المملكة المغربية
ROYAUME DU MAROC

INSTITUT AGRONOMIQUE
ET VETERINAIRE HASSAN II

معهد الحسن الثاني
للزراعة والبيطرة

Rapport de la réalisation d'un

Geoanalytical Dashboard

Préparé par :

El faouz imane N°21

Essouti Laila N° 35

Sous l'encadrement :

M .HAJJI Hicham

Adresse : Madinat Al Irfane, B.P. 6202. Rabat – Maroc
Tél : (00 212) 0537 77 17 58/59
Fax : (00 212) 0537 77 58 45
Site web : http://www.iav.ac.ma

العنوان: ص. ب 6202 الرباط المعاهد الرباط – المغرب
الهاتف: 59 /58 77 17 0537 (00 212)
الفاكس: 45 58 770537 (00 212)
موقع الأنترنت:http://www.iav.ac.ma

REMERCIEMENT :

Nous tenons à exprimer nos sincères remerciements au Professeur Hajji Hicham pour son accompagnement précieux et son soutien continu tout au long du cours de webmapping. Sa passion, son expertise et sa dévotion ont été des atouts inestimables pour nous guider dans la compréhension approfondie des concepts complexes liés à la cartographie en ligne. Ses conseils éclairés et son dévouement à nous pousser vers l'excellence ont grandement enrichi notre parcours académique. Nous lui sommes reconnaissants pour son engagement indéfectible envers notre apprentissage et sa disponibilité à répondre à nos interrogations, nous inspirant ainsi à approfondir nos connaissances dans ce domaine captivant du webmapping.

Table de matière :

Liste des figures

INTRODUCTION :

Ce rapport présente en détail le processus complet de création d'un Dashboard GeoAnalytique basé sur la bibliothèque Streamlit et ses extensions géospatiales. L'objectif principal de cette initiative était de concevoir un tableau de bord offrant une palette de fonctionnalités avancées pour la manipulation et la représentation de données géospatiales. Le schéma de données retenu pour ce projet comporte une dimension spatio-temporelle, intégrant des attributs variés tels que la géométrie de type point, des données numériques et textuelles, ainsi qu'une séquence temporelle sur plusieurs jours. Au fil de ce rapport, nous détaillerons les étapes clés du projet,

I. Création des données :

Pour générer les données, nous avons employé un script Python qui répartit 1500 points répartis sur Maroc, en leur assignant les attributs requis. Voici les étapes que nous avons suivies pour ce faire

- Charge un polygone à partir d'un fichier shapefile pour définir une zone.
- Génère aléatoirement 1500 points à l'intérieur de ce polygone.
- Crée des données aléatoires pour ces points, y compris des dates, des noms, des mesures de météo (température, humidité, pression atmosphérique, pluviométrie) et des réflectances.
- Crée un DataFrame Pandas avec ces données.
- Convertit ce DataFrame en un GeoDataFrame à l'aide de GeoPandas pour conserver les informations géospatiales.
- Écrit ce GeoDataFrame dans un fichier GeoParquet.

II. L’utilité de geoparquet :

Le format GeoParquet présente une approche novatrice en stockant les données géospatiales selon un schéma prédéfini, privilégiant une disposition en colonnes plutôt qu'en lignes, ce qui facilite la gestion efficace de vastes ensembles de données géographiques. Contrairement aux formats classiques, GeoParquet compresse l'ensemble des informations dans un unique fichier, réduisant ainsi l'espace de stockage nécessaire sans altérer la qualité ni l'accessibilité des données. Cette méthode optimisée offre une structure organisée, favorisant la gestion et l'exploitation optimale des données géographiques, tout en intégrant des fonctionnalités de compression performantes.

La conversion de données de geoparquet en SHP a produit quatre fichiers (CPG, DBF, PRJ, SHP, SHX) totalisant 985 Ko, tandis que le fichier geoparquet original ne fait que 391 Ko. Cela souligne l'avantage de la réduction significative de la taille de stockage avec geoparquet, qui offre également la commodité d'un seul fichier pour le transfert et le stockage des données.En évaluant la performance pour l'application de visualisation mentionnée dans le titre à venir, il est notable que le temps d'affichage des données SHP est plus long que celui des données geoparquet. Cette observation souligne la supériorité de geoparquet en termes de rapidité de traitement par rapport au format SHP spécifiquement dans le cadre de cette application.

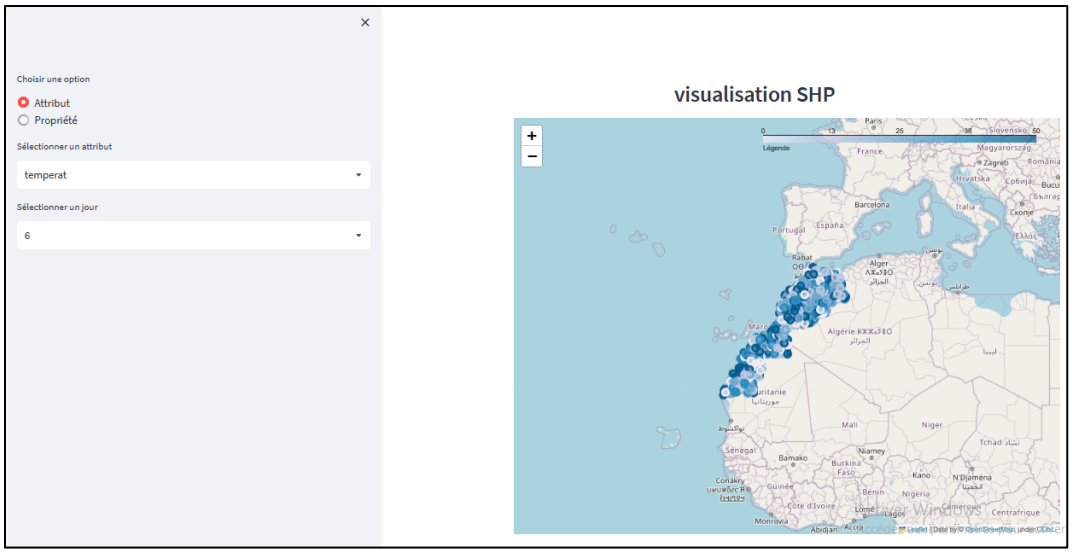


Figure 1:visualisation par les données shp

III. Cartographie des données :

- Exploration des données

A l'aide de la bibliothèque python matplotlib.pyplot on a pu afficher les histogrammes de données

