Ejercicios de Percepción Remota con Google Earth Engine[1]

Ricardo Andres Martinez Soto[2]

Resumen

Este informe presenta ejercicios prácticos de percepción remota desarrollados en Google Earth Engine (GEE). Tiene como objetivo fortalecer y facilitar la comprensión de las herramientas de percepción remota en la nube y su aplicación práctica en el análisis de imágenes satelitales mediante conceptos teóricos-prácticos de análisis espacial y procesamiento de imágenes satelitales. Para esto se realizaron ejercicios propuestos en el curso y otros del libro "Cloud-Based Remote Sensing with Google Earth Engine". Las temáticas abordadas incluyen composiciones de color y el cálculo de índices espectrales. Para replicar estos ejercicios se requiere una cuenta en GEE y conocimientos básicos en Python.

Palabras claves: GEE, Percepción Remota, Python

Remote Sensing Exercises with Google Earth Engine

Abstract

This report presents practical remote sensing exercises developed in Google Earth Engine (GEE). It aims to strengthen and facilitate the understanding of remote sensing tools in the cloud and their practical application in the analysis of satellite images through theoretical and practical concepts of spatial analysis and satellite image processing. For this purpose, exercises proposed in the course and others from the book "Cloud-Based Remote Sensing with Google Earth Engine" were carried out. The topics addressed include color compositions and the calculation of spectral indices. To replicate these exercises a GEE account and basic knowledge of Python are required.

Keywords: GEE, Remote Sensing, Python

[1] Reporte técnico de ejercicios con Google Earth Engine (GEE) como parte del curso de Percepción Remota de la Maestría en Geomatica de la Universidad Nacional de Colombia.

[2] Correo electrónico autor: martinezsoto.ra@gmail.com o rimartinezs@unal.edu.co.

¿Qué es Google Earth Engine?

Google Earth Engine es una plataforma en la nube para el análisis geoespacial que aprovecha las capacidades computacionales de Google para abordar diversos problemas sociales de gran impacto, como la deforestación, la sequía, los desastres, las enfermedades, la seguridad alimentaria, la gestión del agua, la monitorización del clima y la protección del medio ambiente.

Es única en su campo como plataforma integrada, diseñada para empoderar no solo a los científicos especializados en teledetección, sino también a un público mucho más amplio que carece de la capacidad técnica necesaria para utilizar supercomputadoras tradicionales o recursos de computación en la nube a gran escala. (Gorelick et al., 2017)

1. **Interfaz de usuario GEE y**

Como se mencionó anteriormente, GEE es una aplicación en la nube, la cual presenta un interfaz que se divide en cuatro grandes secciones como se puede evidenciar en la figura 1. Estas cumplen las siguientes funciones:

1. Herramientas de la cuenta: Permite identificar fácilmente los archivos creados y los roles de cada uno de los códigos disponibles.

2. Code editor: Cuenta con entorno de desarrollo integrado (IDE) basado en JavaScript para procesar y analizar datos geoespaciales.

3. Resultados y tareas: Muestra las tareas que se han iniciado la Console Output de la ejecución del script.

4. Mapa: Visualización principal de resultados de análisis, las capas añadidas, las geometrías dibujadas e interacción con los datos.

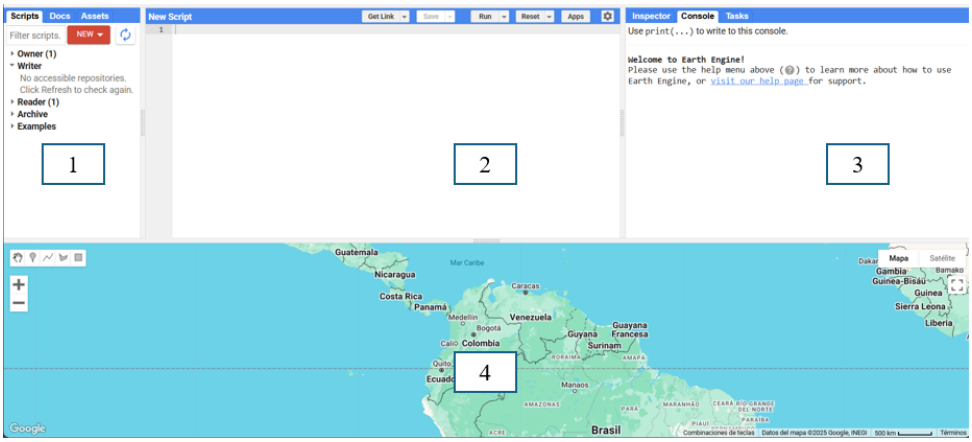

Figura 1: Interfaz GEE

Es importante mencionar que dentro de las secciones anteriormente mencionadas existen otras funciones que cumplen tareas específicas y hacen que sea un interfaz más completa y fácil de usar. No obstante, con el tiempo puede generarse una dependencia hacia esta interfaz. Por ello, surgen alternativas de uso, como el acceso a través de APIs, en particular la API de Python. A continuación, se muestran las diferencias principalmente en términos de ventajas y desventajas del uso de Python frente a la interfaz gráfica de Google Earth Engine (GEE):

