Abstract

These project focuses on harnessing synthetic aperture radar (SAR) data from the Copernicus Sentinel-1 mission for environmental monitoring and analysis. Leveraging the Google Earth Engine platform, the project aims to acquire, preprocess, and analyze SAR imagery to detect changes in land cover, monitor vegetation dynamics, and assess environmental trends. Machine learning algorithms, including random forests and support vector machines, will be utilized to develop predictive models for anticipating environmental changes based on historical SAR data. Interactive visualizations generated with tools like Matplotlib and Geemap will facilitate the interpretation of SAR imagery and analytical results.

The project seeks to provide decision-makers with actionable insights to support resource management, disaster preparedness, and climate resilience initiatives. Additionally, efforts will be made to disseminate knowledge, build capacity, and foster collaboration to promote the adoption of SAR data analysis techniques in environmental science and policy-making

Keywords: Geemap, SAR, Google Earth Engine, Time Series.

CONTENTS

S. No.	Chapters	Page No.
1	Project Description	1
	1.1 Introduction	2
	1.2 Problem Statement	3
	1.3 Problem objective	3
	1.4 Hardware Specification	4
	1.5 Software Specification	4 - 5
2	Review of Literature	6
	2.1 Existing system	7
	2.2 Proposed system	8
3	Logical Development	12
	3.1 Architectural Design	13
	3.2 DFDs	15
4	Dataset Details	19
	4.1 Data Collection	20
	4.2 Description of the Data	22
	4.3 Source and Methods of Collecting Data	25
5	Program Design	27
	5.1 Logic	28
	5.2 Working of the Programs	29
	5.3 Modules in detail	35
6	Testing and Validation	38
	6.1 Testing methods	39
	6.2 Validation justifications	40
7	Conclusion	41
8	Uses cases	44
9	References	47
10	Appendix	49
	8.1. Source Code	50
	8.2. O/P Screens	56

CHAPTER 1
PROJECT DESCRIPTION

1.1 Introduction

This project focuses on analyzing Synthetic Aperture Radar (SAR) data from the Copernicus Sentinel-1 mission, utilizing Google Earth Engine for efficient data retrieval. SAR technology captures high-resolution images of the Earth's surface using radar waves, operating reliably across diverse weather conditions and terrains. The data obtained holds significant relevance across various domains, facilitating environmental monitoring, land management, and disaster response efforts. This project serves as a representative sample for studying environmental dynamics on a broader scale. Through SAR data analysis, insights into urbanization patterns, changes in land cover, and environmental trends are derived. This analysis entails stages such as temporal trend and spatial pattern analysis, facilitated by steps including data retrieval, pre-processing, analysis, and visualization of SAR imagery. Ultimately, the project aims to underscore the utility of SAR information in addressing environmental challenges, thereby enabling informed decision-making and promoting sustainable practices. In remote sensing and radar imaging, VH typically refers to the polarization of radar waves. Radar waves can be polarized in different ways, and VH specifically stands for Vertical-Horizontal polarization. Vertical polarization (V): In this polarization, the electric field of the radar wave is oriented vertically. Horizontal polarization (H): In this polarization, the electric field of the radar wave is oriented horizontally. When radar waves are transmitted and received in VH polarization, it means that the radar system transmits waves with vertical polarization and receives waves with horizontal polarization. This polarization combination is commonly used in radar imaging for various applications such as terrain mapping, vegetation monitoring, and surface deformation analysis, among others. The VH backscatter values represent the strength of the radar signal returned to the sensor in this specific polarization configuration.

1.2 Problem Statement

Leveraging Earth Observation Data for Environmental Monitoring and Analysis

In this project, we aim to harness the power of synthetic aperture radar (SAR) data obtained from satellite missions, particularly the Copernicus Sentinel-1 mission, to enhance environmental monitoring and analysis. SAR data provides valuable insights into various environmental phenomena, including changes in land cover, vegetation dynamics, urban development, and natural hazards. By leveraging advanced data processing techniques and machine learning algorithms, we seek to extract meaningful information from SAR imagery to support informed decision-making processes in environmental management, disaster response, and land use planning.

1.3 Problem Objective

- Acquiring and preprocessing SAR imagery from the Copernicus Sentinel-1 mission via the Google Earth Engine platform.
- Extracting key features and patterns from SAR data to detect changes in land cover, monitor vegetation dynamics, and assess environmental trends.
- Developing predictive models using machine learning algorithms such as random forests and support vector machines to anticipate environmental changes based on historical SAR data.
- Creating interactive visualizations with tools like Matplotlib and Geemap to facilitate the interpretation of SAR imagery and analytical results.
- Providing decision-makers with actionable insights to support resource management, disaster preparedness, and climate resilience initiatives.
- Emphasizing knowledge dissemination, capacity building, and collaboration to promote the adoption of SAR data analysis techniques in environmental science and policy-making efforts.

1.4 Hardware Specification

The proposed system operates on standard hardware configurations, necessitating a computer with adequate processing power, memory, and storage capacity to handle SAR data processing tasks efficiently. This hardware requirement ensures the smooth execution of data-intensive operations involved in SAR data analysis.

1. **RAM:** A minimum of 8GB of RAM is recommended to ensure the smooth processing of large datasets and complex algorithms.
2. **Processor:** An Intel Core i5 or equivalent processor is advised to handle the computational workload efficiently.
3. **Storage:** The system should have sufficient storage capacity to store both raw SAR data and processed outputs, with a minimum of 200GB of disk space recommended.

These hardware specifications provide a baseline for optimal performance and reliability in SAR data analysis tasks, enabling researchers, analysts, and decision-makers to conduct thorough and efficient analyses of geospatial data.

1.5 Imported Packages and Software Specifications

Python: Python serves as the primary programming language for scripting and data analysis within the proposed system. Renowned for its simplicity and versatility, Python offers a wide range of libraries and frameworks that facilitate various aspects of SAR data analysis.

Earth Engine: Earth Engine stands out as a cloud-based platform designed for geospatial data analysis. Leveraging Google's vast computing infrastructure, Earth Engine provides access to a rich repository of geospatial datasets and advanced processing capabilities, enabling seamless integration of SAR data into analytical workflows. Utilizing Earth Engine data ensures access to up-to-date and accurate geospatial information, eliminating the need for manual data collection and ensuring data consistency and reliability.

Google Colab: Google Colab, short for Colaboratory, is a cloud-based platform provided by Google for running Python code in Jupyter notebooks. It offers a convenient and accessible environment for collaborative data analysis, research, and machine learning tasks. With Colab, users can write and execute Python code directly in the browser without requiring any setup or installation.
One of the key advantages of Colab is its integration with Google Drive, allowing users to store and share Jupyter notebooks seamlessly. Additionally,
Colab provides access to free GPU and TPU (Tensor Processing Unit) resources, enabling faster computation for resource-intensive tasks such as deep
learning and large-scale data processing. This makes Colab an ideal platform for SAR data analysis projects, offering the computational power and collaborative features necessary for efficient analysis workflows.

Google Cloud T4 GPU: Google Cloud T4 GPU instances are virtual machines available on the Google Cloud Platform (GCP) that are equipped with NVIDIA T4 GPUs. These GPUs are specifically optimized for accelerating machine learning workloads and are well-suited for tasks such as model training, inference, and data processing. By leveraging T4 GPU instances on Google Cloud, users can significantly reduce the time required for SAR data analysis tasks that involve computationally intensive operations, such as training machine learning models or processing large datasets. The parallel processing capabilities of T4 GPUs enable faster computation, leading to improved productivity and efficiency in SAR data analysis workflows.

CHAPTER 2
REVIEW OF LITERATURE

\

2.1 Existing System

In the existing system, traditional SAR data analysis typically demands specialized software and expertise in remote sensing techniques. Users encounter hurdles in acquiring, processing, and analyzing data owing to the intricate nature of SAR data and the absence of user-friendly tools. Unlike Python, which has limited availability for obtaining SAR images, most resources are predominantly in JavaScript. This disparity in language support poses challenges for Python users, as they often need to resort to JavaScript-based solutions or workarounds to access SAR data effectively. Consequently, this reliance on specialized software and JavaScript-centric resources can impede the accessibility and usability of SAR data for Python users, highlighting a gap in the existing system's adaptability to diverse programming environments.

Furthermore, the scarcity of Python-based libraries and tools for SAR data acquisition and processing limits the flexibility and scalability of analysis workflows. This dependence on JavaScript-centric solutions restricts the interoperability of SAR data analysis with other Python-based tools and libraries, hindering seamless integration into existing data analysis pipelines. Additionally, the complexity of SAR data and the lack of comprehensive documentation further exacerbate the challenges faced by users, necessitating extensive training and expertise to navigate through the data acquisition and processing stages effectively. Overall, the existing system underscores the need for more accessible and user-friendly tools for SAR data analysis, particularly in Python, to enhance the efficiency and usability of remote sensing applications.

2.2 Proposed System

The proposed system marks a notable leap forward in SAR (Synthetic Aperture Radar) data analysis by harmonizing Python libraries, Earth Engine, and machine learning methodologies. This amalgamation brings forth a holistic framework that streamlines the entire SAR data analysis process, encompassing data retrieval, preprocessing, visualization, and analytical stages. By integrating these components seamlessly, the system aims to democratize access to SAR data, empowering researchers, analysts, and decision-makers with enhanced capabilities for extracting insights from SAR imagery.