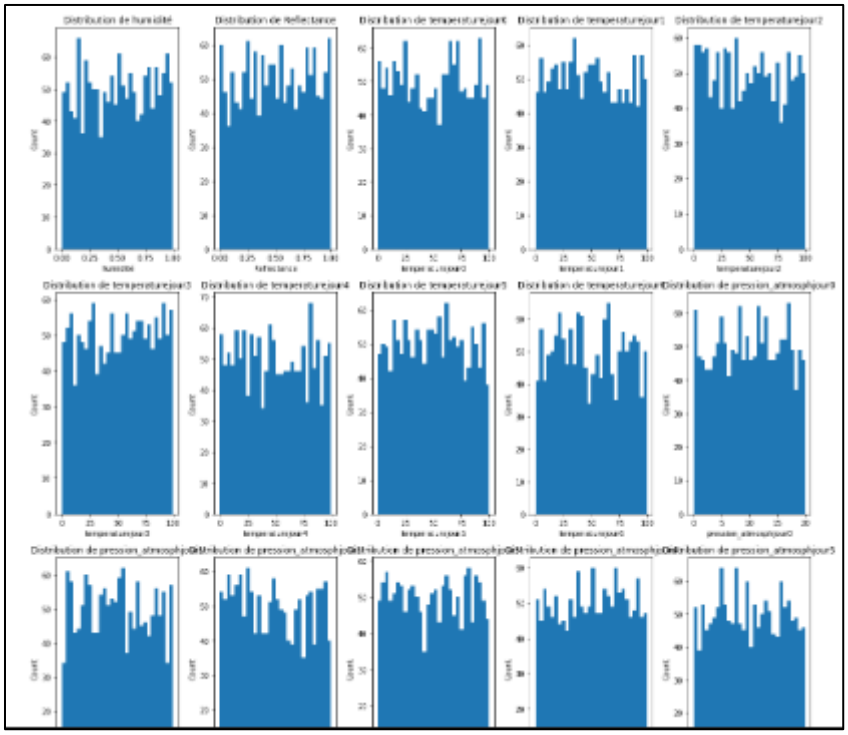


Figure 2:extrait des histogrammes des attributs

- Choix de la méthode de classification

On remarque que les distributions de données ne suivent ni une structure homogène ni une distribution normale. Dans ce contexte, la méthode de classification la plus appropriée pourrait être la détection de ruptures naturelles.

- Créer une application par streamlit qui permet de visualiser les données :

L'application Streamlit créée permet aux utilisateurs de choisir entre les options "Attribut" ou "Propriété". En sélectionnant "Attribut", ils peuvent opter pour la température, la pression atmosphérique ou la pluviométrie avec une journée spécifique, tandis que "Propriété" offre le choix entre humidité et réflectance. Les données sont ensuite traitées en utilisant la méthode de classification de Jenks pour regrouper les valeurs en cinq classes distinctes en fonction des attributs ou propriétés sélectionnés. Une carte interactive Folium du Maroc est générée, affichant les données géospatiales sous forme de marqueurs de cercle colorés selon les classes déterminées par la méthode de Jenks , et une légende est intégrée pour clarifier la signification des différentes couleurs utilisées pour les classes.

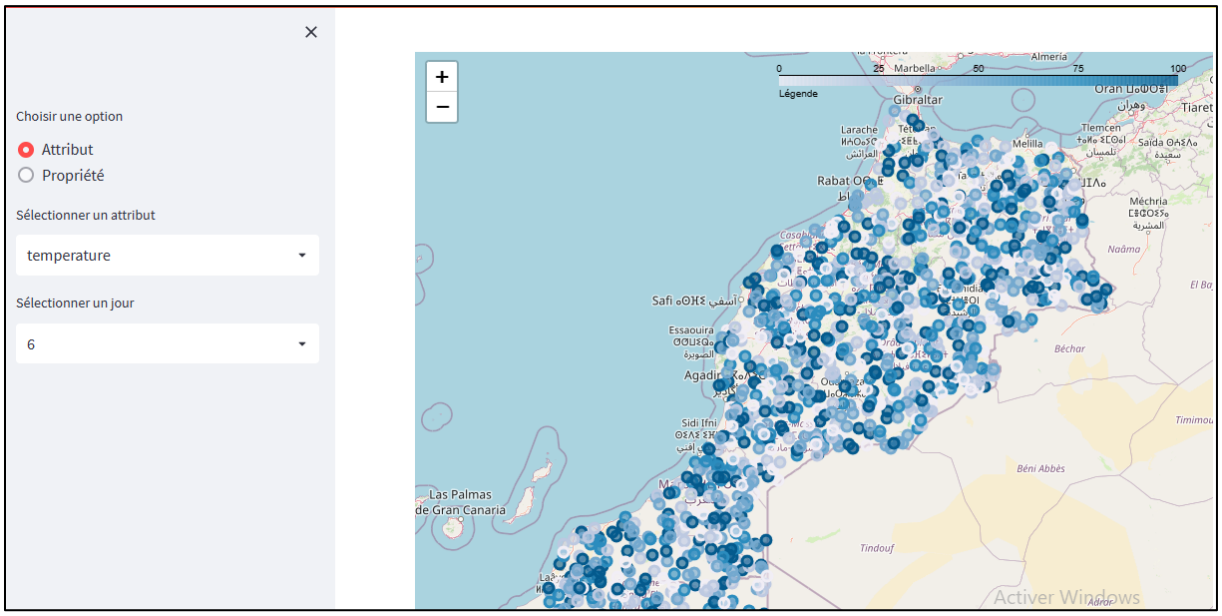


Figure 3:interface de l'application de visualisation de données

IV. Slider :

- Dans ArcGIS, nous avons utilisé la méthode d'interpolation par distance inverse pour générer des images raster au format TIFF pour chaque attribut
- Créer une interface qui permet à l'utilisateur de sélectionner un attribut parmi la température, la pression atmosphérique ou la pluviométrie, ainsi que de choisir un jour spécifique à l'aide d'un curseur.
- En fonction de ces sélections, il charge les données géospatiales correspondantes depuis un fichier GeoParquet et un raster (image) TIFF associé à l'attribut et au jour choisis. Il utilise rasterio pour lire et traiter les données raster.
- Générer une carte interactive Folium du Maroc en superposant l'image raster avec la carte de base et crée une légende avec une échelle de couleurs pour interpréter les valeurs de l'image raster

