Jour 7 : Introduction à *Jupyter* et à la visualisation de données avec *seaborn* et *matplotlib*

Sommaire

Introduction :

Cette journée se concentre sur la découverte de la visualisation de données en utilisant seaborn (majoritairement) et Matplotlib. Seaborn est une librairie Python puissante qui s'appuie sur Matplotlib et offre une interface de haut niveau pour la création de graphiques statistiques. Elle permet de produire des visualisations complexes avec un code relativement simple.

La visualisation des données est une compétence cruciale en Big Data car elle permet de comprendre rapidement les tendances, de repérer les anomalies, et d'identifier les patterns dans les données qui pourraient ne pas être évidents à partir des données brutes seules. Des visuels efficaces peuvent communiquer clairement et de manière persuasive les résultats aux publics non techniques, ce qui les rend indispensables dans les environnements commerciaux par exemple. En transformant les résultats complexes de diverses analyses en visuels compréhensibles, il est possible de prendre des décisions basées sur les données plus efficacement et transmettre des idées de manière convaincante.

Aujourd'hui, vous allez apprendre à créer un ensemble de représentations visuelles des données, des graphiques de base comme les histogrammes et les nuages de points jusqu'à des visualisations plus complexes telles que les cartes thermiques et les graphiques de régression. Cette expérience pratique améliorera non seulement vos compétences analytiques mais aussi votre capacité à communiquer vos découvertes à travers l'art du récit visuel.

Note :
Cette journée ne sera pas évaluée automatiquement via des tests unitaires. En effet, elle est conçue pour vous permettre d'explorer et de découvrir les capacités des Notebooks Jupyter et l'art de la visualisation des données. Cette expérience pratique est cruciale pour approfondir votre compréhension de ces outils. Bien qu'il n'y ait pas de notation automatisée, vos progrès et vos soumissions seront examinés manuellement pour s'assurer qu'un travail a été fourni tout au long de la journée. Profitez de cette opportunité pour appliquer de manière créative ce que vous avez appris et démontrer vos compétences analytiques ! Elles vous seront utiles pour vos projets à venir.

De plus, de la même façon que pour le jour 6, vous allez découvrir de multiples méthodes de visualisation de données, cependant, il y aura peu de détails sur pourquoi et / ou quand utiliser chaque type de visualisation. C'est à vous d'explorer chacune d'entre elles pour voir avec quel type de données elles fonctionnent le mieux.

Partie 0 : Prérequis

1. Jupyter et Notebooks

Il existe plusieurs façon de créer, éditer et exécuter des Notebooks. Voici quelques possibilités, vous êtes libres de les comparer et choisir celle qui vous convient le mieux :

- **JupyterLab** est le dernier outil en date officiel pour la gestion de Notebooks. C'est un IDE dédié aux Notebooks. Vous pouvez l'installer en suivant les étapes décrites ici. Une page de la documentation est également dédiée à JupyterLab.
- **PyCharm Professional** supporte nativement les Notebooks Jupyter. Il suffit de créer un projet Jupyter en suivant les étapes de la documentation officielle. Voici également une page de documentation expliquant l'interface dédiée pour les Notebooks.
- **Visual Studio Code** supporte nativement les Notebooks Jupyter, comme décrit dans la documentation.

2. Librairies Python

Pour cette journée, vous aurez besoin des librairies Python suivantes :

- pandas que vous connaissez déjà,
- ipywidgets, une librairie ajoutant des widgets interactifs pour les Notebooks Jupyter,
- Matplotlib, la librairie principale pour créer des visualisations en Python,
- seaborn, une librairie de visualisation de données avancée. Elle est basée sur Matplotlib. Les deux librairies peuvent être utilisées de paires afin d'avoir plus de possibilités de customisation dans les visualisations générées.

Ajoutez ces librairies au fichier *requirements.txt* comme vu lors des jours précédents et installez les prérequis dans un nouvel environnement virtuel pour la journée.