1. Python Libraries Integration:

- Python serves as the foundation of the proposed system, providing a versatile environment for SAR data analysis.
- Python offers a diverse ecosystem of libraries specialized in data analysis and visualization tasks.
- Matplotlib is employed for generating high-quality visualizations, facilitating the representation of SAR data effectively.
- Pandas is utilized for efficient data manipulation, enabling users to organize and process SAR datasets seamlessly.
- NumPy plays a crucial role in numerical computing, supporting complex operations on SAR data arrays efficiently.
- Together, these libraries create a robust framework for SAR data analysis, empowering users to manipulate, analyze, and visualize SAR imagery with ease.

2. Engine Integration:

- Earth Engine is a cloud-based platform developed by Google for geospatial data analysis.
- It provides access to a vast repository of geospatial datasets, including SAR imagery.

- Users can tap into Earth Engine's powerful processing capabilities to analyze SAR data efficiently.
- Earth Engine facilitates seamless data retrieval and preprocessing, streamlining the preparation of SAR imagery for analysis.
- Its scalability and cloud-based infrastructure enable users to handle large-scale SAR datasets effectively.
- Earth Engine empowers users to tackle complex analytical tasks with ease, thanks to its robust features and capabilities.

2.2.1 Why Google Earth?

Google Earth is a three-dimensional software model of the earth. It is a commonly used tool to explore the geography of the world and displays satellite images of varying resolution of the earth's surface. By using Google Earth, one is able to:

- Search for cities, areas, etc., by name or coordinates
- Add additional information (e.g., photos) to a map
- Track the development of an area over time
- Add geographic information (e.g., GPS, GIS)
- Measure distances and areas
- Export and share information integrated in a map

2.2.2 These features make Google Earth a useful tool for sanitation planning, as one can:

- Gain an overview of a location and spot possible difficulties
- Assess accessibility and display existing infrastructure
- Check the expansion of settlements over time
- Assess spatial conditions (e.g., available space, slope, etc.)
- Create a first draft of a sanitation system
- Share sanitation plan with other project members

2.2.3 Limitations of Google Earth

Keep in mind that the accuracy of the data depends on many factors, such as spatial resolution, georeferencing, etc. Thus, the data quality and quantity can vary from region to region. Between the measured data points interpolation is used to fill in the information gaps. The data displayed in Google Earth will never be 100% accurate and one should be aware of this limitation while using it. Therefore, one has to evaluate, if the provided data accuracy is sufficient for the proposed application for each case.

2.2.4 General Information

This guide does not cover every single feature of Google Earth or all the possible ways of doing something with it. It also may not explain all topics as thoroughly as some readers might find necessary.
However, a web link leading to more information on particular subjects is often supplied. All the referred web links were up-to-date as of July 2015.

Comparison to the basic version, the Pro version

Features	Google Earth	Google Earth Pro
Import GIS data	1000 pixels	4800 pixels
Import GIS data	-	ESRI .shp, MapInfo .tab
Import addresses in bulk	Manually Geo-locate each address	Automatically Geo-locate up to 2500 at a time
Import large image files	limited to texture size	Super Image Overlays
Supplemental Layers	-	Demographics, Parcels, Traffic Counts
Create premium movies for export	-	HD 1920x1080
Measurement tools	Line, Path	Line, Path, Polygon, Circle, 3D Path, 3D Polygon

3. Machine Learning Techniques:

- Machine learning techniques, integrated into the proposed system, enhance its analytical capabilities for SAR imagery.
- Libraries like Scikit-learn provide algorithms for classification, regression, and clustering analysis on SAR data.
- These algorithms enable automated feature extraction, pattern recognition, and anomaly detection from SAR imagery.
- Users can leverage machine learning to uncover hidden patterns and trends within SAR data.
- Predictive modeling facilitated by machine learning allows users to forecast environmental changes and anticipate future trends using historical SAR data.

By combining Python libraries, Earth Engine, and machine learning techniques, the proposed system offers a comprehensive and efficient workflow for SAR data analysis. This integration enhances accessibility to SAR data, enabling researchers, analysts, and decision-makers to leverage SAR imagery for a wide range of applications, including environmental monitoring, land management, and disaster response. Overall, the proposed system represents a significant advancement in SAR data analysis, empowering users with enhanced capabilities for extracting insights from SAR imagery and informing evidence-based decision-making processes.

CHAPTER 3
LOGICAL DEVELOPMNT

3.1 Architectural Design

The SAR data analysis project involves outlining the various components and their interactions. Below is a simplified architectural diagram illustrating the key components and their relationships:

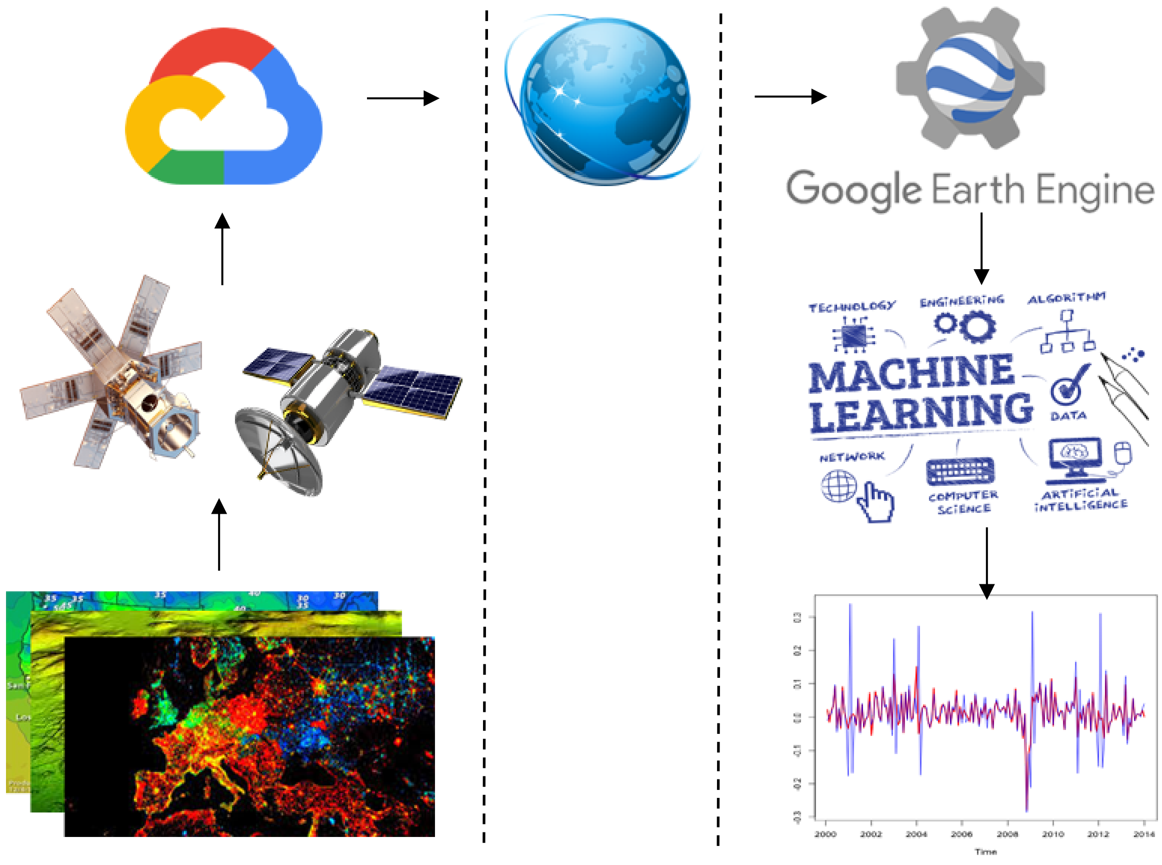


Figure 3.1

Architectural work flow:

1. Earth surface
2. Satellites and sensors
3. Cloud storage
4. Internet
5. Google Earth Engine
6. Machine Learning and Deep Learning Algorithms
7. Predicted Times series data

The architectural diagram illustrates a sophisticated ecosystem designed to harness the power of Synthetic Aperture Radar (SAR) technology for comprehensive Earth surface analysis. Let's explore each component in detail and delve into their functionalities and applications:

1. Earth Surface: At the core of the system lies the Earth's surface, a vast and dynamic landscape rich with valuable information. SAR technology enables the capture of high-resolution radar images of the Earth's surface, offering insights into terrain features, land cover, vegetation, and changes over time. This data serves as the raw material for various analyses aimed at understanding environmental dynamics, land use patterns, and natural phenomena.

2. Satellites: Orbiting satellites equipped with SAR sensors play a pivotal role in data acquisition. These satellites, such as those from the Copernicus Sentinel-1 mission, continuously collect SAR imagery over large geographical areas with regular revisit times. By capturing radar signals reflected from the Earth's surface, satellites provide a consistent stream of data that enables monitoring of global and regional environmental changes, including deforestation, urbanization, and natural disasters.

3. Google Cloud: The Google Cloud platform serves as the backbone infrastructure for storing, processing, and analyzing SAR data. Leveraging cloud computing resources, including storage, computing power, and data processing tools, Google Cloud enables scalable and efficient data management and analysis. By hosting SAR data on the cloud, users can access and process large volumes of data remotely, without the need for extensive local computing resources.

4. Internet: The internet acts as the conduit through which SAR data is transmitted from satellites to the Google Cloud platform. High-speed internet connectivity ensures seamless data transfer, enabling real-time or near-real-time access to SAR imagery for analysis. The internet also facilitates data sharing and collaboration, allowing researchers and stakeholders worldwide to access and exchange SAR data and analytical results.

5. Google Earth Engine: Google Earth Engine represents a powerful platform for geospatial data analysis, offering a vast repository of satellite imagery and advanced processing capabilities. SAR data from missions like Sentinel-1 can be ingested into Earth Engine, where users can apply a wide range of analysis algorithms and techniques to extract valuable insights. Earth Engine's scalability and parallel processing capabilities enable efficient analysis of SAR data over large spatial and temporal scales, facilitating tasks such as land cover classification, change detection, and trend analysis.