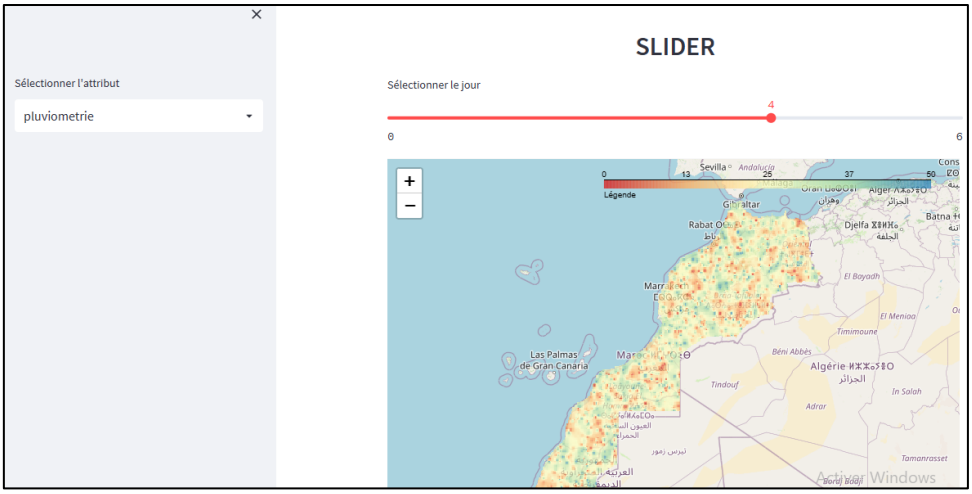


Figure 4:interface de l'application du slider

V. Timelapse :

Pour le time laps on opter pour deux méthodes la première est basé sur la création d'un GIF superpose avec la carte de base et la deuxième est basé sur la création d'un vidéo a partir des images raster superposé sur la map :

- Créer une interface qui permet au l'utilisateur de choisir un attribut parmi la température, la pression atmosphérique ou la pluviométrie à l'aide d'une barre latérale.
- Pour la première méthode : à travers la fonction create_timelapse() on parcourt les fichiers image raster correspondant à chaque jour et attribut choisi. Pour chaque image, puis on fusionne les différentes bandes d'image, insère un texte annoté contenant le nom de l'attribut et du jour associé, Ensuite on compile ces images en une séquence pour former un GIF représentant l'évolution temporelle des données et finalement, ce GIF est ensuite intégré à une carte interactive via la bibliothèque Folium
- Pour la deuxième méthode on parcourt les fichiers image raster correspondant à chaque jour et attribut choisi, puis les images raster sont superposées sur une carte interactive Folium, qui inclut une légende pour visualiser les différentes plages de valeurs. Ensuite, on utilise la fonction FuncAnimation pour créer une vidéo timelapse à partir de ces images. Enfin, ces vidéos timelapse sont affichées dans l'interface Streamlit

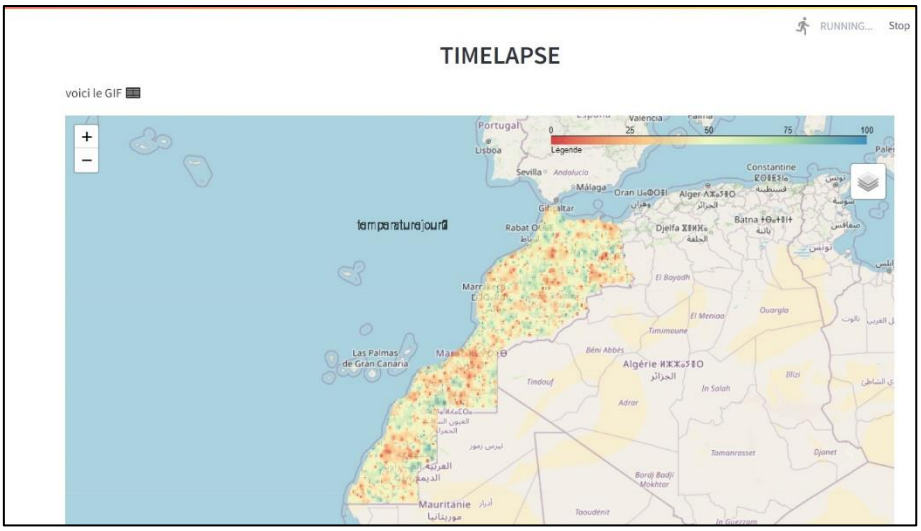


Figure 5:interface pour le timelaps par GIF

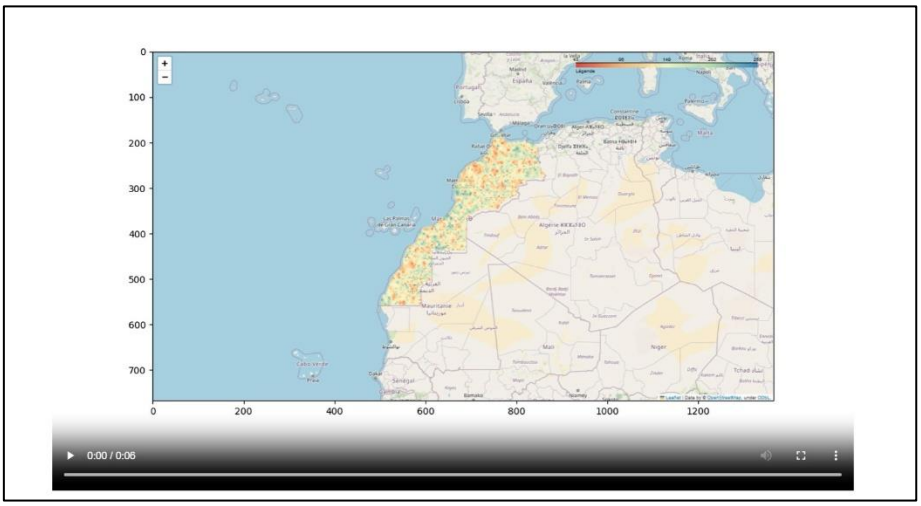


Figure 6:interface pour timelaps en video

VI. SplitMap :

- Tout d'abord, les images ont été hébergées sur GitHub.
- Ensuite, une interface a été créée permettant à l'utilisateur de choisir parmi la température, la pression atmosphérique ou la pluviométrie en plus de sélectionner les deux jours à représenter à l'aide de deux curseurs.
- Utiliser Leafmap pour afficher une carte interactive où les données géospatiales pour deux jours sélectionnés sont présentées en deux sections distinctes : gauche et droite. Chaque section représente les données correspondant à un jour spécifique pour l'attribut choisi, permettant à l'utilisateur de comparer visuellement les variations de cet attribut entre les deux jours.
- Une légende est ajoutée à la carte pour interpréter les classes de valeurs. Les valeurs sont réparties en classes basées sur les seuils définis pour chaque attribut.