Partie 1 : Introduction aux Notebooks Jupyter

« Un notebook est un document partageable qui combine du code, des descriptions en langage clair, des données, des visualisations riches comme des modèles 3D, des graphiques, des diagrammes et des figures, ainsi que des contrôles interactifs. Un notebook, accompagné d'un éditeur (comme JupyterLab), offre un environnement interactif rapide pour le prototypage et l'explication du code, l'exploration et la visualisation des données, et le partage d'idées avec d'autres. » *Jupyter.org*

Il serait trop long d'expliquer et résumer ici ce que sont Jupyter et les Notebooks. Heureusement, le site officiel de Jupyter le fait pour nous, il est fortement recommandé que vous lisiez les pages suivantes avant de commencer cette journée :

- « What is Jupyter? »
- « What is a Notebook? »

Exercice 1 : Les différentes type de cellules

Rendu : partie1.ipynb

« Un notebook se compose d'une séquence de cellules. Une cellule est un champ de saisie de texte multiligne, et son contenu peut être exécuté en utilisant « Shift + Entrée » […]. Le comportement d'exécution d'une cellule est déterminé par le type de la cellule. Il existe trois types de cellules : les cellules de code, les cellules markdown et les cellules brutes. Chaque cellule commence en tant que cellule de code, mais son type peut être changé […]. » *Documentation Jupyter.org*

Créer un nouveau Notebook nommé *partie1.ipynb*. Réaliser les actions suivantes :

- Ajouter une cellule markdown contenant un titre de votre choix et une description (cette page décrit la syntaxe markdown si vous ne la connaissez pas),
- Ajouter une cellule de code pour afficher un texte de votre choix. « Hello World! » par exemple,
- Ajouter une cellule de code qui créé une liste contenant diverses valeurs numériques et l'afficher. Nommer la liste *my_list*. Indice : Vous n'avez pas besoin d'utiliser *print* pour afficher votre liste,
- Ajouter une cellule markdown décrivant les opérations effectuées dans l'action suivante,
- Ajouter une cellule de code qui pour *my_list*, calcule et affiche dans un tuple :
    - La somme de toutes les valeurs
    - La moyenne arrondie à la deuxième décimale
    - La valeur maximale
    - La valeur minimale

Indice : Les cellules sont indépendantes par défaut. Cela veut dire que vous pouvez exécuter le code de celles-ci dans n'importe quel ordre à une exception : si vous faites référence à une variable créée dans une cellule précédente, il faut d'abord exécuter la cellule dans laquelle la variable est créée avant d'exécuter celle qui y fait référence.
Il est cependant préférable d'exécuter les cellules dans leur ordre d'apparition afin d'éviter des erreurs. Par exemple, si une variable est modifiée dans une cellule puis que des calculs sont effectués dans une autre, les résultats des calculs peuvent être différents en fonction de si la cellule modifiant la variable a été exécutée ou non. Vous pouvez tester par vous-même en ajoutant une celle entre la création et les calculs de *my_list.*

Indice 2 : Pour réinitialiser le calculs de toutes les cellules, il faut redémarrer le kernel Jupyter.

Exercice 2 : Les commandes magiques

Rendu : partie1.ipynb

Les commandes magiques d'IPython (le kernel Python utilisé par Jupyter) étendent la syntaxe Python standard avec des améliorations puissantes. Ces commandes, précédées de % pour les commandes sur une seule ligne ou de %% pour les commandes sur toute la cellule, sont conçues pour résoudre de manière succincte divers problèmes courants dans l'analyse de données. Par exemple, %time mesure le temps d'exécution d'une seule instruction, tandis que %%timeit fournit un benchmark approfondi en exécutant un bloc de code plusieurs fois pour obtenir une métrique de performance plus précise. Time et timeit sont deux commandes qui peuvent être exécuté en ligne ou en bloc.
Vous pouvez retrouver une liste de toutes les commandes magiques dans la documentation de IPython.

Compléter le notebook *partie1.ipynb* avec les actions suivantes :

- Ajouter une cellule de code qui effectue une action prenant du temps, par exemple ouvrir un fichier *parquet* ou *CSV* de plusieurs centaines de Mo. Utiliser la commande magique time pour évaluer le temps d'exécution.
- Ajouter une cellule de code qui effectue plusieurs actions prenant du temps, par exemple faire des manipulations sur les fichiers précédemment ouverts. Utiliser la commande magique timeit pour évaluer le temps d'exécution sur 25 itérations et 20 répétitions.
- Explorer et tester d'autres commandes magiques

Exercice 3 : Exporter et partager un Notebook

Rendus :

- partie1.py
- partie1.html

Il est possible d'exporter les Notebooks dans divers formats statiques comme le Markdown, HTML, PDF permettant de partager des résultats / visualisation obtenue lors de l'exécution du Notebook, mais aussi en un script Python exécutable, permettant par exemple d'apprendir le développement dans un IDE « classique ».
Exporter le fichier *partie1.ipynb* en un script Python et un fichier HTML.