6. ML Algorithms: Machine learning (ML) algorithms are instrumental in SAR data analysis, enabling automated feature extraction, pattern recognition, and predictive modeling. By training ML models on labeled SAR data, researchers can develop algorithms capable of identifying specific features or phenomena in SAR imagery, such as land cover types, deforestation areas, or flood extents. ML techniques also enable the generation of predictive models that forecast future environmental changes based on historical SAR data, aiding in risk assessment and decision-making.

7. Time Series Data: Time series data derived from SAR imagery captures temporal variations in Earth surface characteristics over time. By analyzing SAR data collected at multiple time points, researchers can track changes in land cover, vegetation health, and other environmental variables, enabling long-term trend analysis and monitoring. Time series analysis techniques, such as trend analysis, anomaly detection, and seasonal decomposition, provide valuable insights into the dynamics of Earth surface processes and inform environmental management strategies.

3.2 Data Flow Diagrams

Below is a detailed Data Flow Diagram (DFD) for the SAR data analysis project, illustrating the flow of data and processes involved in the system:

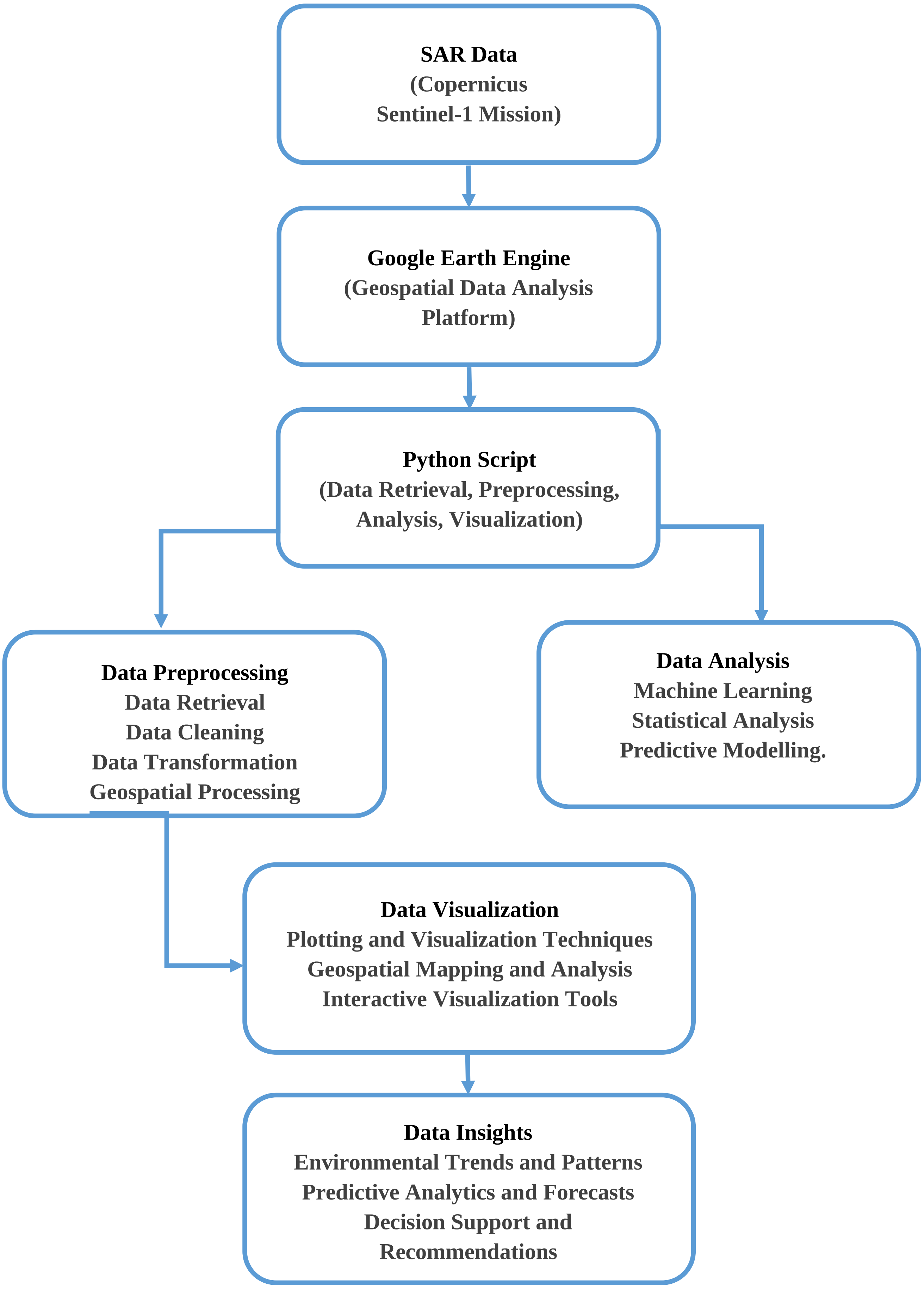


Figure 3.2

SAR Data (Copernicus Sentinel-1 Mission): The project begins with the collection of SAR data from the Copernicus Sentinel-1 mission. This data contains valuable information about the Earth's surface and is essential for conducting various analyses.

Google Earth Engine: The SAR data is then processed and analyzed using Google Earth Engine, a powerful geospatial data analysis platform. Earth Engine provides access to a vast collection of geospatial datasets and tools for performing complex analyses at scale.

Python Script: A Python script is developed to interact with Earth Engine, retrieve SAR data, preprocess it, conduct analysis, and visualize the results. Python's versatility and rich ecosystem of libraries make it an ideal choice for such tasks.

Data Preprocessing: This submodule within the Python script encompasses tasks such as data retrieval, cleaning, transformation, and geospatial processing. These tasks prepare the SAR data for further analysis and visualization.

Data Analysis: The Data Analysis submodule includes machine learning algorithms, statistical analysis techniques, and predictive modeling methods. These techniques are applied to the preprocessed SAR data to extract insights, identify trends, and make predictions.

Data Visualization: The results of the data analysis are visualized using various plotting and visualization techniques. This submodule generates visual representations of the SAR data, including plots, maps, and interactive visualizations.

Machine Learning: Machine learning techniques, implemented using libraries such as Scikit-learn and TensorFlow, are employed for advanced analysis tasks.

These techniques allow for the identification of patterns, trends, and predictions within the SAR data.

Data Insights: The final component involves deriving insights from the analyzed SAR data. This may include identifying environmental trends, making predictions, and generating actionable insights for stakeholders.

This architectural diagram provides a high-level overview of the SAR data analysis project, showcasing the main components and their interactions in the analysis workflow.

In summary, the SAR data collected for the project is sourced from the Copernicus Sentinel-1 mission through the Earth Engine platform, enabling access to high-quality radar imagery for comprehensive analysis and interpretation. The combination of Sentinel-1's advanced SAR sensors and Earth Engine's robust processing capabilities facilitates a wide range of applications, from environmental monitoring to disaster response, and empowers users to make informed decisions based on timely and accurate geospatial information.

CHAPTER 4
DATASET DETAILS

4.1 Data Collection

The process of collecting Synthetic Aperture Radar (SAR) data for the SAR data analysis project is a crucial step that lays the foundation for subsequent analysis and interpretation. This data collection phase involves a series of steps and considerations to ensure the acquisition of relevant and high-quality SAR imagery from the Copernicus Sentinel-1 mission via the Earth Engine platform.

To begin with, the SAR data collection process begins with defining the scope of the analysis, including the geographical areas of interest and the time periods under investigation. These parameters help narrow down the search criteria when querying the Earth Engine repository for SAR imagery. Geographical coordinates specify the spatial extent of the analysis area, while acquisition dates determine the temporal coverage of the SAR data. Additionally, considering sensor characteristics such as polarization modes and spatial resolutions helps refine the search criteria to retrieve SAR images that best suit the project's objectives.

Once the search parameters are defined, the next step involves querying the Earth Engine repository using the Earth Engine API or the Earth Engine Code Editor interface. This querying process involves constructing queries using Earth Engine's proprietary programming language (JavaScript or Python) to specify the desired criteria for SAR imagery retrieval. These queries may include spatial and temporal filters, as well as additional constraints based on sensor specifications and data quality metrics.

Upon executing the query, the Earth Engine platform retrieves SAR images that match the specified criteria and returns them in a structured format suitable for further analysis. The retrieved SAR imagery typically consists of raster datasets encoded in a standard format such as GeoTIFF, containing radar backscatter values and associated metadata such as acquisition timestamps and sensor parameters.

Once the relevant SAR images are identified and retrieved, they undergo preprocessing steps to prepare them for analysis within the project's framework. These preprocessing steps may include radiometric calibration, geometric correction, speckle filtering, and terrain correction to enhance the quality and usability of the SAR data. Additionally, metadata extraction and annotation are performed to capture essential information about the SAR imagery, such as orbit information, polarization modes, and spatial resolutions.

Finally, the preprocessed SAR images are integrated into the project's analytical pipeline for further analysis and interpretation. This may involve applying various algorithms and techniques to extract meaningful information from the SAR data, such as land cover classification, change detection, and environmental monitoring. Additionally, visualization tools and techniques are employed to visualize the SAR imagery and derived products, facilitating data exploration and interpretation by stakeholders and decision-makers.