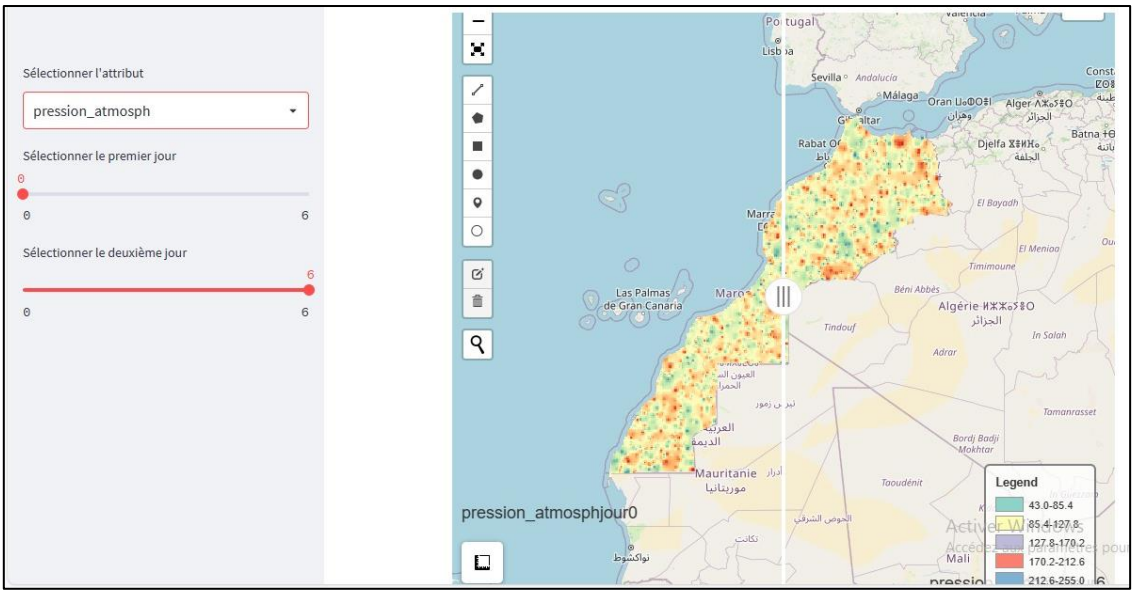


Figure 7:interface de l'application de SPLIT_MAP

VII. Pop-up :

- Créer une interface pour visualiser la carte
- Crée une carte Folium où chaque point géospatial est représenté par un marqueur. De plus, pour chaque point, un graphique Altair est généré afin d'illustrer l'évolution des attributs tels que la température, la pression atmosphérique et la pluviométrie sur une période de six jours.

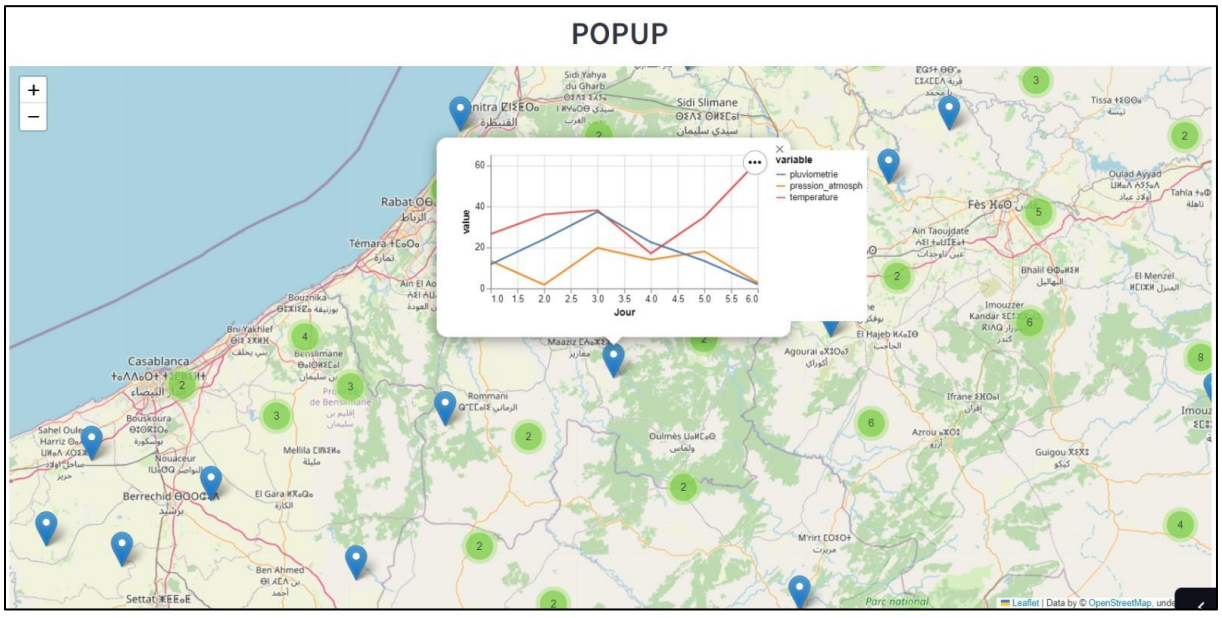


Figure 8: interface de l'application de popup

VIII. Recherche des données par coordonnées :

Dans la même interface de visualisation de données, un texte-box a été intégré pour la recherche. Cette fonctionnalité permet à l'utilisateur d'entrer les coordonnées de longitude et de latitude du point recherché. Si le point est trouvé, il sera marqué sur la carte à l'aide d'un marqueur.

