|100.39375342280641|
|9.0  |10.639768129911317|
|10.0 |42.06586665330701 |
|97.0 |91.46735118736592 |
|175.0|105.52116391162758|
|240.0|114.29006136448061|
|55.0 |101.64084782602208|
+-----+------------------+
only showing top 10 rows


=== Entrainement du Random Forest Regressor ===
26/01/04 22:40:33 WARN DAGScheduler: Broadcasting large task binary with size 1224.6 KiB
26/01/04 22:40:35 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
Modele Random Forest entraine
Nombre d'arbres : 60
Profondeur max  : 8

=== Evaluation modele Random Forest ===
RMSE = 69,252
R▒▒   = 0,436
=== Exemple de predictions (Random Forest) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|179.09402480418535|
|124.0|179.66705224062952|
|405.0|152.91891895263885|
|178.0|114.19514898840858|
|9.0  |18.83395340647753 |
|10.0 |25.54819764756526 |
|97.0 |110.35021898506425|
|175.0|125.51836394210409|
|240.0|216.97495575527367|
|55.0 |85.859140824484   |
+-----+------------------+
only showing top 10 rows
```

```
 Entrainement du GBT Regressor (Gradient Boosted Trees)
Modele GBT entraine
Nombre d'iterations (arbres) : 60
Profondeur max               : 5

=== Evaluation modele GBT ===
RMSE = 64,570
R²   = 0,510
=== Exemple de predictions (GBT) ===
+-----+------------------+
|label|prediction        |
+-----+------------------+
|393.0|237.08861111301735|
|124.0|221.4099155310929 |
|405.0|163.66643068930912|
|178.0|131.89933971499758|
|9.0  |47.66029102578206 |
|10.0 |18.671234000399654|
|97.0 |130.06898426883902|
|175.0|64.98024841517895 |
|240.0|249.49361126358443|
|55.0 |72.52911361759323 |
+-----+------------------+
only showing top 10 rows
```

VI. Évaluation et comparaison des modèles

Après l'entraînement des différents modèles de prédiction, une phase d'évaluation est indispensable afin de mesurer leurs performances et de comparer leur capacité à prédire correctement le niveau de pollution.
Cette étape permet également d'identifier les limites de chaque approche et de justifier le choix du modèle final.

1. Métriques d'évaluation

Deux métriques principales ont été utilisées pour évaluer les performances des modèles de régression :

- ✓ **RMSE (Root Mean Squared Error)**

La RMSE mesure l'erreur moyenne entre les valeurs réelles et les valeurs prédites par le modèle.

Elle pénalise fortement les erreurs importantes, ce qui est particulièrement pertinent dans le contexte de la pollution, où les pics extrêmes sont critiques.

Une valeur de RMSE faible indique une meilleure précision du modèle.

- ✓ **Coefficient de détermination (R^2)**

Le coefficient R^2 mesure la proportion de la variance de la variable cible expliquée par le modèle.
Une valeur proche de 1 indique un bon ajustement, tandis qu'une valeur proche de 0 indique une faible capacité explicative.

Ces deux métriques combinées permettent d'avoir une vision globale de la qualité des modèles.

2. Résultats obtenus

Les trois modèles ont été entraînés et évalués sur le même découpage des données (jeu d'entraînement et jeu de test), afin de garantir une comparaison équitable.

Les résultats obtenus sont résumés dans le tableau suivant :