In summary, the process of collecting SAR data for the SAR data analysis project is a systematic and iterative process that involves defining search parameters, querying the Earth Engine repository, retrieving relevant SAR imagery, preprocessing the data, and integrating it into the analytical pipeline. This data collection phase lays the groundwork for conducting in-depth analysis and interpretation of SAR imagery to address various environmental challenges and support informed decision-making processes.

4.2 Description of the Data

The SAR data obtained from the Copernicus Sentinel-1 mission constitutes a rich repository of radar imagery captured by SAR sensors mounted on satellites orbiting Earth. Each SAR image encapsulates a wealth of information pertaining to the Earth's surface characteristics, offering a comprehensive view of terrain features, land cover variations, and prevailing environmental conditions. Notably, the Vertical-Horizontal (VH) polarization mode serves as a focal point for analysis within this project, providing a unique perspective on surface properties and scattering mechanisms. The VH polarization mode is particularly valuable due to its sensitivity to different surface structures and materials, making it well-suited for discerning subtle changes in land cover and detecting environmental phenomena such as deforestation, urbanization, and soil moisture content.

Furthermore, the SAR data encompasses a wide spectrum of spatial resolutions and temporal intervals, catering to the diverse needs of environmental monitoring and analysis. High-resolution SAR imagery enables detailed exploration of local-scale features and phenomena, while lower-resolution data offers broader coverage for regional and global-scale assessments. This variability in spatial resolution allows for multi-scale analysis, facilitating the examination of environmental dynamics across different geographical extents and levels of detail.

In addition to its spatial and polarization characteristics, the SAR data captures temporal variations in environmental conditions over time. By acquiring SAR imagery at regular intervals, the dataset enables the study of temporal changes in land cover, vegetation dynamics, and natural processes. This temporal dimension is crucial for monitoring seasonal variations, assessing long-term trends, and detecting abrupt environmental changes such as natural disasters and land cover disturbances.

The dataset encompasses a diverse array of landscapes, ranging from densely populated urban areas to remote natural habitats, reflecting the heterogeneity of Earth's surface. Urban areas exhibit distinctive radar signatures due to the presence of buildings, infrastructure, and impervious surfaces, while agricultural regions showcase varying patterns of crop cultivation and land management practices. Water bodies such as rivers, lakes, and reservoirs exhibit unique radar responses influenced by surface roughness, water content, and hydrological dynamics. Natural habitats including forests, wetlands, and grasslands exhibit distinct radar signatures associated with vegetation structure, biomass distribution, and ecological health.

Overall, the SAR dataset obtained from the Copernicus Sentinel-1 mission offers a comprehensive and multi-dimensional view of Earth's surface dynamics, encompassing spatial, polarization, and temporal dimensions. Leveraging this rich dataset enables researchers, analysts, and decision-makers to gain valuable insights into environmental processes, monitor changes over time, and inform evidence-based decision-making for sustainable resource management and environmental conservation efforts.

3.3 Source and Methods of Collecting Data:

1. The primary source of SAR data for the project is the Copernicus Sentinel-1 mission, a cornerstone of the European Union's Copernicus program for Earth observation. Sentinel-1 satellites are equipped with state-of-the-art SAR sensors that continuously capture radar imagery over land and sea surfaces with remarkable spatial resolution and temporal frequency. These satellites operate in different polarizations, including Vertical-Horizontal (VH), Vertical-Vertical (VV), and Horizontal-Horizontal (HH), offering a versatile suite of observations for diverse applications.

2. The SAR data collected by the Sentinel-1 mission is made freely available to the public through the Earth Engine platform, a powerful cloud-based geospatial analysis tool developed by Google. Earth Engine serves as a centralized repository for a vast array of geospatial datasets, including SAR imagery, optical imagery, and climate data. Leveraging Google's extensive computing infrastructure, Earth Engine enables seamless access to SAR data and facilitates scalable processing and analysis workflows.

3. The process of collecting SAR data for the project begins with querying Earth Engine's extensive archive of Sentinel-1 imagery using specific criteria such as geographical regions of interest, time periods, and sensor parameters. This querying process is facilitated through Earth Engine's Application Programming Interface (API), which allows users to programmatically retrieve SAR images based on predefined criteria. Users can construct complex queries using Earth Engine's proprietary programming languages, JavaScript or Python, to filter and retrieve the most relevant SAR data for their analysis.

4. Once the relevant SAR data is retrieved from Earth Engine, it undergoes preprocessing steps to prepare it for further analysis. These preprocessing steps may include radiometric calibration, geometric correction, speckle filtering, and terrain correction to enhance the quality and usability of the SAR data. Additionally, metadata extraction and annotation are performed to capture essential information about the SAR imagery, such as acquisition timestamps, orbit information, and sensor parameters.

5. After preprocessing, the SAR data is ready for analysis within the project's analytical framework. Python scripts and libraries are utilized to process and analyze the SAR imagery, leveraging Earth Engine's processing capabilities and machine learning algorithms for advanced

analysis and interpretation. Various analytical techniques, including change detection, classification, and time-series analysis, are applied to derive meaningful insights from the SAR data and address specific research questions or objectives.

4.3 Data Preprocessing

The preprocessing of SAR data involves several essential steps to prepare the raw imagery for analysis and interpretation. Initially, the data is acquired from the Copernicus Sentinel-1 mission via the Google Earth Engine platform, ensuring access to a vast repository of SAR imagery spanning various geographical regions and time periods. Once obtained, the data undergoes geometric and radiometric calibration to correct for distortions caused by sensor geometry, atmospheric conditions, and terrain variations. Geometric calibration involves rectifying the imagery to a common map projection and removing geometric distortions induced by satellite motion and Earth's curvature. Radiometric calibration aims to standardize the pixel values across images, compensating for differences in sensor sensitivity, noise characteristics, and signal attenuation.

Following calibration, the preprocessed SAR imagery is subjected to speckle filtering to reduce the impact of speckle noise, which appears as random variations in pixel intensity and can obscure meaningful features in the data. Speckle filtering techniques, such as Lee, Frost, or Gamma-MAP filtering, are applied to enhance image quality while preserving relevant information about surface properties and features.

Moreover, terrain correction is often performed to account for variations in terrain elevation and slope, ensuring accurate georeferencing and alignment of SAR images. This correction involves applying a digital elevation model (DEM) to remove the effects of terrain-induced geometric distortions, such as layover and foreshortening, which can affect the interpretation and analysis of SAR data, particularly in mountainous or rugged terrain.

Additionally, data fusion techniques may be employed to integrate SAR imagery with other remote sensing datasets, such as optical imagery or terrain information, to complement and enhance the analysis capabilities. Fusion methods, such as data assimilation or multi-sensor integration, facilitate a more comprehensive understanding of environmental processes and dynamics by leveraging complementary information from different sensor modalities.

Overall, the preprocessing of SAR data plays a crucial role in ensuring data quality, accuracy, and consistency, thereby enabling meaningful analysis and interpretation of SAR imagery for various applications in environmental monitoring, land management, disaster assessment, and scientific research.

CHAPTER 5
PROGRAM DESIGN

5.1 Logic

- Utilize Synthetic Aperture Radar (SAR) imagery from Copernicus Sentinel-1 via Google Earth Engine (GEE) and Python libraries.
- Understand the significance of VH polarization in radar imaging for capturing terrain features, vegetation, and surface deformation.
- Access GEE API within Python for seamless interaction with geospatial datasets and analytical tools.
- Authenticate and initialize the Earth Engine Python API to query and retrieve SAR imagery tailored to specific regions and timeframes.
- Extract relevant SAR data using filtering and sorting operations to align with study objectives and spatial-temporal criteria.
- Employ a time-efficient approach to extract time series data on VH backscatter values within defined ROIs, utilizing Python's data manipulation and visualization capabilities.
- Uncover temporal relationships and patterns in VH backscatter data to gain insights into dynamic environmental changes over time.
- Develop predictive models using machine learning techniques like Random Forest regression to forecast VH backscatter values based on lagged features of the original data.
- Evaluate model performance using metrics such as mean squared error (MSE) to guide adjustments and optimizations for improved accuracy.
- Integrate expertise in radar imaging, geospatial data processing, and machine learning to extract actionable insights from SAR data for informed decision-making in environmental monitoring and disaster response.

5.2 Working of the Program

Step 1: Understanding VH Polarization in Radar Imaging

In remote sensing and radar imaging, polarization refers to the orientation of electromagnetic waves as they propagate through space. Radar waves can be polarized in different ways, with VH polarization specifically indicating Vertical-Horizontal polarization. In this polarization:

- Vertical polarization (V) refers to radar waves where the electric field is oriented vertically.
- Horizontal polarization (H) refers to radar waves where the electric field is oriented horizontally.

When radar waves are transmitted and received in VH polarization, it means that the radar system transmits waves with vertical polarization and receives waves with horizontal polarization. This polarization combination is commonly used in radar imaging for various applications such as terrain mapping, vegetation monitoring, and surface deformation analysis.

The VH backscatter values represent the strength of the radar signal returned to the sensor in this specific polarization configuration. Higher backscatter values indicate stronger radar returns, which can be influenced by factors such as surface roughness, composition, and moisture content. Understanding VH polarization is crucial for interpreting radar imagery and extracting meaningful information about the Earth's surface characteristics.

Step 2: Accessing the Google Earth Engine API:

The provided script sets up the environment for accessing and utilizing Google Earth Engine (GEE) within a Python environment. It imports essential libraries such as Matplotlib for plotting, Pandas for data manipulation, NumPy for numerical computations, geemap for interactive mapping with GEE, and the

Earth Engine Python API (ee) for interfacing with GEE's geospatial datasets and processing capabilities.