Interfaz de GEE (JavaScript)	
Ventajas	***Desventajas***
Facilidad de uso: Ideal para usuarios que no tienen experiencia en programación.	Limitaciones en automatización: Menor capacidad para automatizar flujos de trabajo extensos o complejos.
Rápida visualización: Permite generar mapas y resultados de forma rápida e intuitiva.	Dependencia de la plataforma: La personalización avanzada puede ser limitada.
Menor configuración: No requiere instalaciones adicionales más allá de un navegador web.	Escalabilidad: Puede ser poco eficiente para proyectos de gran tamaño o que requieran procesos complejos.
Acceso a ejemplos: Incluye abundantes ejemplos, plantillas y documentación visual.	Reproducibilidad: Resulta más difícil versionar y documentar procesos de forma estructurada.

Tabla 1: Ventajas y desventajas del interfaz de GEE

Python con GEE	
Ventajas	***Desventajas***
Automatización: Permite automatizar tareas repetitivas mediante scripts.	Curva de aprendizaje: Requiere conocimientos previos de programación en Python y de cómo funciona la API de GEE.
Integración: Se puede integrar con otras bibliotecas de procesamiento de datos (como Pandas, NumPy, Matplotlib, TensorFlow, entre otras).	Mayor configuración inicial: Es necesario instalar paquetes y autenticar el acceso a GEE.
Escalabilidad: Es más sencillo gestionar proyectos de mayor complejidad mediante programación estructurada.	Depuración de errores: La identificación y corrección de errores puede ser más compleja que en el entorno gráfico.

Reproducibilidad: Facilita documentar y reproducir flujos de trabajo completos.	Menor inmediatez: Para tareas simples, puede ser más lento que usar directamente la interfaz gráfica de GEE.

Tabla 2: Ventajas y desventajas de Python con GEE

Teniendo en cuenta las ventajas y desventajas expuestas anteriormente para el desarrollo de los ejercicios se realizarán en Python. Esto permitirá aprovechar la flexibilidad, capacidad de automatización y análisis de con otras herramientas de análisis. Adicionalmente, se encuentran almacenados en un repositorio de GitHub[3].

Para realizar la conexión al api de Google Earth Engine debemos tener una versión de Python compatible con las últimas versiones, estas están disponibles en la documentación de Google Claude Platform[4]. Dependiendo de la manera en que se tenga instaldo Python se debe realizar la instalación del paquete de GEE. A través de ***pip*** o ***Conda forge*** para estos se debe ejecutar en la terminal alguno de los siguientes comandos:

conda update -c conda-forge earthengine-api

pip install earthengine-api --upgrade

Una vez instalado correctamente se debe realizar importar la ee dentro de nuestro codigo, para esto se mostrará como cargar un archivo GeoJson como assets cualquier proyecto de GEE.

Importar librerias requeridas

```python
import ee
import geemap
import json
import os
```

Python

Figura 2: Librerías importadas para la ejecución del código

[3] Repositorio de GitHub; https://github.com/RicardoMartinezS/GEE-Remote-Sensing - RicardoMartinezS/GEE-Remote-Sensing: This repository presents practical exercises of remote sensing developed in Google Earth Engine. It is part of the Remote Sensing course of the Master in Geomatics of the National University of Colombia.

[4] Documentación versiones de Python disponible https://developers.google.com/earth-engine/guides/python_install

Las librerias importadas son:

- os: Para el acceso a la terminal del equipo y con esto poder acceder de manera nativa y sencilla rutas relativas y absolutas de los archivos cargados.
- Geemap: Para poder visualizar de manera grafica las salidas de GEE dentro de un entrono de HTML
- Json: Extensión para poder interpretar archivos json y geojson como nativa.
- ee: Para poder usar el API de GEE en el entorno de python

Es importante mecionar que al ejecutar ee.Authenticate en el navegador de internet nos desplegará una ventana de inicio de sesión de Google en el cual debemos ingresar nuestras credenciales además arrojando un token que se debe ingresar en la terminal. Este ejercicio solo se debe realizar una vez y como previo al desarrollo de este ejercicio ya se habia hecho no se puede mostrar. Por otra parte es necesario inicializar un proyecto ya creado de GEE y que se cuente con permisos de edición para que pueda interactuar de manera correcta la API

Figura 3: Inicializador GEE y carga de datos de un Feature Collection

2. Composiciones a Color[5]

Una composición a color es una técnica para visualizar imágenes satelitales asignando colores (rojo, verde y azul) a diferentes bandas espectrales captadas por sensores ya que la percepción remota se basa en la detección y análisis de la radiación electromagnética reflejada o emitida por los objetos en la superficie terrestre(Chuvieco, 1996). A continuación, se mostrarán algunos ejemplos de unas composiciones a color.