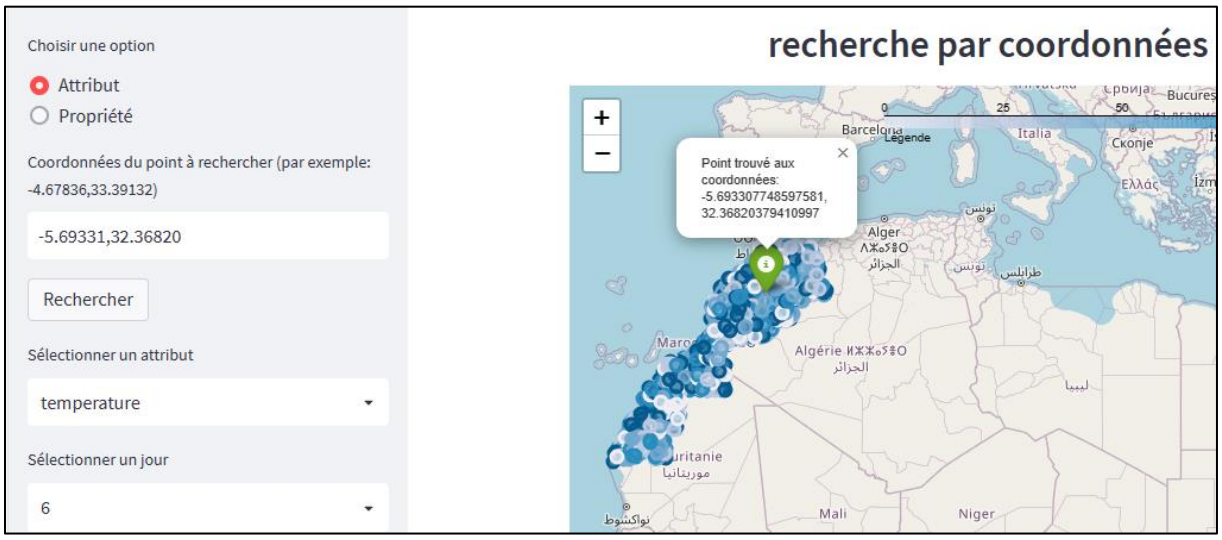


Figure 9:interface de l'application de recherche par coordonnées

IX. Les requêtes :

- Créer une interface permettant de composer des requêtes en sélectionnant des propriétés, des opérateurs et des valeurs pour filtrer les données.
- Une fois les filtres définis, il affiche le nombre de points filtrés. En utilisant les filtres spécifiés, la carte est générée avec des marqueurs circulaires représentant les données filtrées.

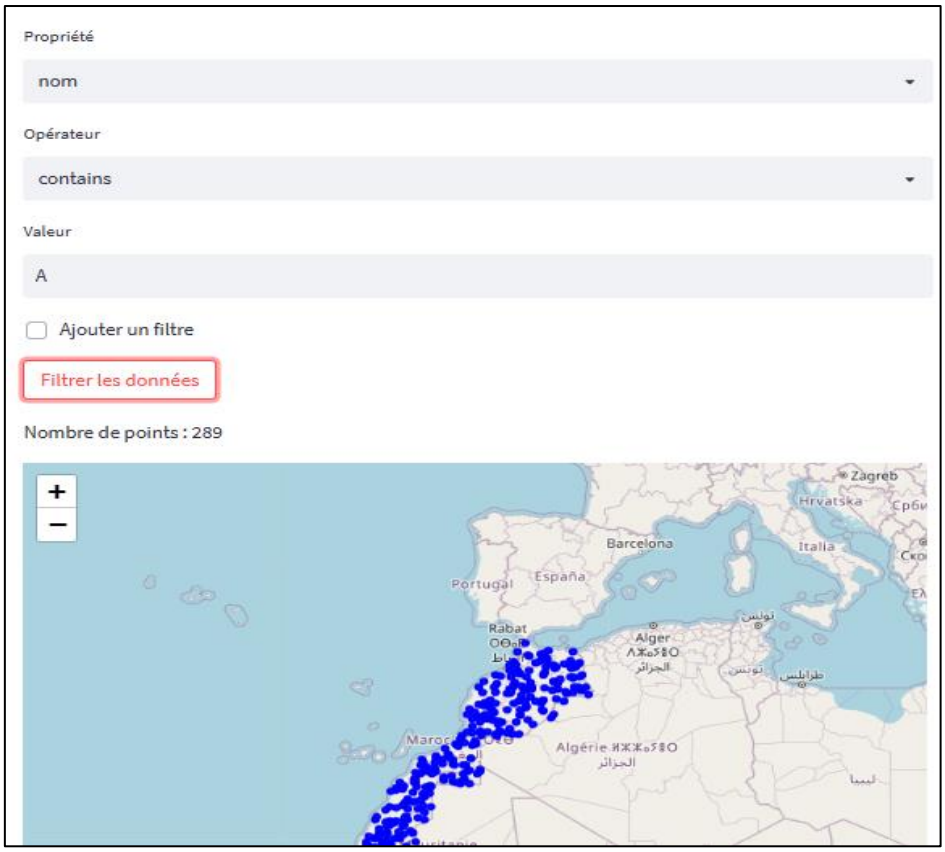


Figure 10:interface de l'application des requêtes

X. Explorer l'utilisation du format COG :

Le COG (Cloud Optimized GeoTIFF) est un format de fichier raster géospatial optimisé pour le stockage et la diffusion sur le cloud. Il stocke les données raster de manière structurée en utilisant une hiérarchie interne qui permet d'accéder efficacement à différentes parties de l'image sans avoir besoin de charger l'intégralité du fichier. Les informations géospatiales sont organisées en une série de tuiles, ce qui facilite un accès rapide et sélectif aux données en fonction des besoins, réduisant ainsi la quantité de données chargées pour une manipulation spécifique.

Pour explorer ce format on a suivi les étapes suivantes :

- Premièrement on a converti les images geotif en cog en utilisant la fonction leafmap.numpy_to_cog

```python
import leafmap

folder = "RASTERS"  # Assurez-vous d'utiliser le bon chemin

# Boucle pour convertir chaque fichier tiff en COG
for attribute in ['temperature', 'pression_atmosph', 'pluviometrie']:
    for i in range(6):
        in_tiff = f"{folder}/{attribute.lower()}jour{i}.tif"  # Chemin
        out_cog = f"{folder}/cog{attribute.lower()}jour{i}.tif"  # Chem

        # Convertir le fichier tiff en COG
        arr = leafmap.image_to_numpy(in_tiff)
        leafmap.numpy_to_cog(arr, out_cog, profile=in_tiff)
```