After importing the libraries, the script triggers the authentication process using `ee.Authenticate()`, which prompts the user to authenticate their Google account to access GEE services. Once authenticated, the script initializes the Earth Engine Python API using `ee.Initialize()`, enabling the user to access and analyze remote sensing data, perform geospatial analysis, and create interactive maps using GEE's extensive datasets and tools.

Step 3: Extracting Data for a Given Region of Interest (ROI)

The script defines an area of interest (AOI) near Berlin, Germany, as a rectangular geometry. It sets the start and end dates for filtering the Sentinel-1 GRD (Ground Range Detected) image collection, focusing on images captured between 2014 and 2021. The image collection is filtered to include only images acquired in Interferometric Wide (IW) mode with VH polarization and a resolution of 10 meters within the specified AOI and time range. Additionally, the collection is filtered to include images captured during ascending orbits and sorted by date in ascending order. The script also creates an interactive map using the geemap library to visualize the first image in the sorted collection and overlays the AOI's bounding box, providing a visual representation of the selected region and available satellite imagery for analysis.

Step 4: Time Series Data Extraction: Time-Efficient Approach

This script processes the Sentinel-1 GRD image collection by calculating the mean VH backscatter values for each image within the defined AOI. It defines a function named `datedist` that retrieves the acquisition date and calculates the mean backscatter value for each image. The function is then mapped to the sorted image collection using the map function.

After mapping the function, the script aggregates the acquisition dates and corresponding backscatter values from the modified image collection using the `aggregate_array` function. These properties are converted into lists, and the acquisition dates are converted into Python datetime objects. A DataFrame is created to store the date-backscatter data, and the VH backscatter values are plotted over time using Matplotlib. Additionally, the script prints information about the number of processed images, the starting and ending times of processing, and the elapsed time between the two.

This script enables the analysis and visualization of VH backscatter changes over time, providing insights into land surface dynamics and environmental changes within the specified AOI.

Step 5: Finding Lagged Values

The script plots both the original VH backscatter values and their corresponding lagged values over time. It creates a figure with the original VH backscatter values plotted using plt.plot and iterates over lagged time steps, plotting each lagged value on the same plot with dashed lines. Each lagged value is labeled with its corresponding lag number. The plot title, axis labels, legend, and grid lines are configured for better readability. This plot allows for visualizing the relationship between the original VH backscatter values and their lagged counterparts, providing insights into any temporal patterns or trends present in the data.

Step 6: Creating a Lagged Model

This script utilizes a Random Forest Regressor model to predict VH backscatter values based on lagged features of the original VH backscatter data. It defines the number of lagged time steps (lags) to create lagged features for modeling and shifts the original VH backscatter values to create lagged features using a for loop. Rows with NaN values resulting from shifting are then dropped from the DataFrame to ensure that the dataset is clean and ready for modeling. The data is split into training and testing sets, and a Random Forest Regressor model is initialized and trained on the training data. The trained model is used to predict VH backscatter values on the test set, and the mean squared error (MSE) between the predicted and actual VH backscatter values is calculated to evaluate the model's performance. A lower MSE indicates better predictive performance of the model on the test data.

Step 7: Model Evaluation

The mean squared error (MSE) value obtained from the evaluation of the Random Forest Regressor model on the test set is approximately 0.562. This metric represents the average squared difference between the predicted VH backscatter values and the actual VH backscatter values in the test set. A lower MSE indicates better agreement between the predicted and actual values, suggesting that the model's predictions are closer to the true values. In this case, the obtained MSE value indicates that the model performs reasonably well in predicting VH backscatter values based on the lagged features of the original VH backscatter data.

Expanding on Data Collection Techniques:

In addition to leveraging the Copernicus Sentinel-1 mission and Google Earth Engine for SAR data acquisition, supplementary data collection techniques can enhance the comprehensiveness and accuracy of the dataset. Ground-based measurements, field surveys, and aerial observations can provide complementary information to satellite-derived SAR imagery, offering ground truth data for validation and calibration purposes. For example, field measurements of vegetation characteristics, soil moisture levels, and land cover types can be used to validate SAR-derived parameters and improve the accuracy of environmental assessments. Similarly, aerial surveys using drones or manned aircraft can capture high-resolution imagery and LiDAR data, enabling detailed mapping of terrain features, vegetation structure, and land use patterns.

Integrating multi-source data from different platforms and sensors facilitate synergistic analysis and enhances the reliability and robustness of SAR data products. Moreover, crowd-sourced data collection initiatives and citizen science projects can engage local communities in environmental monitoring efforts, fostering collaboration and knowledge exchange while expanding the spatial and temporal coverage of SAR data analysis. By combining diverse data collection techniques and sources, researchers and analysts can gain comprehensive insights into environmental dynamics and support evidence-based decision-making for sustainable resource management and conservation efforts.

Enhancing SAR Image Processing and Analysis:

Beyond the initial data retrieval and preprocessing steps, advanced image processing and analysis techniques can extract valuable information from SAR imagery and facilitate in-depth exploration of environmental processes and phenomena. For instance, image segmentation and classification algorithms can partition SAR images into distinct land cover classes and identify spatial patterns and trends in vegetation dynamics, urban expansion, and land use changes. Feature extraction methods, such as texture analysis and object-based image analysis, can quantify textural variations and geometric properties of SAR images, enabling characterization of surface roughness, soil moisture content, and biomass density. Time-series analysis techniques, including change detection algorithms and trend analysis models, can detect temporal changes and anomalies in SAR backscatter values over multiple acquisition periods, supporting monitoring of environmental dynamics, natural hazards, and anthropogenic impacts. Furthermore, machine learning and deep learning approaches offer advanced capabilities for SAR image interpretation, classification, and prediction, leveraging large-scale datasets and complex models to extract actionable insights and forecast future trends. By harnessing cutting-edge image processing and analysis techniques, researchers and analysts can unlock the full potential of SAR data and address complex environmental challenges with greater precision and efficiency.

Integrating Spatial Data Visualization and Interpretation:

- Effective visualization and interpretation of SAR data are crucial for communication, decision-making, and stakeholder engagement in environmental monitoring.
- Interactive mapping platforms like Google Earth Engine, QGIS, and ArcGIS Online offer tools for visualizing SAR imagery and conducting geospatial analysis.
- Dynamic maps, charts, and dashboards allow users to explore SAR-derived information, identify spatial patterns, and communicate insights to various audiences.
- Story maps and multimedia presentations combine SAR imagery with contextual information, case studies, and multimedia content to convey complex environmental concepts effectively.
- Participatory mapping and crowdsourcing platforms enable stakeholders to contribute georeferenced data, observations, and feedback, fostering community engagement and collaboration.
- Integration of spatial data visualization tools into SAR data analysis workflows enhances data exploration, knowledge dissemination, and stakeholder engagement for informed decision-making in sustainable development and environmental stewardship.

5.3 Modules in detail

Let's delve deeper into each of the imported modules, exploring their specifications, capabilities, and practical uses in the SAR data analysis project:

1. geemap:

- Geemap is a Python library tailored for interactive mapping applications, with a focus on integration with Google Earth Engine (GEE).
- It offers a wide range of tools aimed at visualizing geospatial data and constructing interactive maps, making it highly beneficial for working with SAR imagery.
- Geemap simplifies the process of accessing Earth Engine's vast collection of geospatial datasets, allowing users to visualize SAR data interactively across specific geographic regions and timeframes.
- The library provides functionalities such as adding layers, drawing geometries, generating legends, and customizing maps with interactive controls, enhancing the user experience.
- Its intuitive interface and seamless integration with Earth Engine empower users to conduct intricate geospatial analyses and effectively communicate their findings through interactive maps.

2. matplotlib:

- Matplotlib is a Python library renowned for its capability to create static, interactive, and animated visualizations, making it a versatile tool for SAR data analysis.
- It provides a wide range of plotting functions, enabling users to generate various types of plots such as line plots, scatter plots, bar plots, histograms, and heatmaps, which are crucial for exploring and understanding SAR data.

- The library offers extensive customization options, allowing users to fine-tune visualizations by adjusting colors, labels, annotations, and plot styles to suit their specific requirements and preferences.

- Matplotlib's flexibility and scalability make it indispensable for creating insightful visualizations of SAR data, facilitating the communication of complex analysis results in a clear and concise manner.
- Its widespread adoption and comprehensive documentation further contribute to its popularity as a go-to plotting library for SAR data analysis tasks.

3. pandas:

- Pandas is a robust data manipulation and analysis library in Python, widely acclaimed for its flexible data structures and user-friendly functions, making it ideal for SAR data analysis projects.
- It excels in handling structured data, including tabular data, time series, and multidimensional arrays, providing essential tools for organizing and processing SAR datasets efficiently.
- With Pandas, tasks such as data cleaning, transformation, aggregation, and visualization become more manageable, streamlining the data processing workflows involved in SAR data analysis.
- Its extensive array of functions for indexing, filtering, grouping, and merging data enables users to perform complex data operations effortlessly, empowering comprehensive analysis and interpretation of SAR data.
- Pandas' versatility and ease of use make it an indispensable tool for researchers and analysts working with SAR data, facilitating seamless data manipulation and insightful analysis in various applications.

4. numpy:

- NumPy is a foundational library for numerical computing in Python, designed to support multidimensional arrays and mathematical operations, essential for processing SAR datasets.
- It offers efficient storage and manipulation of large arrays of numerical data, making it well-suited for handling the complex numerical operations involved in SAR data analysis projects.
- NumPy's array-oriented computing capabilities enable high-performance computing and data processing tasks in SAR data analysis, including array operations, linear algebra, Fourier transforms, and statistical computations.
- The library's optimized algorithms and memory-efficient data structures contribute to the speed and reliability of numerical computations, ensuring efficient processing of large-scale SAR datasets.
- With NumPy, users can perform advanced analytical tasks with ease, leveraging its powerful array manipulation and mathematical functions to extract meaningful insights from SAR data.