Note : Les méthodes magiques n'existent pas en Python, lors de l'export elles seront cependant toujours présentes. Pensez à les supprimer du script ou à les remplacer par un équivalent Python si elles sont cruciales pour votre code.

Exercice 4 : Widgets interactifs

Rendu : partie1.ipynb

« Les notebooks prennent vie lorsque des widgets interactifs sont utilisés. Les utilisateurs peuvent visualiser et manipuler leurs données de manière intuitive et facile. Les chercheurs peuvent facilement voir comment la modification des entrées d'un modèle impacte les résultats. Les scientifiques peuvent partager des résultats interactifs avec des interfaces graphiques que d'autres peuvent explorer sans voir le code. Explorer, apprendre et partager devient une expérience immersive et amusante. » *Jupyter Widgets Documentation*

Compléter le notebook *partie1.ipynb* avec les actions suivantes :

- Ajouter une cellule de code contenant un widget de type IntSlider. Ajouter au slider une valeur minimale et maximale, une valeur de départ, une incrémentation supérieure à 1 et une description,
- Ajouter une cellule de code affichant la valeur du widget précédent,
- Ajouter une cellule de code contenant un widget de type ToggleButton. Mettez le bouton par défaut à True, ajouter une description et un style au choix,
- Ajouter une cellule de code affichant la valeur du widget précédent.

Note : Vous aurez l'occasion de tester plus en profondeur les widgets lors de la partie suivante sur la visualisation des données.

Partie 2 : Visualisation de données basique

Exercice 0 : Mise en place

Créer trois Notebooks, un pour chaque dataset fourni avec le sujet :

- weather.csv weather.ipynb
- flights.parquet flights.ipynb
- sales.csv sales.ipynb

Pour chaque Notebook, créer une cellule et coder les actions suivantes dans celle-ci :

- Importer les librairies utiles (par exemple pandas, seaborn, …)
- Charger le dataset associé dans un DataFrame
- Afficher le DataFrame

Indice : Faire attention lors du chargement du dataset « sales » à faire en sorte que la région « NA » ne soit pas comptée comme une valeur manquante.

Exercice 1 : Histogramme

Pour chaque point principal, créer une nouvelle cellule de code ou markdown en fonction de ce qui est demandé.

Pour cet exercice, vous devez utiliser la fonction histplot de seaborn pour afficher les graphiques.

Dans le Notebook « flights » :

- Afficher un histogramme de la colonne représentant le temps passé par les avions dans les airs. Observez la répartition des données, elle semble suivre la loi de puissance.
- Copier le DataFrame, et sur la même colonne que précédemment :
  - Afficher la variance de la colonne,
  - Supprimer toutes les valeurs valant 0 ou moins,
  - Appliquer une normalisation logarithmique,
  - Afficher de nouveau la variance de la colonne,
  - Afficher un histogramme de la colonne modifiée,
  - Observer la nouvelle répartition des données qui devrait être beaucoup plus équilibrée.
- Sur le DataFrame initial et toujours la même colonne :
  - Grouper l'abscisse par échelon de 30 minutes passé en vol (0 - 30, 30 - 60, 60 - 90, …),
  - Afficher un histogramme de la colonne groupée par les échelons définis précédemment,

- Faire en sorte que tous les échelons soient notés sur l'abscisse avec une rotation de 45°,
- Ajouter un titre personnalisé pour le graphique, ainsi que pour l'abscisse et l'ordonnée.

Note : Le troisième point va demander un peu de calculs et de réflexion afin d'être sûr que vous regroupez bien l'ensemble des données et ce dans des échelons fixes.

Indice : Pour le troisième point, vous allez avoir besoin de la méthode xticks de matplotlib afin d'afficher tous les échelons.

Indice 2 : Vous pouvez utiliser la méthode figure de matplotlib avant d'appeler la méthode d'affichage du graphique seaborn afin de changer le ratio d'affichage du graphique.

Exercice 2 : Nuage de points

Pour cet exercice, vous devez utiliser la fonction scatterplot de seaborn pour afficher les graphiques.

Dans le Notebook « flights » :

- Afficher un nuage de point montrant la relation entre le temps passé dans les airs (x) et la distance parcourue (y). Mettre des titres personnalisés sur le graphique.
- Afficher un nouveau nuage de point montrant cette fois-ci la relation entre le retard au départ (x) avec le retard à l'arrivée (y).
- Observez les graphiques et tirez-en vos conclusions sur les relations affichées. Écrire vos conclusions dans une cellule markdown.