- Deuxièmement on a utilisé ces images pour l'application de slider et splitmap

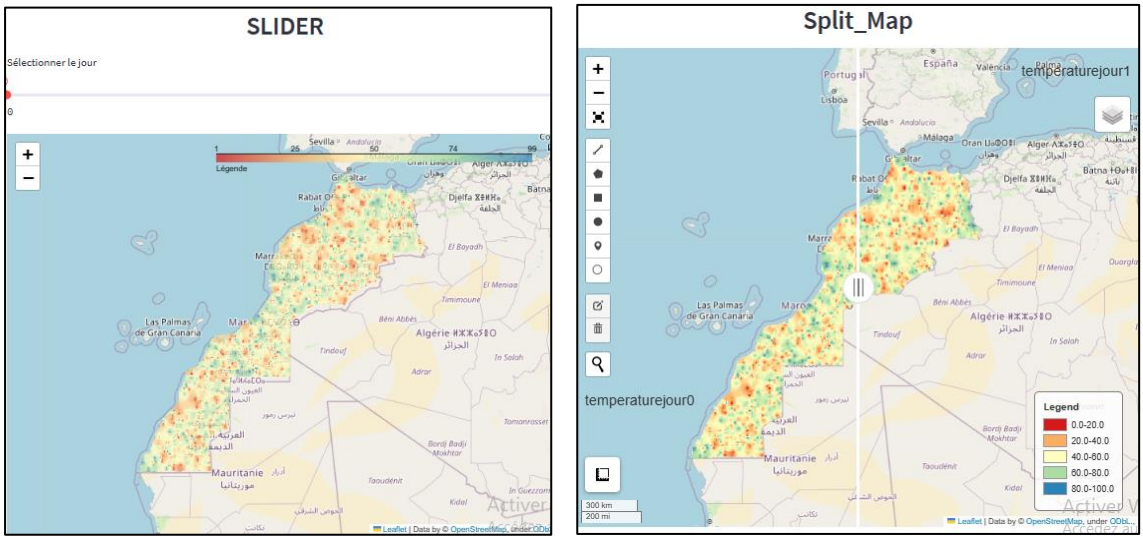


Figure 11:l'utilisation du COG pour les applications splitmap et slider

Comparaison entre geotiff et COG :

→Au niveau de stockage :la taille de l'image a diminué de 261 Ko à 41 Ko, soit une optimisation de 84%. Ce potentiel d'optimisation représente un impact significatif, surtout dans le domaine du big data où chaque réduction de taille peut avoir des répercussions majeures.
→Au niveau des applications : la durée d'exécution des applications utilisant le COG et celles employant le GeoTIFF est pratiquement similaire, peut être attribuée à la petite taille des images. Cependant, des écarts de temps significatifs se manifestent lors de la manipulation d'images de plus grande taille.

XI. Exporter le contenu en format pdf

- Créer la même interface de visualisation de données

- Utilisation de Tkinter pour capturer une zone spécifique de l'écran et obtenir une capture d'écran.
- Création d'un PDF avec une image de capture d'écran et une description détaillée de l'application de visualisation géospatiale.

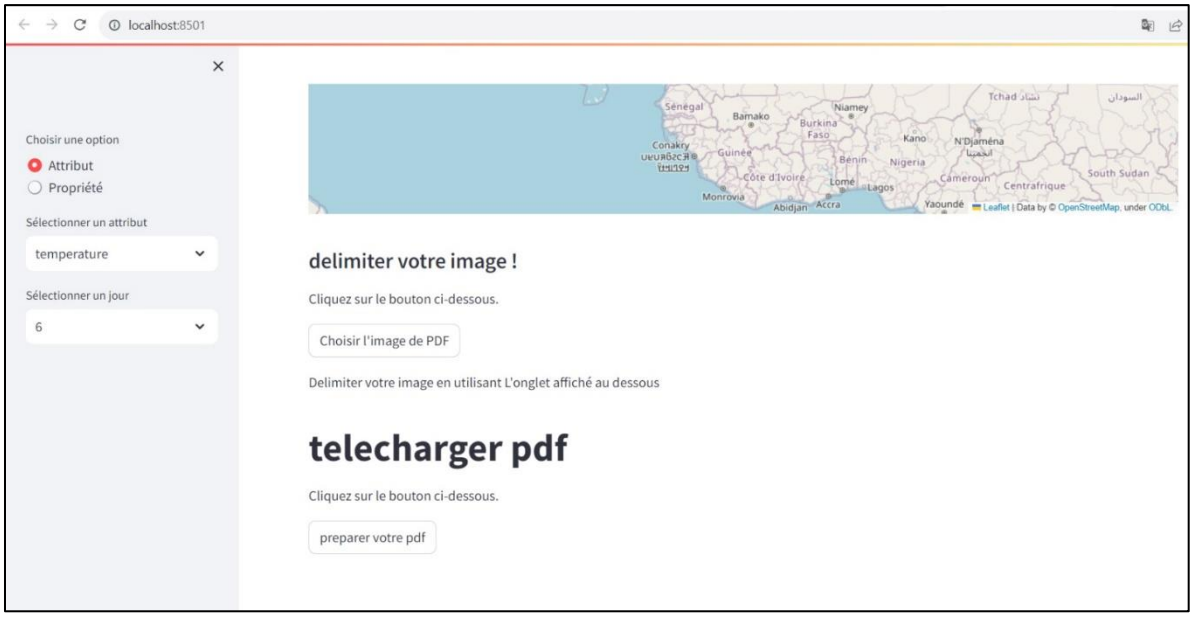


Figure 12:interface de l'application de création du pdf

XII. La creation du dashboard

Le déploiement des applications :

Les étapes de déploiement des applications ont suivi un processus méthodique :

- Premièrement héberger le dossier des applications et des données dans github
- Ensuite, un fichier requirement.txt a été préparé pour répertorier les bibliothèques utilisées avec leurs versions dans toutes les applications.
- Un compte a été créé sur Streamlit via GitHub.
- Sur la plateforme web de Streamlit, une nouvelle application a été ajoutée en spécifiant le chemin vers le répertoire de l'application. Ensuite, le déploiement de l'application a été effectué, fournissant ainsi un lien direct vers celle-ci.

La création d'une page HTML :

On crée une page html qui nous permet d'assembler toutes les applications en introduisant le lien de chacune