Para el desarrollo de este ejercicio se utilizó una colección de imágenes satelitales extraídas de Google Earth Engine (GEE), específicamente del conjunto de datos **'LANDSAT/LC08/C02/T1_L2'**, filtrado para un período de un año (2023) y restringido al límite

[5]Repositorio GitHub: GEE-Remote-Sensing/Notebooks/1_Color_Composition.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

administrativo de la ciudad de **Bogotá D.C., Colombia** como zona de estudio. Esta extracción se logró mediante el siguiente fragmento de código:

Extracción de la colección e imagenes de GEE

```python
landsat8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
    .filterBounds(zona_estudio) \
    .filterDate('2023-01-01', '2023-12-31')

print('Número de imágenes:', landsat8.size().getInfo())
image = landsat8.sort('CLOUD_COVER').first()
image_id = image.getInfo()['id']
print("Id imagen a trabajar",image_id)
```

Python

Figura 4: Código de Python para la extracción de imágenes Landsat

Dado que el período de análisis abarca un año completo, la colección resultante contiene múltiples imágenes. Como el objetivo del estudio es generar composiciones en color, se seleccionó aquella imagen con la menor cobertura de nubes. Al ejecutar el código anterior, se obtuvieron los siguientes resultados:

Número total de imágenes: 18

ID de la imagen seleccionada: LANDSAT/LC08/C02/T1_L2/LC08_008057_20230603

Para visualizar esta imagen en GEEMap, es necesario proporcionar como parámetros: la imagen a mostrar, la información correspondiente a las bandas, así como los valores mínimo, máximo y gamma, organizados en un arreglo que GEEMap pueda interpretar, junto con el nombre de la capa.

Finalmente, con el fin de optimizar los recursos y mejorar la visualización, la imagen seleccionada fue recortada para ajustarse a la zona de interés.

```
rgb_vis = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 7000,
    'max': 20000,
    'gamma': 2.5
}

reflectance_vis = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 0.00,
    'max': 0.12,
    'gamma': 2.0
}
escala = 0.0000275
intercepto = -0.2

image_SR = image.select(['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7'])
    .multiply(escala).add(intercepto)
```

Figura 5: Parámetros de rescalado de imágenes Landsat 8

A continuación, se crea y configura un mapa interactivo utilizando la biblioteca geemap, una herramienta de Python que permite visualizar datos geoespaciales provenientes de Google Earth Engine. En primer lugar, se inicializa el mapa con dimensiones específicas y se centra en el área de estudio, aplicando un nivel de zoom de 10. Posteriormente, se añaden al mapa las distintas capas generadas a partir de diversas composiciones de color.

```
Map = geemap.Map()
Map = geemap.Map(height=300, width='70%')
Map.centerObject(zona_estudio, zoom =10)
Map.addLayer(image_clipped, rgb_vis, "True Color")
Map.addLayer(image_clipped_SR,reflectance_vis,"True color reflectance")
Map.addLayer(image_clipped_SR, false_color_564, "False Color 564")
Map.addLayer(image_clipped_SR, false_color_572, "False Color 572")

Map
```

Figura 6: Creación del mapa interactivo

Cada tipo de cobertura de la tierra tiene una firma espectral única que representa el porcentaje de reflectancia capturada por el sensor, lo que en términos gene en esto permite su identificación en imágenes multiespectrales o hiperespectrales. Gracias a estas firmas espectrales es posible utilizar algoritmos de clasificación supervisada y no supervisada (Richards, 1999).

