Project Report

Extraction of Earth Observation Data from Google Earth Engine (GEE) and Storage in PostgreSQL

BY: ZAID AHAMED

Table of Contents

Objective

The objective of this project is to extract satellite imagery data for a Pakistan over the past year using Google Earth Engine (GEE), process the data to calculate NDVI (Normalized Difference Vegetation Index is a metric used to measure the health and density of vegetation) and store the processed data in a PostgreSQL database for further analysis.

Tools and Technologies Used

1. **Google Earth Engine (GEE):** For accessing and processing satellite imagery data.
2. **Python:** To write scripts for interacting with GEE and PostgreSQL.
3. **PostgreSQL:** To store the extracted data.
4. **Libraries:**
   - earthengine-api
   - psycopg2
   - pandas
   - datetime

Steps Involved

Step 1: Setup and Initialization

1. **Authenticate** and initialize the Google Earth Engine API in Python.
2. **Define the Area of Interest** (AOI) using geographical coordinates (Pakistan).
3. **Specify the time range** for data extraction (past one year).

Step 2: Data Extraction from GEE

1. **Dataset Selection**:
   - Used Sentinel-2 surface reflectance data (COPERNICUS/S2_HARMONIZED).
2. **Filtering Criteria:**
   - Temporal range: January 1, 2024, to December 20, 2024.
   - Spatial range: AOI defined for the specific region (Pakistan).
   - Cloud cover threshold: <20% cloudy pixels.
3. **Processing**:
   - Computed Normalized Difference Vegetation Index (NDVI) using GEE function:
     NDVI = (B8 - B4) / (B8 + B4)
   - Extracted date and NDVI values.

Step 3: Handling GEE Query Limitations

1. **Reduced the spatial resolution** to 1000 meters.
2. **Divided the time range into monthly chunks** to limit the number of images queried at a time.
3. **Filtered images by cloud cover** to reduce unnecessary data.

Step 4: Data Transformation and Cleaning

1. Used the Python Pandas library to covert data points into Dataframe.
2. Manually extracted the date from ID.
3. Dropped the rows that had null values in NDVI data.
4. Dropped the ID column.

Step 5: Data Export and Storage

1. Merged the Dataframes (for 1 month data).
2. Created a Database in PostgreSQL.
3. Created tabled for every month.
4. Inserted the merged data into a PostgreSQL database.
5. Created the table that include all the data using SQL query.

Step 6: Error Handling

1. Addressed issues such as missing dates in images.
2. Split queries into manageable chunks to avoid exceeding GEE's element limits.

Python Code Implementation

The Python script includes

- Authentication with GEE.
- AOI definition.
- Split the AOI into small lists.
- Generate Date ranges.
- Extract the monthly NDVI data for every splitted AOI.
- Data transformation and data cleaning.
- Merge data.
- Insert the data into Database.

Results

1. NDVI values and Dates were successfully extracted for the 1-year Pakistan.
2. The data was stored in a PostgreSQL database for further analysis.
3. The approach handled GEE limitations effectively, ensuring smooth execution.

Challenges and Solutions

Challenge 1: GEE Query Limitations

Reason: Large AOI, fine resolution, and a large number of images can't extracted once.

Solution:

- Splitted the AOI into small chunks.
- Reduced resolution (1km).
- Divided queries into monthly chunks.
- Filtered by cloud cover.

Challenge 2: Missing Dates Property

Reason: Some Images lacked timestamp metadata.

Solution: Applied manual date extraction using GEE functions.

Conclusion

This project demonstrates the feasibility of extracting and processing Earth observation NDVI data from GEE and storing it in PostgreSQL. By overcoming challenges such as query limitations, the workflow ensures efficient data extraction for further analysis and decision-making.

Future Work

1. Automate the entire process.
2. Visualize the data using tools.
3. Implement machine learning models for predictive analysis.

References

1. Google Earth Engine Documentation: https://developers.google.com/earth-engine
2. Sentinel-2 Data Overview: https://sentinels.copernicus.eu
3. PostgreSQL Documentation: https://www.postgresql.org