```
 Comparaison des modeles (tries par RMSE croissant) :
Modele                        |   RMSE   |    R┐
------------------------------+----------+--------
Gradient Boosted Trees        |   64,570 |  0,510
Random Forest                 |   69,252 |  0,436
Regression lineaire           |   80,764 |  0,233
```

3. Analyse comparative des modèles

- ✓ **Régression linéaire**

La régression linéaire présente les performances les plus faibles parmi les trois modèles. La valeur élevée de la RMSE et le faible coefficient R^2 indiquent que ce modèle n'est pas capable de capturer efficacement la complexité des relations entre la pollution et les variables explicatives.

Ce résultat confirme les limites d'un modèle linéaire face à un phénomène environnemental non linéaire et multivarié.

- ✓ **Random Forest Regressor**

Le modèle Random Forest améliore significativement les performances par rapport à la régression linéaire.
La baisse de la RMSE et l'augmentation du coefficient R^2 montrent que ce modèle est capable de mieux représenter les interactions entre variables.

Cependant, malgré cette amélioration, certaines erreurs importantes persistent, notamment lors des épisodes de pollution extrême.

- ✓ **Gradient Boosted Trees Regressor**

Le modèle Gradient Boosted Trees obtient les **meilleures performances globales**. Il présente la RMSE la plus faible et le coefficient R^2 le plus élevé, ce qui indique une meilleure capacité de généralisation.

Ce modèle est particulièrement efficace pour capturer les relations non linéaires et s'adapter aux variations complexes des données environnementales.

4. Analyse des prédictions et des erreurs

L'analyse des prédictions montre que :

- les modèles sont globalement précis pour les niveaux de pollution faibles à moyens ;
- les erreurs augmentent lors des pics de pollution extrêmes.

Ce comportement est cohérent avec la nature du phénomène étudié, les événements extrêmes étant plus rares et plus difficiles à prédire.

Malgré ces limites, le modèle Gradient Boosted Trees reste le plus robuste parmi ceux étudiés.

5. Choix du modèle final

Au vu des résultats obtenus, le modèle **Gradient Boosted Trees** est retenu comme modèle final pour la prédiction de la pollution.

Ce choix est justifié par :

- ses meilleures performances quantitatives ;
- sa capacité à modéliser des relations complexes ;
- sa robustesse face aux variations des données.

6. Apports de l'évaluation

La phase d'évaluation a permis de :

- comparer objectivement plusieurs approches de modélisation ;
- mettre en évidence les limites des modèles simples ;
- valider l'intérêt des modèles ensemblistes pour la prédiction de la pollution.

Cette étape confirme la pertinence de la démarche adoptée tout au long du projet.

VII. Importance des variables

Au-delà des performances globales des modèles, il est essentiel de comprendre **quelles variables influencent réellement la prédiction de la pollution**. L'analyse de l'importance des variables permet d'interpréter les modèles et de relier les résultats obtenus aux connaissances du domaine environnemental.

Dans ce projet, cette analyse a été réalisée principalement à partir des modèles **Random Forest** et **Gradient Boosted Trees**, qui fournissent des mesures d'importance des caractéristiques.

1. Principe de l'importance des variables

Les modèles basés sur les arbres de décision évaluent l'importance des variables en mesurant leur contribution à la réduction de l'erreur lors des divisions successives des données.

Une variable est considérée comme importante si :

- elle est fréquemment utilisée dans les arbres ;
- elle permet de réduire significativement l'erreur de prédiction ;
- elle améliore la séparation des données.

Cette approche permet d'obtenir une interprétation qualitative du modèle, même lorsque celui-ci est non linéaire.

2. Résultats obtenus

L'analyse des importances met en évidence plusieurs variables dominantes dans la prédiction de la pollution.

Les variables les plus influentes sont :

- le **mois**, qui capture la saisonnalité du phénomène ;
- le **point de rosée (dew)**, indicateur de l'humidité de l'air ;

- la **température** ;
- la **pression atmosphérique** ;
- la **vitesse du vent**.

Les variables **pluie** et **neige** présentent une importance plus faible, ce qui peut s'expliquer par leur occurrence relativement rare dans le dataset.

3. Interprétation des résultats

L'importance élevée de la variable *mois* confirme les conclusions de l'analyse exploratoire, qui avait mis en évidence une forte saisonnalité de la pollution.

Le point de rosée et la température jouent également un rôle clé, car ils influencent la stabilité de l'atmosphère et la capacité de dispersion des polluants.

La pression atmosphérique et la vitesse du vent interviennent dans la dynamique de transport et d'accumulation des polluants, ce qui explique leur contribution non négligeable à la prédiction.