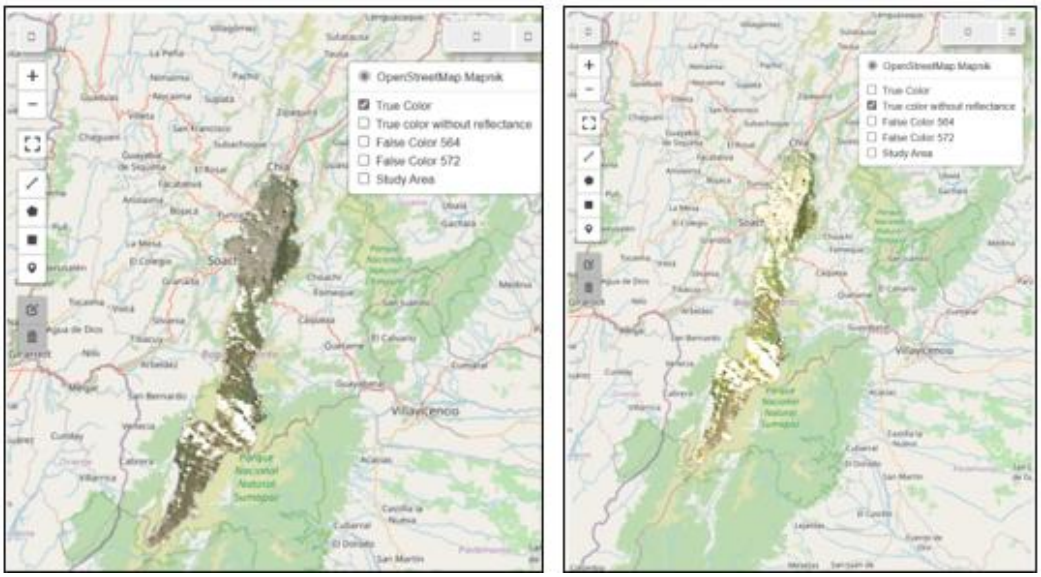

Figura 7: Composiciones de color natural de la ciudad de Bogotá D.C.

Las dos imágenes mostradas en la figura 7 presentan dos composiciones en color natural una en su resolución original y otra rescalada. Estas imágenes se utilizan principalmente para la interpretación visual en teledetección, ya que permiten observar la superficie terrestre de forma similar a como la veríamos a simple vista lo que facilita la interpretación visual.

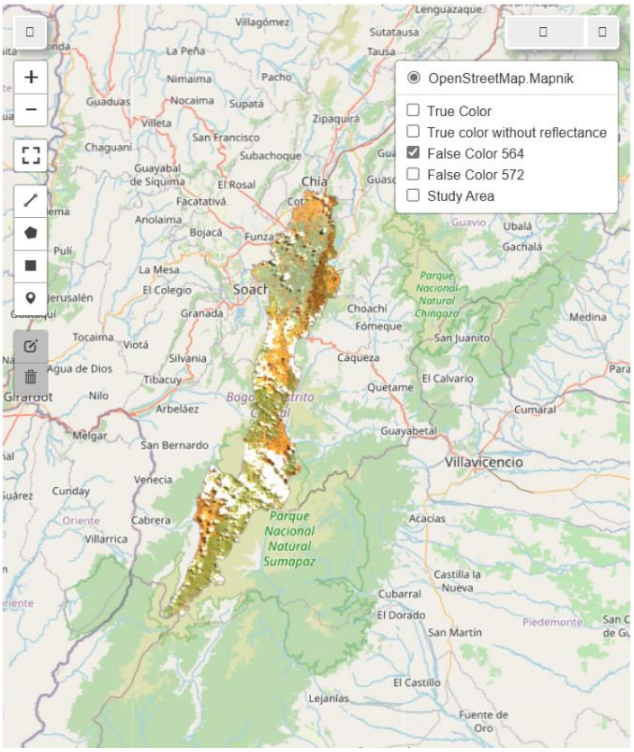

Figura 8: Composición falso color 584

Esta combinación asigna infrarrojo cercano (Banda 5) al canal rojo, SWIR 1 (Banda 6) al canal verde y rojo (Banda 4) al canal azul. Es altamente efectiva para delinear cuerpos de agua, distinguirlos de tierras estériles e identificar formaciones de hielo. El agua típicamente aparece muy oscura o negra debido a la fuerte absorción en NIR y SWIR, mientras que las características del terreno exhiben una respuesta espectral más variada, lo que permite una clara separación. (Joyner Elizabeth, 2025)