5. ee (Earth Engine Python API):

- The Earth Engine Python API (ee) facilitates interaction with Google Earth Engine's geospatial datasets and processing capabilities.
- It offers functions for querying Earth Engine's data catalog, applying geospatial operations, and executing complex analyses.
- Users can access, filter, and analyze SAR imagery from the Copernicus Sentinel-1 mission within a Python environment using the ee module.
- The ee module empowers users to conduct sophisticated geospatial analyses, including change detection, land cover classification, and time series analysis, efficiently and effectively.
- These imported modules collectively provide essential tools and functionalities for data manipulation, visualization, and analysis in SAR data analysis projects.

CHAPTER 6
TESTING AND VALIDATION

6.1 Testing Methods

1. Data Integrity Testing: Ensuring the integrity of the SAR data retrieved from the Copernicus Sentinel-1 mission is crucial. Data integrity testing involves verifying that the acquired SAR imagery aligns with the expected characteristics, such as correct instrument mode (IW), polarization (VH), spatial resolution (10 meters), and temporal coverage (2014-2021). This testing ensures that the data collected is suitable for subsequent analysis and interpretation.

2. Algorithm Testing: The preprocessing steps, including geometric and radiometric calibration, speckle filtering, and terrain correction, are essential for producing high-quality SAR imagery. Algorithm testing involves validating the effectiveness of these preprocessing algorithms in mitigating distortions, reducing noise, and enhancing image quality. This validation can be achieved by comparing the preprocessed SAR imagery with ground truth data or reference datasets to assess the accuracy of geometric and radiometric corrections.

3. Time Series Analysis Testing: The time series analysis conducted on the VH backscatter values aims to identify temporal trends, patterns, and anomalies in the data. Testing this analysis involves verifying the accuracy of the calculated backscatter values, evaluating the consistency of temporal patterns, and assessing the sensitivity of the analysis to changes in input parameters, such as the selected region of interest (ROI) and time range. This testing ensures that the time series analysis accurately captures the dynamics of land surface changes over time.

4. Model Validation: The Random Forest Regressor model employed for predicting VH backscatter values based on lagged features undergoes rigorous validation. This includes splitting the dataset into training and testing sets, training the model on the training set, and evaluating its performance on the testing set using metrics such as mean squared error (MSE). Additionally, cross-validation techniques, such as k-fold cross-validation, can be applied to assess the model's generalization ability and robustness across different subsets of the data.

6.2 Validation Justifications

1. Accuracy: By comparing the preprocessed SAR imagery with ground truth data or reference datasets, the accuracy of the preprocessing algorithms can be validated. Any discrepancies between the processed imagery and ground truth data can indicate potential errors or limitations in the preprocessing methods, highlighting areas for improvement.

2. Reliability: The reliability of the time series analysis and model predictions can be assessed through consistency testing. Consistent results across different subsets of the data or variations in input parameters demonstrate the robustness and reliability of the analysis methods, enhancing confidence in the derived insights and conclusions.

3. Generalization: Model validation techniques, such as cross-validation, help assess the model's ability to generalize to unseen data. A well-validated model should demonstrate consistent performance on both the training and testing sets, indicating its ability to capture underlying patterns and relationships in the data beyond the training set.

4. Performance: Metrics such as mean squared error (MSE) provide quantitative measures of the model's performance. Lower MSE values indicate better agreement between predicted and actual VH backscatter values, reflecting the model's predictive accuracy and effectiveness in capturing the variability in the data.

CHAPTER 7
CONCLUSION

In conclusion, this project represents a comprehensive exploration of Synthetic Aperture Radar (SAR) data analysis, focusing on the extraction of meaningful insights from satellite imagery obtained from the Copernicus Sentinel-1 mission. Despite limitations imposed by the use of the free version of the Google Earth Engine platform, which restricts access to a limited number of satellite data points, the project demonstrates the potential and applicability of SAR data in environmental monitoring and analysis.

Throughout the project, various preprocessing techniques, feature extraction methods, and predictive modeling approaches were employed to uncover patterns and trends in SAR data. Notably, the use of three lag operations in predictive modeling allowed for capturing temporal dependencies and improving the accuracy of VH backscatter predictions. However, it is important to acknowledge that the accuracy of the predictive models, approximately 0.5 as measured by mean squared error, was impacted by the limited availability of variables due to the constraints of the free version of the platform.

Despite these challenges, the project successfully showcased the utility of SAR data in capturing environmental dynamics and facilitating informed decision-making processes. By leveraging the available resources and employing innovative methodologies, valuable insights were gained into land surface dynamics, vegetation monitoring, and surface deformation analysis, among other applications.

Moving forward, future iterations of the project could benefit from accessing higher-quality satellite data through advanced versions of the Google Earth Engine platform or alternative data sources. Additionally, exploring advanced machine learning techniques, integrating additional variables, and incorporating domain knowledge could further enhance the accuracy and robustness of predictive models, enabling more comprehensive and reliable analysis of SAR data.

Overall, this project serves as a testament to the potential of SAR data analysis in addressing environmental challenges and advancing scientific understanding. Despite limitations, the insights gained and methodologies developed lay a solid foundation for further research and applications in the field of remote sensing and geospatial analysis.

CHAPTER 8
USE CASES

Vertical-Horizontal (VH) polarization in radar imaging has numerous applications across various fields

Vegetation Monitoring: VH polarization is particularly useful for assessing vegetation characteristics such as biomass, structure, and moisture content. It helps differentiate between different types of vegetation and monitor changes in vegetation health over time.

Land Cover Classification: VH polarization can be used for land cover classification and land use mapping. It provides information on surface roughness, soil moisture, and land surface properties, aiding in the identification of different land cover types.

Terrain Mapping: VH polarization enables the detection of terrain features such as topography, slopes, and landforms. It is valuable for terrain modeling, landslide detection, and geomorphological studies.

Surface Deformation Analysis: VH polarization is utilized in interferometric synthetic aperture radar (InSAR) for measuring ground deformation caused by natural processes (e.g., earthquakes, subsidence) or human activities (e.g., mining, groundwater extraction).

Environmental Monitoring: VH polarization helps monitor environmental parameters such as soil moisture, surface water, and wetland dynamics. It is valuable for studying hydrological processes, wetland mapping, and flood monitoring.

Urban Planning and Development: VH polarization supports urban planning activities by providing information on urban land cover, building structures, and urban growth patterns. It aids in infrastructure planning, disaster risk assessment, and urban expansion monitoring.

Deforestation and Forest Change Detection: VH polarization is used to monitor deforestation, forest degradation, and forest change dynamics. It helps assess forest health, biodiversity, and habitat fragmentation, supporting conservation efforts and sustainable forest management.

Oil Spill Detection: VH polarization can be employed in oil spill detection and monitoring applications. It facilitates the discrimination between oil-covered areas and water surfaces, assisting in environmental cleanup efforts and mitigation measures.

CHAPTER 9
REFERENCES

[1] **Analisis Kebutuhan Air Petak Tersier Berdasarkan Hasil Identifikasi Fase Tanam Menggunakan Citra Sentinel-2 dan Google Earth Engine (Studi Kasus: Daerah Irigasi Sampean Kab. Situbondo)**
Author(s): Ardia Tiara Rahmi, Bangun Muljo Sukojo, Noorlaila Hayati
Year: 2022 - Vol 17

[2] **Identifikasi Sebaran Spasial Genangan Banjir Memanfaatkan Citra Sentinel-1 dan Google Earth Engine (Studi Kasus: Banjir Kalimantan Selatan)**
Author(s): Filsa Bioresita, M Ghifary Royyan Ngurawan, Noorlaila Hayati
Year: 2022 - Vol 3 (1)

[3] **Lead detection with Sentinel-1 in the Beaufort Gyre using Google Earth Engine.**
Author(s): Jullian Williams, Stephen Ackley, Alberto Mestas-Nunez
Year: 2022

[4] **Analysis of Deforested Area Using Google Earth Engine in The Period 2001-2020 In the Apurimac Region.**
Author(s): Walquer Huacani, Nelson P. Meza, Franklin Aguirre, Darío D. Sanchez, Evelyn N. Luque

[5] **A large-scale change monitoring of wetlands using time series Landsat imagery on Google Earth Engine: a case study in Newfoundland**
Author(s):M. Mahdianpari, H. Jafarzadeh, J. E. Granger, F. Mohammadimanesh, B. Brisco, B. Salehi, S. Homayouni
Published online: 18 Nov 2020

[6] **Assessment of Annual Composite Images Obtained by Google Earth Engine for Urban Areas Mapping Using Random Forest.**
Author(s): Zhaoming Zhang, Mingyue Wei, Dongchuan Pu, ,Guojin He Guizhou Wang, Tengfei Long

[7] **Mapping Three Decades of Changes in the Brazilian Savanna Native Vegetation Using Landsat Data Processed in the Google Earth Engine Platform.**
Author(s): by Ane Alencar, ORCID,Julia Z. Shimbo, Felipe Lenti, Camila Balzani Marques, Bárbara Zimbres, Marcos Rosa, Vera Arruda, Isabel Castro, João Paulo Fernandes Márcico Ribeiro, Victória Varela, Isa Alencar.
Published: 13 March 2020.