Dans le Notebook « weather » :

- Afficher un nuage de point montrant la répartition des températures minimales (x) et maximales (y) réparties par saison pour une ville donnée

Exercice 3 : Graphique camembert

La librairie seaborn ne gère pas nativement les graphiques en camembert, pour cet exercice, il va donc falloir utiliser matplotlib et sa méthode pie. Contrairement à seaborn, matplotlib ne fait pas les agrégations et autres calculs pour vous lorsque vous appelez une fonction pour créer un graphique. Ce sera donc à vous de faire les calculs avant, puis de passer le résultat à matplotlib pour l'affichage.

Dans le Notebook « sales » :

- Afficher la répartition des ventes (prix total) par région de vente. Afficher le pourcentage de chaque catégorie dans les parts du camembert,
- Afficher la répartition des ventes (prix total) par catégorie de produit. Afficher le pourcentage de chaque catégorie dans les parts du camembert,
- Noter vos conclusions / observations pour cet exercice.

Exercice 4 : Graphique de distribution

Pour cet exercice, vous devez utiliser la fonction displot de seaborn pour afficher les graphiques.

Dans le Notebook « sales » :

- Afficher la distribution du prix total de vente,
- Afficher la distribution du prix total de vente par catégorie,
- Pour chaque région, afficher la distribution du prix total de vente par catégorie (Indice : il suffit d'ajouter un paramètre de plus à la méthode afin de générer un graphique par région),
- Afficher la distribution du prix total de vente par région,
- Afficher la distribution du prix total de vente par « dealsize »,
- Pour chaque région, afficher la distribution du prix total de vente par « dealsize »,
- Utiliser la fonction histplot pour afficher la distribution du prix total de vente par « dealsize ». Indice : Tester le paramètre « multiple » afin d'améliorer l'affichage,
- Noter vos conclusions / observations pour cet exercice.

Dans le Notebook « weather » :

- Afficher la distribution des températures minimales (x) et maximales (y) réparties saison pour une ville donnée,
- Afficher la distribution de la température moyenne par saison pour une ville donnée

Indice : La méthode displot à trois modes d'affichage, tester ceux-ci et choisir celui qui vous semble le plus compréhensible.

Exercice 5 : Boîte à moustaches

Une boîte à moustaches (Box plot en anglais) est une représentation graphique permettant d'observer simplement et rapidement la médiane, les quartiles haut et bas ainsi que les bornes permettant d'identifier les valeurs aberrantes (comme vu lors de la journée précédente). La page Wikipédia explique très bien le concept.

Pour cet exercice, vous devez utiliser la fonction boxplot de seaborn pour afficher les graphiques.

Dans le Notebook « weather », afficher une boîte à moustache pour :

- La vitesse moyenne du vent séparé par ville
- La vitesse moyenne du vent pour une ville au choix séparé par saison
- La température moyenne séparé par ville
- La température moyenne pour une ville au choix séparé par saison

Dans le Notebook « sales », afficher une boîte à moustache pour :

- La quantité vendue par région
- Le prix total de vente par région

Partie 3 : Visualisation de données avancées

Exercice 1 : Graphique de régression

Il est possible avec seaborn, de calculer et d'afficher automatiquement des modèles de régression. Cela permet d'observer facilement des tendances dans les données analysées. Il existe deux méthodes dans seaborn permettant d'afficher des modèles de regression : regplot et lmplot. Pour cet exercice, vous devez utiliser la deuxième.

Dans le Notebook « weather » :

- Calculer la température moyenne pour chaque année pour une ville donnée. Supprimer les données manquantes puis afficher le modèle de régression associé,
- Calculer la température moyenne par saison et par année pour une ville donnée. Supprimer les données manquantes puis afficher le modèle de régression associé,
- Noter vos conclusions / observations pour cet exercice.

Afin de choisir la ville pour laquelle on affiche le modèle de régression, ajouter un widget interactif permettant de sélectionner un élément (radio ou dropdown par exemple). Vous aurez surement besoin de la méthode interact.

Exercice 2 : Graphique de paires

Pour cet exercice, vous devez utiliser la fonction pairplot de seaborn pour afficher les graphiques. Pour chaque Notebook, notez vos conclusions dans une cellule markdown après l'affichage du graphique de paires demandé.