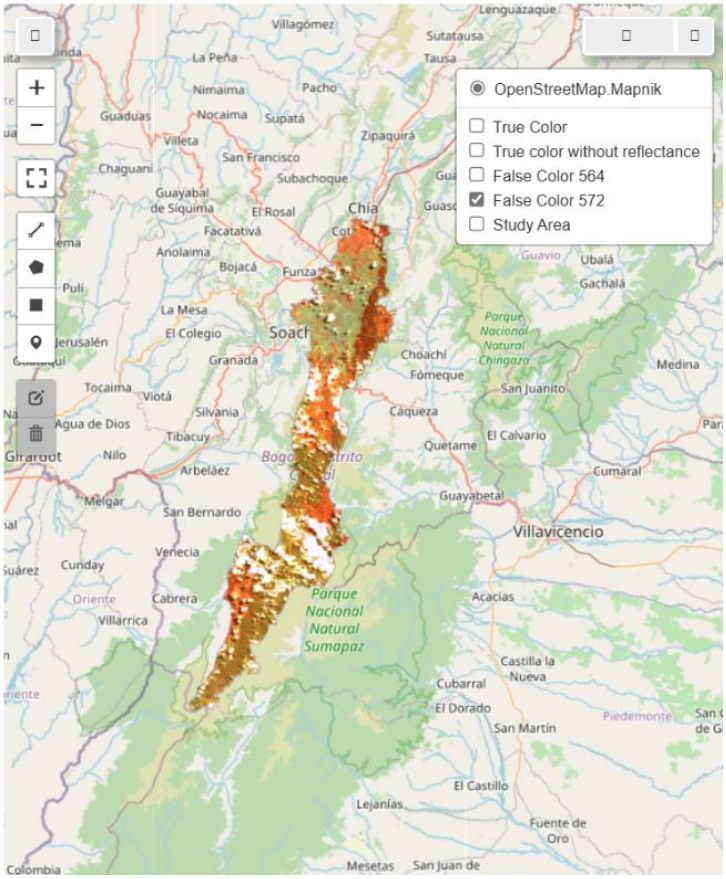

Figura 9: Composición falso color 572

Esta combinación asigna SWIR 2 (Banda 7) al canal rojo, infrarrojo cercano (Banda 5) al canal verde y azul (Banda 2) al canal azul. Es extremadamente efectiva para identificar y mapear cicatrices de quemaduras resultantes de incendios forestales y para evaluar la extensión de las áreas quemadas. Las áreas quemadas exhiben cambios espectrales distintos, a menudo apareciendo en colores únicos debido a la alteración de las propiedades de reflectancia de la superficie después del fuego, lo que facilita el monitoreo de la recuperación (Joyner Elizabeth, 2025)

3. NDVI Landsat (Polígonos aleatorios)[6]

Para este ejercicio, se realizó un muestreo de puntos aleatorios dentro de la zona de estudio correspondiente a la localidad de Usme, en Bogotá D.C. Esta área fue seleccionada debido a su diversidad en los usos del suelo, ya que presenta tanto zonas urbanas como rurales, lo que permite analizar diferentes tipos de coberturas.

A partir de estos puntos, se generaron polígonos de Voronoi, que es una herramienta geométrica que divide el plano en regiones denominadas celdas de Voronoi. Cada celda contiene todos los puntos del espacio que están más cerca de su punto generador que de cualquier otro. Estas celdas son polígonos convexos que, en conjunto, forman una teselación completa del plano, es decir, una partición sin superposiciones ni espacios vacíos(Banik et al., 2021). Esto se realizó mediante el software QGIS con el Bbox de la zona de interés y se realizó el respectivo clip tal y como se muestra en la figura 10.

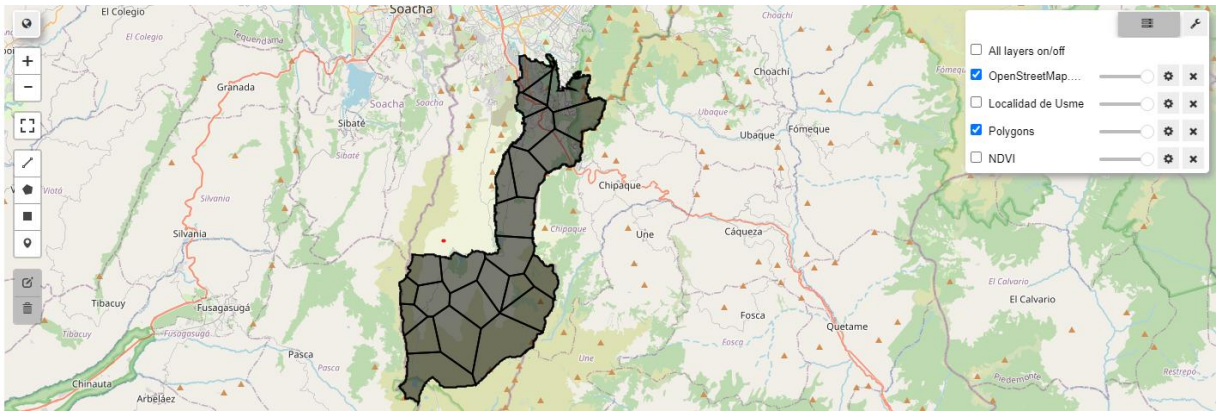

Figura 10: Polígonos de Voronoi en la localidad de Usme

El Índice de Vegetación de Diferencia Normalizada (NDVI) es el índice de vegetación más utilizado y fundamental, a menudo denominado "índice de verdor". Cuantifica la verdor, la densidad y la salud de la vegetación aprovechando el comportamiento espectral contrastante de las plantas sanas en las porciones roja e infrarroja cercana (NIR) del espectro electromagnético. Como se mencionó, la vegetación sana absorbe fuertemente la luz roja para la fotosíntesis y refleja en gran medida la luz NIR debido a su estructura foliar interna

$$\mathrm{NDVI} = \frac{\rho\mathrm{NIR} - \rho\mathrm{RED}}{\rho\mathrm{NIR} + \rho\mathrm{RED}}$$

Ecuación 1: Formula para el cálculo del NDVI

[6] Repositorio Github GEE-Remote-Sensing/Notebooks/2_NDVI_Landsat_calculation.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

Los datos del NDVI de Landsat se encuentran disponibles para descargar a través de la pagina web de la USGS específicamente en la sección de EROS Science Processing Architecture On Demand Interface[7]. Esta información se puede descargar a través de formatos GeoTIFF o atraves de la API incluyendo los siguientes parámetros de búsqueda