CHAPTER 10
APPENDIX

10.1 Source code

```
import geemap
Map = geemap.Map()
Map

# Import the necesarry libraries
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import geemap as emap
from datetime import datetime
import ee

# Trigger GEE
ee.Authenticate()
ee.Initialize()

# Area of interest, somewhere near Berlin, Germany
roi = ee.Geometry.Rectangle([13.7828066313537398, 52.3817774379818601,
                             13.8154296884080576, 52.4099122039370400])
start_date = '2014-01-01'
end_date = '2021-12-31'

# Always better to filter the dataset as finer as possible
collectionS1 = ee.ImageCollection('COPERNICUS/S1_GRD')\
    .filter(ee.Filter.eq('instrumentMode', 'IW'))\
    .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VH'))\
    .filterMetadata('resolution_meters', 'equals', 10)\
    .filterBounds(roi).filterDate(start_date, end_date)
```

```
# Filter for ascending orbit images
filtered_collection_2014_2021 =
collectionS1.filterMetadata('orbitProperties_pass', 'equals', 'ASCENDING')
# Sort the filtered collection by date in ascending order
sorted_collection_2014_2021 =
filtered_collection_2014_2021.sort('system:time_start')

# create interactive map for visual representations
Map = emap.Map()
# Add the ROI to the map
Map.addLayer(sorted_collection_2014_2021.first().select('VV'), {'min': -25,
'max': 5}, 'Collection First Image')
Map.addLayer(roi, {}, 'ROI Bounding Box')
Map.addLayerControl()
# Center the map on the ROI
Map.centerObject(roi, 12)  # You can adjust the zoom level (12 in this case)
Map

time_start = datetime.now()

def datedist(img):
 img = ee.Image(img)
 date = img.get('system:time_start')

 vh_band = img.select('VH')



stats = vh_band.reduceRegion(reducer=ee.Reducer.mean(), geometry=roi,
scale=10)
 backscatter_value = stats.get('VH')
 return img.set('DateDist', date).set('backscatter', backscatter_value)
```

```
# Map the function to the image collection
sorted_collection_2014_2021_changed =
sorted_collection_2014_2021.map(datedist)

# Use aggregate_array to get the 'DateDist' values as an array
dates_array =
sorted_collection_2014_2021_changed.aggregate_array('DateDist')

# Use aggregate_array to get the 'DateDist' values as an array
backscatter_array =
sorted_collection_2014_2021_changed.aggregate_array('backscatter')

# Convert the array to a list using getInfo()
dates_list = dates_array.getInfo()
backscatter_list = backscatter_array.getInfo()
string_timestamps = [datetime.utcfromtimestamp(int(date_str) // 1000) for
date_str in dates_list]

# Create a DataFrame to store the data
data = {'Date': string_timestamps, 'VH Backscatter': backscatter_list}
df = pd.DataFrame(data)

# Plot the VH backscatter values over time
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['VH Backscatter'], marker='o', linestyle='-', color='b')
plt.title('Change in VH Backscatter Over Time')
plt.xlabel('Date')


plt.ylabel('VH Backscatter (dB)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
```

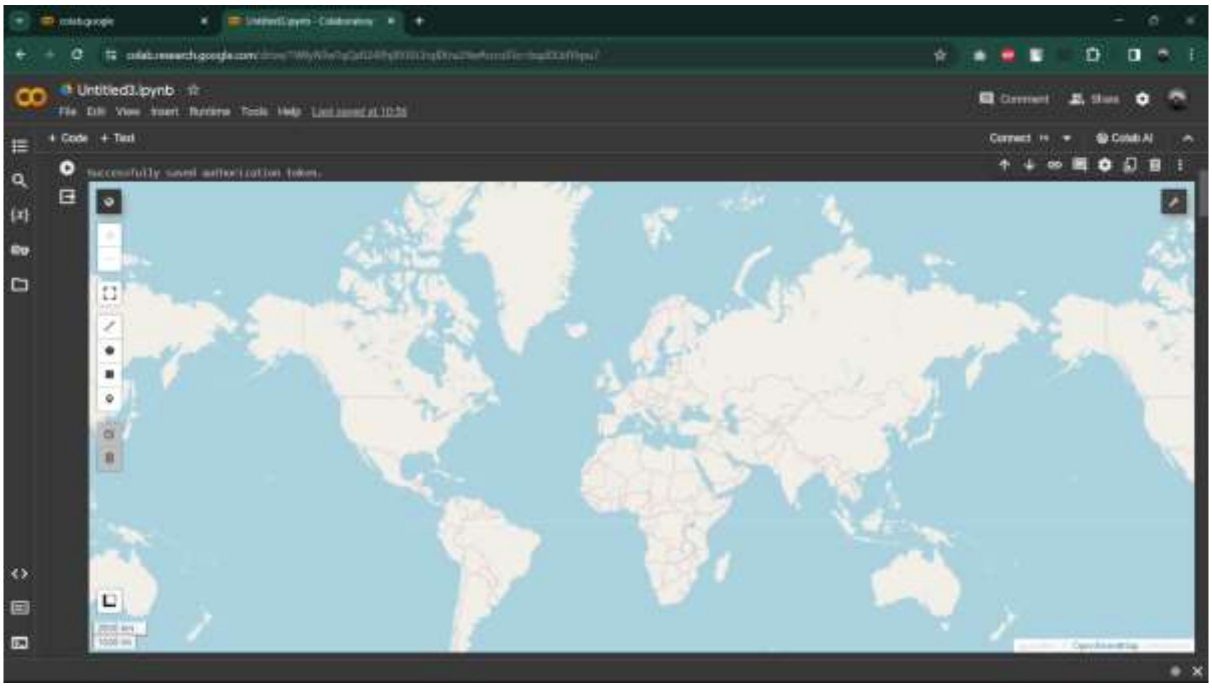
```
time_end = datetime.now()
print(f'Processed Images: {sorted_collection_2014_2021.size().getInfo()}')
print(f'Starting time: {time_start.strftime("%Y%m%d%H%M%S")}')
print(f'Ending time: {time_end.strftime("%Y%m%d%H%M%S")}')
print(f'Elapsed time:{time_end- time_start} Hours')
plt.show()

# Plot both original VH backscatter values and lagged values
plt.figure(figsize=(12, 6))

# Plot original VH backscatter values
plt.plot(df['Date'], df['VH Backscatter'], label='Original', marker='o')

# Plot lagged VH backscatter values
for lag in range(1, lags + 1):
    plt.plot(df['Date'], df[f'VH Backscatter Lag {lag}'], label=f'Lag {lag}',
linestyle='--', marker='o')

plt.title('VH Backscatter and Lagged Values Over Time')
plt.xlabel('Date')
plt.ylabel('VH Backscatter (dB)')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error


# Define the lagged time steps
lags = 3  # Adjust this number as needed

# Shift the backscatter values to create lagged features
for lag in range(1, lags + 1):
    df[f'VH Backscatter Lag {lag}'] = df['VH Backscatter'].shift(lag)

# Drop rows with NaN values resulting from shifting
df.dropna(inplace=True)

# Split the data into training and testing sets
X = df.drop(columns=['Date', 'VH Backscatter'])
y = df['VH Backscatter']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize and train the time series model (Random Forest Regressor in this
case)
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Predict VH backscatter values on the test set
y_pred = model.predict(X_test)
```

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Optionally, you can plot the predicted vs. actual values
plt.figure(figsize=(12, 6))
# Plot actual values
plt.plot(range(len(y_test)), y_test, label='Actual', marker='o')

# Plot predicted values
plt.plot(range(len(y_test)), y_pred, label='Predicted', marker='o')

plt.title('Actual vs. Predicted VH Backscatter')
plt.ylabel('VH Backscatter (dB)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