Dans le Notebook « flights », afficher un graphique de paires pour la relation entre les colonnes « AIR_TIME », « DISTANCE », « DEP_DELAY » et « ARR_DELAY ».

Indice : Vous devriez obtenir les mêmes résultats à ceux de l'exercice 2 de la partie précédentes pour les relations AIR_TIME / DISTANCE et DEP_DELAY / ARR_DELAY.

Dans le Notebook « sales », afficher la relation entre les colonnes « quantity », « unit_price », « shipment_price » et « total_price » par catégorie.

Dans le Notebook « weather », afficher la relation entre les colonnes « avg_tmp_c », « min_temp_c », « max_temp_c » et « precipitation_mm » par saison et pour une ville donnée.

Exercice 3 : Graphique de données temporelles

Dans une seule cellule du Notebook « weather », effectuer les actions suivantes :

- Calculer pour chaque ville une moyenne mobile courte de 7 jours et une moyenne mobile longue de 30 jours des moyennes de températures. Ajouter ces moyennes dans le DataFrame dans deux nouvelles colonnes,
- Filtrer le DataFrame pour ne conserver que les informations comprises entre 2 années (2022 et 2023 par exemple),
- Utiliser la méthode melt pour regrouper toutes les valeurs (SMA courte, SMA longue, températures moyennes) par date et par ville,
- Filtrer le DataFrame par nom de ville dans l'ordre croissant,
- Ajouter une nouvelle colonne étant la fusion du nom des villes et des valeurs résultant de melt (« sydney_avg_temp_c » par exemple),
- Afficher un graphique avec trois courbes par ville (SMA courte, SMA longue, températures moyennes) avec les dates en abscisse et les températures en ordonné.

Indice : Le DataFrame résultant de la méthode melt devrait ressembler à ceci :

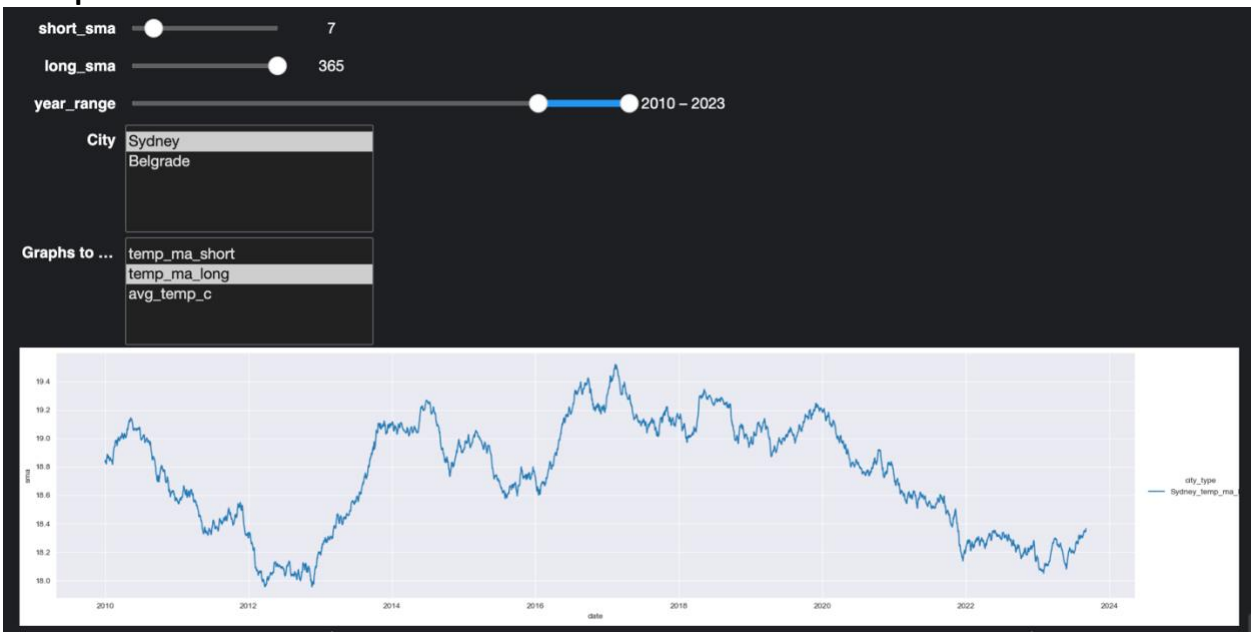
```
           date city_name       sma_type        sma
0    2022-01-01  Belgrade  temp_ma_short   5.200000
2034 2023-02-09  Belgrade   temp_ma_long   3.390000
2035 2023-02-10  Belgrade   temp_ma_long   3.213333
2038 2023-02-11  Belgrade   temp_ma_long   3.063333
2040 2023-02-12  Belgrade   temp_ma_long   2.993333
...         ...       ...            ...        ...
2147 2023-04-07    Sydney   temp_ma_long  21.926667
2146 2023-04-06    Sydney   temp_ma_long  22.076667
764  2023-01-18    Sydney  temp_ma_short  24.071429
2141 2023-04-04    Sydney   temp_ma_long  22.573333
3674 2023-09-05    Sydney     avg_temp_c  16.500000

[3675 rows x 4 columns]
```