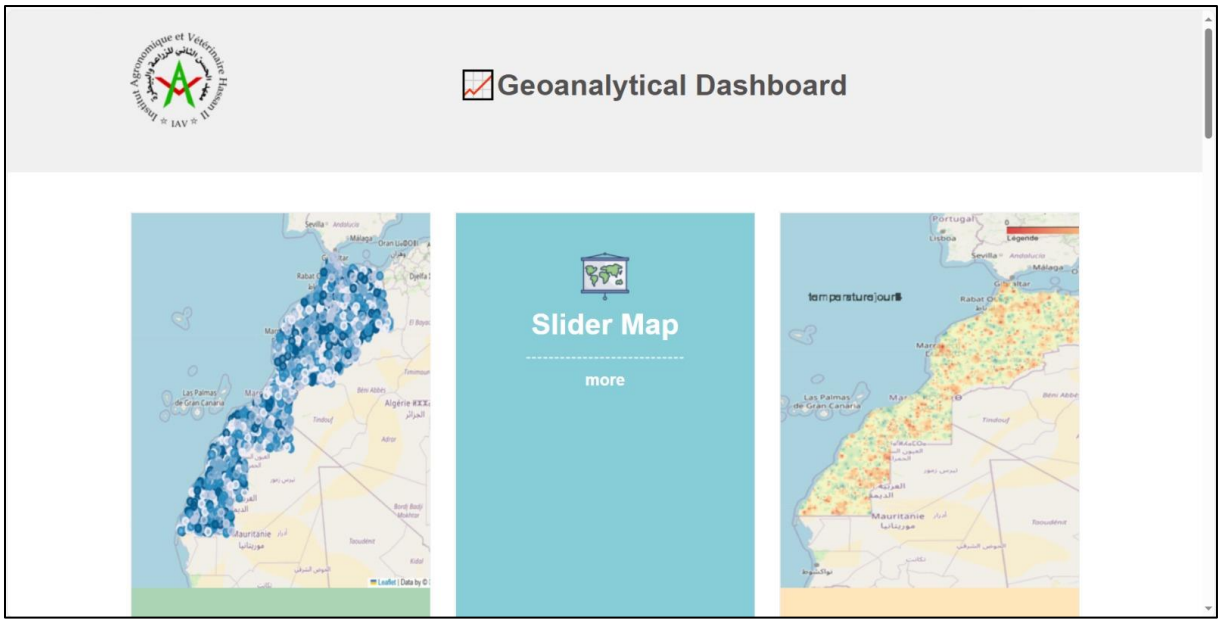


Figure 13:l'interface du geoanalytical dashboard

Conclusion :

En conclusion Ce projet a été une expérience exceptionnelle qui a considérablement enrichi notre expertise en matière de webmapping. En explorant et en mettant en œuvre des fonctionnalités avancées de visualisation géospatiale à travers le développement de ce Dashboard GeoAnalytique, nous avons acquis une compréhension approfondie des défis et des opportunités inhérents à ce domaine. La manipulation des données spatio-temporelles, l'utilisation de bibliothèques telles que Streamlit et ses extensions géospatiales, ainsi que la création d'un outil interactif ont été des étapes formatrices.

ANNEXES :

Code pour la creation des points :

```
import geopandas as gpd
from shapely.geometry import Point
import random
import pyarrow as pa
```

```
import pyarrow.parquet as pq
import pandas as pd
import datetime
import string

# Charger le polygone à partir du fichier shapefile
path_to_shapefile = "MAROC.shp"
polygons = gpd.read_file(path_to_shapefile)

# Assurez-vous qu'il y a un seul polygone
if len(polygons) != 1:
    raise ValueError("Le fichier shapefile ne contient pas un seul
polygone.")
polygon = polygons.geometry.iloc[0]

# Obtenir le CRS du polygone
crs = polygons.crs

# Définir les limites du polygone pour la génération aléatoire
min_x, min_y, max_x, max_y = polygon.bounds

# Générer 1000 points aléatoires à l'intérieur du polygone
random_points = []
while len(random_points) < 1500:
    random_point = Point(random.uniform(min_x, max_x),
random.uniform(min_y, max_y))
    if polygon.contains(random_point):
        random_points.append(random_point)

# Définir les colonnes d'attributs pour chaque point
attrib_data = {
    'date': [datetime.datetime(2022, 1, 1) +
datetime.timedelta(days=random.randint(1, 365)) for _ in range(1500)],
    'nom': [''.join(random.choice(string.ascii_letters) for _ in
range(10))for _ in range(1500)],
    'humidité': [random.uniform(0, 1) for _ in range(1500)],
    'Reflectance': [random.uniform(0, 1) for _ in range(1500)],
    'temperaturejour0': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour1': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour2': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour3': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour4': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour5': [random.uniform(0, 100) for _ in range(1500)],
    'temperaturejour6': [random.uniform(0, 100) for _ in range(1500)],
    'pression_atmosphjour0': [random.uniform(0, 20) for _ in
range(1500)],
    'pression_atmosphjour1': [random.uniform(0, 20) for _ in
range(1500)],
```