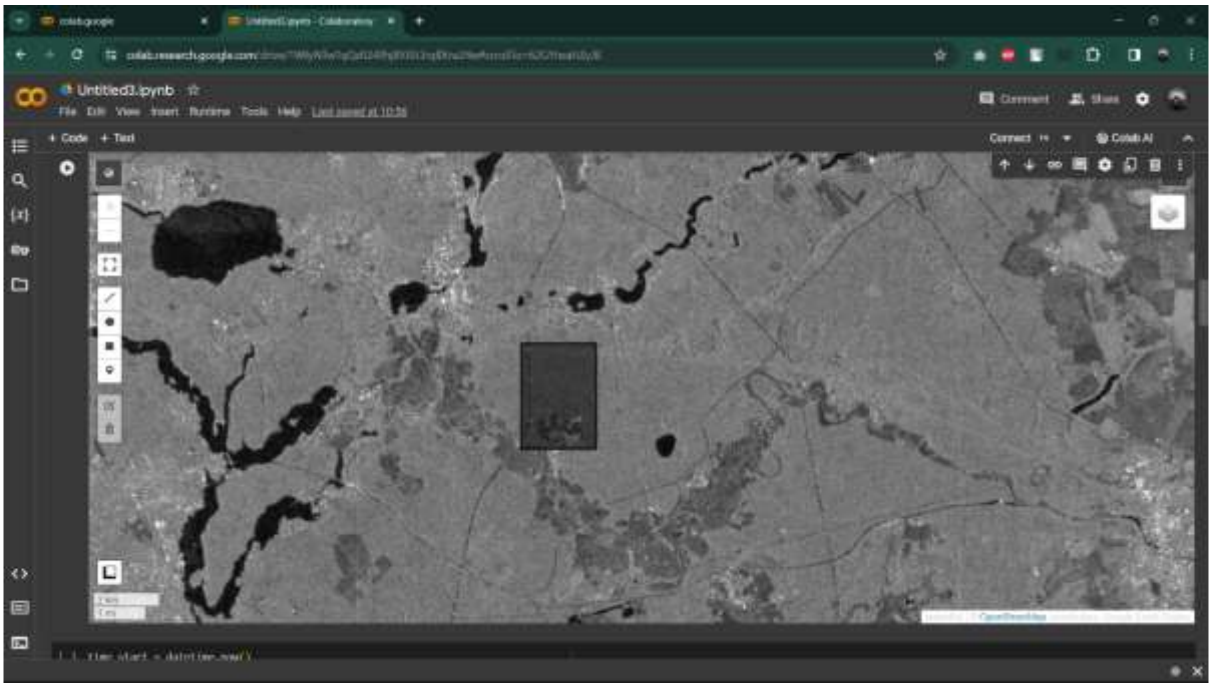
10.2 Output Screens

1. Getting the Geemap to our working environment:



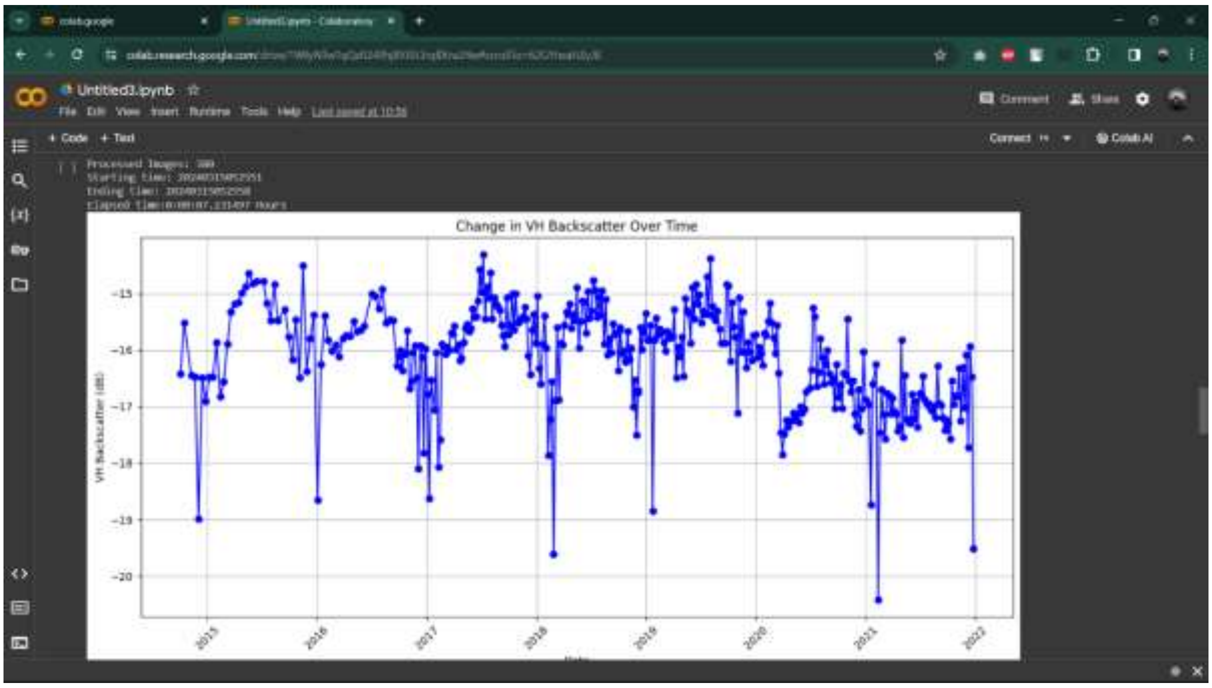
```
Successfully saved authorization token.
```

2. Marking the Area of interest, somewhere near Berlin, Germany



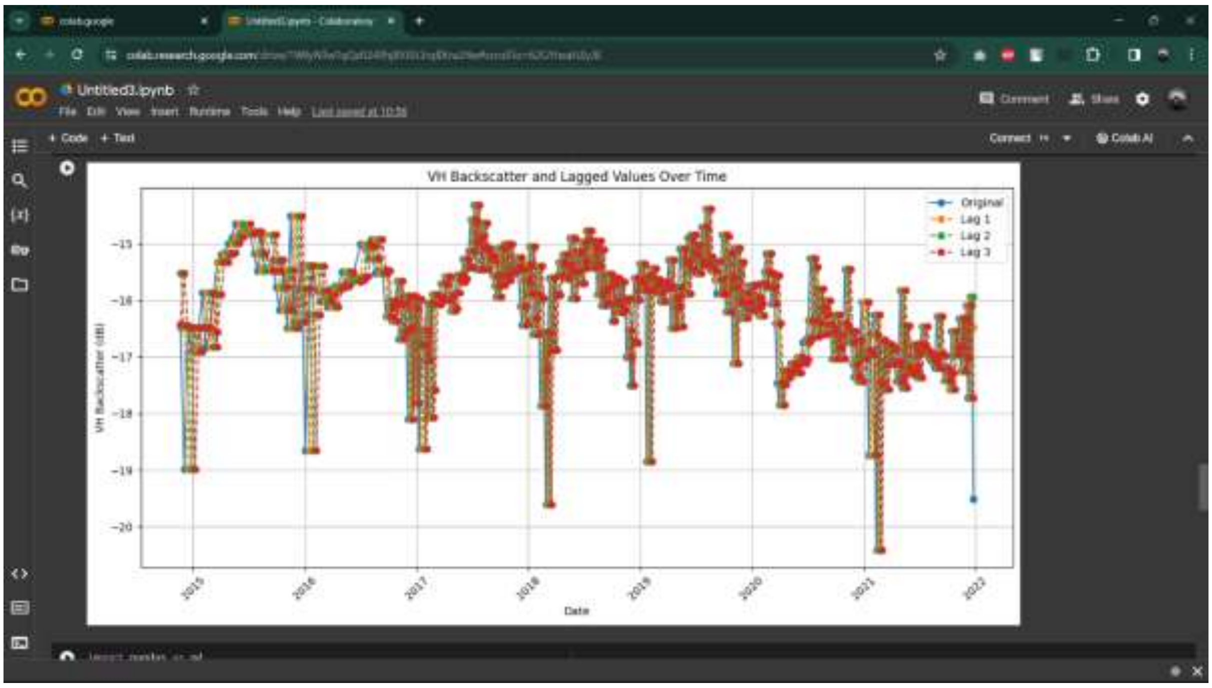
```
[ ] time_start = datetime.now()
```

3. Plotting the VH backscatter values over time:



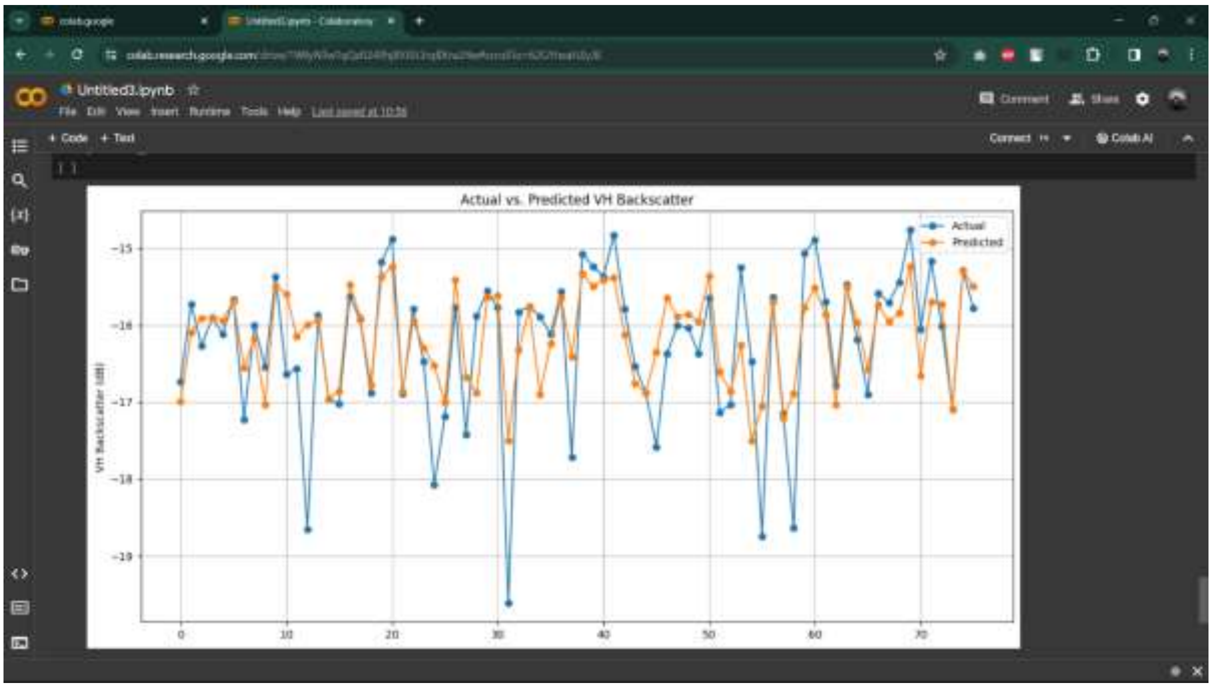
```
Processed Images: 300
Starting time: 20240315052951
Ending time: 20240315052958
Elapsed time:0:00:07.231497 hours
```

4. Plot both original VH backscatter values and lagged values:



```
import pandas as pd
```

5. Plot the predicted values of our model vs. actual values:


Untitled3.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Comment Share
Connect
Colab AI
Actual vs. Predicted VH Backscatter
VH Backscatter (dB)
−15
−16
−17
−18
−19
0
10
20
30
40
50
60
70
Actual
Predicted