4. Cohérence avec l'analyse exploratoire

Les résultats de l'analyse des importances sont cohérents avec les tendances observées lors de l'analyse exploratoire :

- les périodes hivernales, associées à certains mois, sont plus polluées ;
- les conditions météorologiques défavorables à la dispersion augmentent les niveaux de pollution ;
- les facteurs temporels et météorologiques agissent de manière conjointe.

Cette cohérence renforce la crédibilité des modèles et valide la démarche adoptée.

5. Apports de l'analyse de l'importance des variables

L'analyse de l'importance des variables apporte plusieurs bénéfices :

- elle permet d'interpréter les modèles de prédiction ;
- elle relie les résultats numériques à des phénomènes réels ;
- elle renforce la compréhension globale du phénomène étudié ;
- elle facilite l'identification de leviers d'action potentiels.

Cette étape constitue donc un complément essentiel à l'évaluation quantitative des performances.

VIII. Limites et perspectives

Malgré les résultats encourageants obtenus, ce projet présente certaines limites qui doivent être prises en compte. L'identification de ces limites permet de mieux interpréter les résultats et d'envisager des pistes d'amélioration pertinentes.

1. Limites du projet

La première limite concerne la **nature des données** utilisées. Le jeu de données, bien que riche et multivarié, ne contient pas d'informations géographiques précises sur les stations de mesure. Cette absence limite l'analyse spatiale de la pollution et empêche l'étude des interactions entre différentes zones géographiques.

Une autre limite réside dans la **modélisation temporelle**. Les modèles utilisés ne prennent pas explicitement en compte les dépendances temporelles à long terme, telles que les effets retardés (lags). Cela peut réduire la précision des prédictions, en particulier lors des épisodes de pollution extrême.

Par ailleurs, certaines variables météorologiques, comme la **direction du vent**, n'ont pas été encodées de manière catégorielle ou vectorielle. Cette simplification peut entraîner une perte d'information utile pour la prédiction.

Enfin, bien que Spark permette le traitement de données massives, le projet reste basé sur des **données historiques statiques**. Il ne prend pas en charge le traitement en temps réel, ce qui limite son application dans des contextes de surveillance instantanée de la pollution.

2. Perspectives d'amélioration

Plusieurs pistes d'amélioration peuvent être envisagées pour enrichir et étendre ce projet.

Tout d'abord, l'intégration de **Spark Structured Streaming** permettrait de traiter des flux de données en temps réel, ouvrant la voie à un système de surveillance continue de la pollution.

Ensuite, l'ajout de **variables retardées** (par exemple la pollution aux heures précédentes) permettrait de mieux capturer la dynamique temporelle du phénomène. Des modèles de séries temporelles plus avancés, tels que les **réseaux de neurones récurrents (LSTM)**, pourraient également être envisagés.

Une autre amélioration consisterait à enrichir les données avec des **informations géographiques**, afin de réaliser une analyse spatiale plus fine et de modéliser la propagation de la pollution entre différentes zones.

Enfin, une phase de **tuning des hyperparamètres** plus approfondie pourrait améliorer les performances des modèles, en particulier pour les Random Forest et les Gradient Boosted Trees.

Conclusion

Ce projet avait pour objectif d'analyser et de prédire la pollution urbaine à partir de données environnementales massives en exploitant les capacités d'Apache Spark et du langage Scala.

L'ensemble des étapes clés d'un projet de data science a été couvert, depuis l'ingestion et la préparation des données jusqu'à l'évaluation de modèles de machine learning. L'analyse exploratoire a permis de mettre en évidence des tendances temporelles marquées et une forte saisonnalité de la pollution. La modélisation par graphe a offert une vision complémentaire du phénomène, en illustrant la propagation de la pollution entre différentes périodes de la journée.

La phase de modélisation prédictive a montré que les modèles non linéaires, et en particulier les **Gradient Boosted Trees**, sont les plus performants pour prédire la pollution à partir des variables météorologiques et temporelles disponibles. L'analyse de l'importance des variables a permis d'interpréter ces résultats et de confirmer le rôle central de la saisonnalité et des conditions atmosphériques.

Ce projet met en évidence l'intérêt des technologies Big Data pour l'analyse environnementale et démontre la capacité d'Apache Spark à gérer des problématiques complexes combinant analyse de données, graphes et apprentissage automatique. Il constitue une base solide pour des développements futurs orientés vers le temps réel, l'analyse spatiale et des modèles prédictifs plus avancés.