Ajouter des widgets à la cellule précédente afin de laisser l'utilisateur changer les paramètres suivants :

- La taille des fenêtres pour les moyennes mobiles :
  - Courte : minimum 3 maximum 30 valeur par défaut 7
  - Longue : minimum 15 maximum 365 valeur par défaut 30
- Les bornes annuelles pour l'affichage des données (2022 – 2023 par défaut). L'année minimale doit être la plus petite année contenant des données et de même pour l'année maximale,
- Les villes affichées, une ou plusieurs parmi les villes présentes dans le DataFrame,
- Les courbes affichées, une ou plusieurs parmi : SMA Courte, SMA longue, moyenne de températures.

Exemple :



Exercice 4 : Carte thermique (heat map)

Créer un nouveau Notebook nommé « house.ipynb » et charger le dataset « house_prices.parquet ». Afficher une carte thermique de l'influence de chaque colonne sur le prix des maisons à l'aide des méthodes corrwith et heatmap.

Indice : Certaines colonnes sont composées de texte (« yes / no », …), il n'est pas possible de calculer des corrélations avec des valeurs non-numériques. Vous pouvez cependant facilement convertir ces valeurs sous un format numérique (« yes » -> 1, « no » -> 0 par exemple).

Going further : mini-projet

Rendu : *going_further.ipynb* – Un Jupyter Notebook contenant tout le code, les visualisations et les explications narratives des étapes de votre projet.

Le but de ce mini-projet est de synthétiser les connaissances et les compétences que vous avez développées au cours des 7 derniers jours. En sélectionnant un ensemble de données, une API ou un site web de votre choix, vous allez récupérer des données et appliquer diverses manipulations, nettoyages et transformations. Cela afin d'analyser via des visualisations les données et en tirer des conclusions. Ce projet pratique vise à consolider votre compréhension du parcours de traitement des données, de l'acquisition à l'analyse et la visualisation.

1. Acquisition de données

Choisissez une source pour vos données. Cela peut être un ensemble de données disponible publiquement, des données accessibles via une API, ou des données extraites d'un site web.

Pensez à ce qui vous intéresse — des statistiques sportives, des données financières (bourses, crypto, …), des données de réseaux sociaux, ou autre — et trouvez une source qui peut fournir ces données.

2. Nettoyage et transformation des données

Importez vos données sélectionnées dans Python / Jupyter en utilisant la méthode appropriée (fonctions de lecture de fichiers, requêtes API, ou outils de scraping web). Réalisez les processus de nettoyage nécessaires pour gérer les problèmes tels que les valeurs manquantes, les doublons et les types de données incorrects. Transformez les données comme nécessaire pour les préparer à l'analyse.

3. Analyse et visualisation des données

Appliquez les techniques de manipulation de données que vous avez apprises pour explorer les données de manière statistique. Utilisez Seaborn et / ou Matplotlib pour créer une variété de graphiques pour analyser vos données. Essayez de découvrir des tendances, des relations ou des anomalies dans les données.

4. Documentation et présentation

Documentez vos découvertes dans un Jupyter Notebook. Assurez-vous que votre notebook comprend :

- Une introduction expliquant votre source de données et votre intérêt pour celle-ci,
- Des sections pour le nettoyage des données, la transformation et la visualisation, chacune accompagnée d'un commentaire approprié expliquant votre processus et vos découvertes,
- Des conclusions qui récapitulent vos idées et conclusions tirés de l'analyse des données.

Ce mini-projet est l'occasion de démontrer votre capacité à gérer un projet de Big Data de bout en bout, de la collecte initiale des données à la présentation finale des résultats. Il mettra à l'épreuve vos compétences techniques, votre pensée analytique et votre capacité à communiquer des informations complexes de manière claire et efficace.