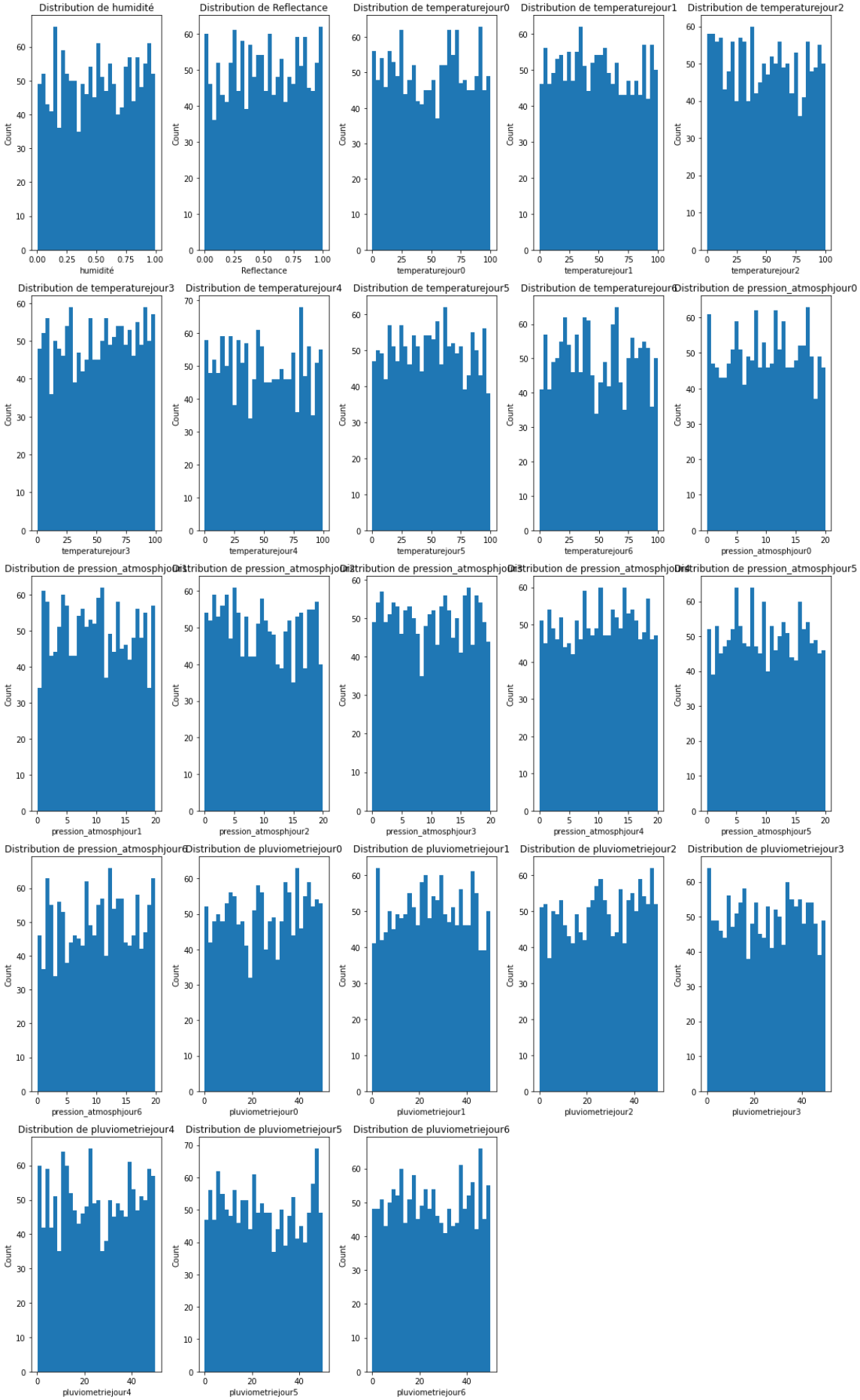
```
    'pression_atmosphjour2': [random.uniform(0, 20) for _ in
range(1500)],
    'pression_atmosphjour3': [random.uniform(0, 20) for _ in
range(1500)],
    'pression_atmosphjour4': [random.uniform(0, 20) for _ in
range(1500)],
    'pression_atmosphjour5': [random.uniform(0, 20) for _ in
range(1500)],
    'pression_atmosphjour6': [random.uniform(0, 20) for _ in
range(1500)],
    'pluviometriejour0': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour1': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour2': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour3': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour4': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour5': [random.uniform(0, 50) for _ in range(1500)],
    'pluviometriejour6': [random.uniform(0, 50) for _ in range(1500)],
}

# Créer un DataFrame Pandas avec les données d'attributs
attrib_df = pd.DataFrame(attrib_data)

# Convertir les colonnes de texte en Unicode
string_columns = attrib_df.select_dtypes(include=['object']).columns
attrib_df[string_columns] = attrib_df[string_columns].astype('U')

attrib_data['geometry'] = random_points
attrib_df = pd.DataFrame(attrib_data)

# Convertir en GeoDataFrame
gdf = gpd.GeoDataFrame(attrib_df, geometry='geometry', crs=crs)

# Écrire le GeoDataFrame dans un fichier GeoParquet
path_to_output_geoparquet = "OUTPUT1500.geoparquet"
gdf.to_parquet(path_to_output_geoparquet, engine='pyarrow')
# Créer une copie du GeoDataFrame avec les colonnes nécessaires pour le
shapefile
gdf_shapefile = gdf[['geometry', 'nom', 'humidité',
'Reflectance','temperaturejour0','temperaturejour1','temperaturejour2',
'temperaturejour3','temperaturejour4','temperaturejour5','temperaturejo
ur6',
'pression_atmosphjour0','pression_atmosphjour1','pression_atmosphjour2'
,'pression_atmosphjour3','pression_atmosphjour4','pression_atmosphjour5
','pression_atmosphjour6','pluviometriejour0','pluviometriejour1','pluv
iometriejour2','pluviometriejour3','pluviometriejour4','pluviometriejou
r5','pluviometriejour6']]

# Convertir les colonnes 'nom' et 'geometry' en chaînes de caractères
gdf_shapefile['nom'] = gdf_shapefile['nom'].astype(str)
```

```
# Écrire le GeoDataFrame dans un fichier shapefile
path_to_output_shapefile = "OUTPUT1500.shp"
gdf_shapefile.to_file(path_to_output_shapefile)
```

Code pour l'exploration des données :

```
import geopandas as gpd
import matplotlib.pyplot as plt

path_to_geoparquet =
"D:\Bureau\projethajji\donnees\geoparquet\OUTPUT1500.geoparquet"
gdf = gpd.read_parquet(path_to_geoparquet)

numerical_columns = [column for column in gdf.columns if
gdf[column].dtype in ['int64', 'float64']]
num_plots = len(numerical_columns)
columns = 5  # Nombre de colonnes pour les sous-graphiques
rows = -(-num_plots // columns)  # Calcul du nombre de lignes
nécessaires

fig, axs = plt.subplots(rows, columns, figsize=(15, 5 * rows))
axs = axs.flatten()

for i, column in enumerate(numerical_columns):
    axs[i].hist(gdf[column], bins=30)
    axs[i].set_title(f"Distribution de {column}")
    axs[i].set_xlabel(column)
    axs[i].set_ylabel('Count')

# Masquer les graphiques non utilisés s'il y a plus de sous-graphiques
que d'éléments
for j in range(num_plots, rows * columns):
    axs[j].axis('off')

plt.tight_layout()
plt.show()
```

resultat :


Distribution de humidité
Count
humidité
Distribution de Reflectance
Count
Reflectance
Distribution de temperaturejour0
Count
temperaturejour0
Distribution de temperaturejour1
Count
temperaturejour1
Distribution de temperaturejour2
Count
temperaturejour2
Distribution de temperaturejour3
Count
temperaturejour3
Distribution de temperaturejour4
Count
temperaturejour4
Distribution de temperaturejour5
Count
temperaturejour5
Distribution de temperaturejour6
Count
temperaturejour6
Distribution de pression_atmosphjour0
Count
pression_atmosphjour0
Distribution de pression_atmosphjour1
Count
pression_atmosphjour1
Distribution de pression_atmosphjour2
Count
pression_atmosphjour2
Distribution de pression_atmosphjour3
Count
pression_atmosphjour3
Distribution de pression_atmosphjour4
Count
pression_atmosphjour4
Distribution de pression_atmosphjour5
Count
pression_atmosphjour5
Distribution de pression_atmosphjour6
Count
pression_atmosphjour6
Distribution de pluviometriejour0
Count
pluviometriejour0
Distribution de pluviometriejour1
Count
pluviometriejour1
Distribution de pluviometriejour2
Count
pluviometriejour2
Distribution de pluviometriejour3
Count
pluviometriejour3
Distribution de pluviometriejour4
Count
pluviometriejour4
Distribution de pluviometriejour5
Count
pluviometriejour5
Distribution de pluviometriejour6
Count
pluviometriejour6

Code de conversion en COG :

```
import leafmap

folder = "RASTERS"  # Assurez-vous d'utiliser le bon chemin

# Boucle pour convertir chaque fichier tiff en COG
for attribute in ['temperature', 'pression_atmosph', 'pluviometrie']:
    for i in range(6):
        in_tiff = f"{folder}/{attribute.lower()}jour{i}.tif"  # Chemin
vers le fichier tiff d'entrée
        out_cog = f"{folder}/cog{attribute.lower()}jour{i}.tif"  #
Chemin de sortie pour le COG

        # Convertir le fichier tiff en COG
        arr = leafmap.image_to_numpy(in_tiff)
        leafmap.numpy_to_cog(arr, out_cog, profile=in_tiff)
```