Landsat Normalized Difference Vegetation Index (NDVI) Specifications

Attribute	Value
Long Name	Normalized Difference Vegetation Index
Short Name	LC8NDVI, LE7NDVI, LT5NDVI, or LT4NDVI
File Name	*_sr_ndvi.tif
Data Type	Signed 16-bit Integer
Units	Spectral Index (Band Ratio)
Valid Range	-10,000 — 10,000
Fill Value	Collection 2: -19999 Collection 1: -9999
Saturate Value	20,000
Scale Factor	*0.0001

Tabla 3: Parametros tecnios para el NDVI en una imagen Landsat

No obstante, para el ejercicio se va a realizar a través de una definición de una función en Python que nos permite calcular el NDVI. Esto se realizó a través del siguiente código:

```
# Calcular NDVI
def calcNDVI(image):
    ndvi = image.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI')
    return image.addBands(ndvi).copyProperties(image, image.propertyNames())

l8Ndvi = aoi_L8_sr.map(calcNDVI)

#Imagen con menos nubes
ImageNDVI = l8Ndvi.sort('CLOUD_COVER').first()

ndvi_palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
                '74A901', '66A000', '529400', '3E8601', '207401', '056201',
                '004C00', '023B01', '012E01', '011D01', '011301']

ImageNDVI_id = ImageNDVI.getInfo()['id']
print(ImageNDVI_id)
```

Figura 11: Calculo de NDVI para la primera imagen de la colección

[7] https://espa.cr.usgs.gov/

Una vez calculado el NDVI de la imagen ya cortado se muestra y se almacena en una composion de color creada partir de la palaeta de colores predefinida en el codigo arrojad el siguiente resultado:

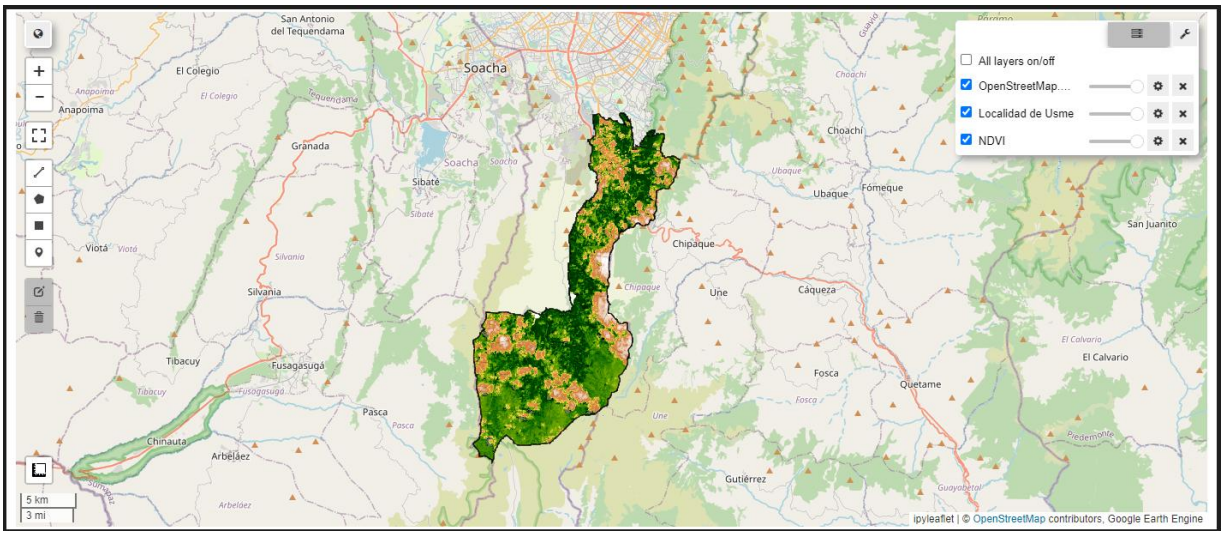

Figura 12: Visualización NDVI de la imagen 1 de la colección

No obstante, como el objetivo del codigo no solo es el calculo del NDVI, si no de analizar las estadísticas zonales en una serie de tiempo, se debe verificar si las imágenes cubren toda la zona de estudio, para esto se realizó un análisis de una mascara de la zona de estudio con las imágenes encontradas como se muestra a continuación:

Verificación cobertura de cada imagen sobre el área de estudio

```
imagenes = aoi_L8col.toList(aoi_L8col.size())
area_total = zona_estudio.geometry().area().getInfo()

resultados = []

for i in range(aoi_L8col.size().getInfo()):
    img = ee.Image(imagenes.get(i))
    img_id = img.id().getInfo()
    mascara_valida = img.select(0).mask().gt(0)
    pix_area = ee.Image.pixelArea().updateMask(mascara_valida)
    area_cubierta = pix_area.reduceRegion(
        reducer=ee.Reducer.sum(),
        geometry=zona_estudio.geometry(),
        scale=10,
        maxPixels=1e10
    ).get('area').getInfo()
    porcentaje_cobertura = (area_cubierta / area_total) * 100 if area_total > 0 else 0
    resultados.append({'id': img_id, 'area_cubierta': area_cubierta, 'porcentaje': porcentaje_cobertura})

# Ordenar por porcentaje de cobertura descendente
resultados_ordenados = sorted(resultados, key=lambda x: x['porcentaje'], reverse=True)

# Mostrar resultados
for r in resultados_ordenados:
    print(f"Imagen: {r['id']}, Área cubierta: {r['area_cubierta']:.2f} m², Porcentaje: {r['porcentaje']:.2f}%")
```

[ ] Python

Figura 13: Calculo de porcentaje de cobertura de todas las imágenes de la colección vs la zona de estudió

Al ejecutar el anterior código muestra que las imágenes en el periodo de tiempo seleccionado cuentan con una cobertura del 99% lo que permite hacer un análisis de la serie de tiempo sin contrar con problemas de falta de datos en alguna zona.

```
Imagen: LC08_008057_20231212, Área cubierta: 214860022.72 m², Porcentaje: 99.54%
Imagen: LC08_008057_20230619, Área cubierta: 214815969.11 m², Porcentaje: 99.52%
Imagen: LC08_008057_20230518, Área cubierta: 214730601.59 m², Porcentaje: 99.48%
Imagen: LC08_008057_20231009, Área cubierta: 214724621.16 m², Porcentaje: 99.48%
Imagen: LC08_008057_20230211, Área cubierta: 214643470.36 m², Porcentaje: 99.44%
Imagen: LC08_008057_20230603, Área cubierta: 214627475.45 m², Porcentaje: 99.44%
Imagen: LC08_008057_20230923, Área cubierta: 214595299.23 m², Porcentaje: 99.42%
Imagen: LC08_008057_20231126, Área cubierta: 213976055.32 m², Porcentaje: 99.14%
Imagen: LC08_008057_20230227, Área cubierta: 213727059.90 m², Porcentaje: 99.02%
```

Figura 14: imágenes disponibles en la colección y su respectiva área en m2 y su porcentaje de cobertura en el área de estudio

Teniendo las imágenes, los polígonos de Voronoi se procede a realizar un cálculo de las estadísticas zonales temporales a través del siguiente código:

Calculo NDVI de cada imagen

```python
imagenes = aoi_L8col.toList(aoi_L8col.size())
n_imagenes = aoi_L8col.size().getInfo()

resultados = []

for i in range(n_imagenes):
    img = ee.Image(imagenes.get(i))
    img_id = img.id().getInfo()
    fecha = ee.Date(img.get('system:time_start')).format('YYYY-MM-dd').getInfo()

    ndvi = img.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI')

    stats = ndvi.reduceRegions(
        collection=Poligons,
        reducer=ee.Reducer.mean(),
        scale=30
    ).getInfo()

    for feature in stats['features']:
        props = feature['properties']
        props['imagen_id'] = img_id
        props['date'] = fecha
        resultados.append(props)
```

Python

Figura 15: Calculo del NDVI y del promedio dentro de cada uno de los polígonos de Voronoi

Teniendo en cuenta que el anteior codigo lo que genera es un informacion alfanumerica que se agrega en un arreglo, se procede a transformalo en un dataframe para poder realizar procesos de analisis a traves de graficos.

Guardar calculos del NDVI en un df

```python
df = pd.DataFrame(resultados)
df = df.rename(columns={'mean': 'NDVI_mean'})
df['date'] = pd.to_datetime(df['date'])
```

Python

Figura 16: Almacenar los resultados de las estadísticas zonales para cada uno de los polígonos

Como ya se encuentran los resultados en un dataframe se pueden realizar gráficos temporales a través de la librería de Matplotlib en la cual en el eje x se plasma la variable temporal y en el eje y se pone las estadísticas zonales de cada uno de los polígonos, esto se logra a través del siguiente fragmento de código:

Grafico NDVI en series de tiempo

```python
plt.figure(figsize=(12, 6))

# Agrupación por el ID de cada polígono
for key, grp in df.groupby('rand_point_id'):
    plt.plot(grp['date'], grp['NDVI_mean'], label=str(key))

plt.title('NDVI promedio por polígono (Landsat 8)')
plt.xlabel('Fecha')
plt.ylabel('NDVI promedio')
plt.ylim(0, 1)
plt.legend(title='ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Python

Figura 17: Código para imprimir el grafico de estadísticas zonales del NDVI en una serie de tiempo

Antes de analizar los resultados que se generan a partir del anterior código es necesario entender que lo significan estos resultados, según (Richards, 1999), Los valores de NDVI proporcionan una medida cuantitativa del vigor y la cobertura de la vegetación que se pueden interpretar de la siguiente manera:

- Valores entre 0.6 a 1.0: Indican vegetación muy densa, sana y vigorosa, como selvas tropicales o cultivos en su pico de crecimiento.

- Valores entre 0.3 y 0.6: Representan áreas con cobertura vegetal moderada a escasa, como matorrales, pastizales o cultivos en etapas intermedias de crecimiento.

- Valores cercanos a 0 y 0.3 Corresponden a áreas con muy poca vegetación, etapas tempranas de cultivo, suelo desnudo, roca o áreas urbanas.

- Valores negativos (por ejemplo, -1 a 0): Suelen asociarse con cuerpos de agua, nieve, hielo o nubes.

Teniendo en cuenta esto, los resultados del cálculo del promedio del NDVI de los 25 polígonos de Voronoi generados a partir de puntos aleatorios presenta el siguiente resultado:

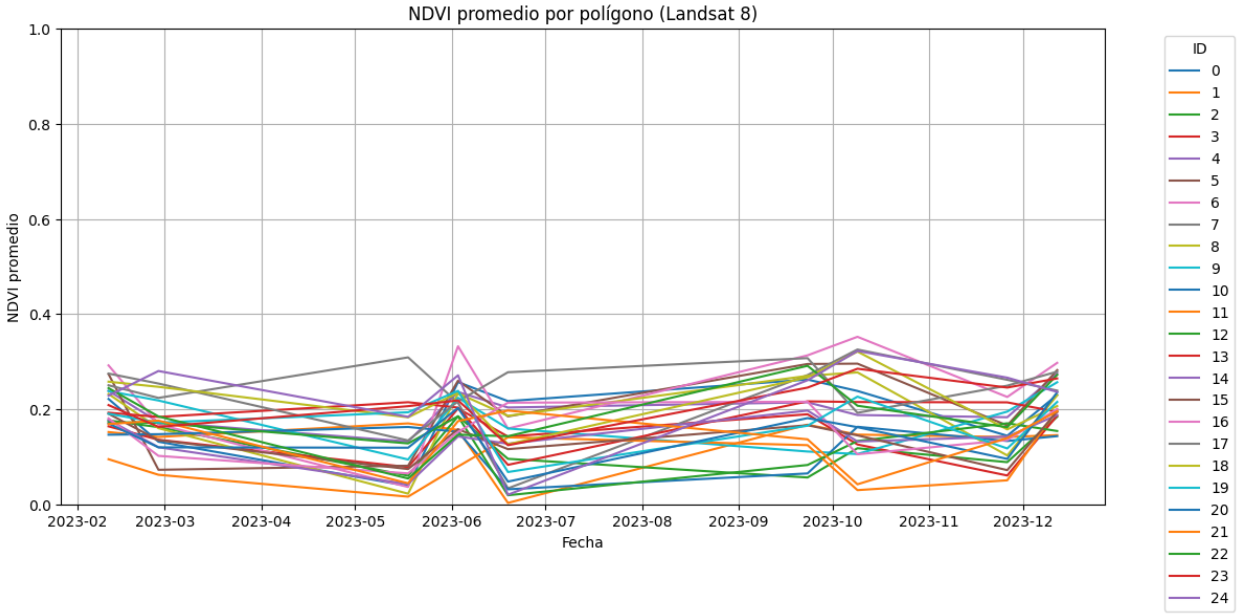

Figura 18: Estadísticas zonales del NDVI en una serie de tiempo (polígonos aleatorios)

Este gráfico no tiene mucho sentido desde el punto de vista del análisis, ya que representa el NDVI promedio por polígonos que fueron seleccionados aleatoriamente. Dado que estos polígonos contienen una mezcla de diferentes tipos de coberturas, vegetación, suelo desnudo, cuerpos de agua, infraestructura, el valor promedio del NDVI no refleja una señal clara o coherente relacionada con algún tipo de cobertura.

Analizar un promedio de NDVI sobre una mezcla heterogénea de coberturas puede generar interpretaciones erróneas, ya que:

- La señal espectral se diluye entre coberturas por los comportamientos espectrales distintos.

- No se pueden extraer conclusiones claras sobre la evolución temporal de una cobertura específica.

- Las variaciones observadas pueden deberse a cambios en la proporción de coberturas dentro de cada polígono y no a cambios reales en la vegetación.

4. NDVI Sentinel-2 (polígonos clusterizados a partir de K-means)[8]

Para este ejercicio se realizo un proceso similar al anterior del calculo del NDVI, pero en este ejercicio se va a realizar a través de una colección de imágenes Sentinel2 lo que hace que se tengan unos ligeros cambios en especial en términos de la extracción y construcción del índice por diferentes motivos, uno de estos es la cantidad de bandas capturadas por el sensor, mientras que en la imagen Landsat se realizaba a partir de las diferencia normalizada de las bandas5 y 4 en este sensor se debe utilizar las bandas 8 y 4. Otro aspecto importante es que en la imagen se filtraron imágenes que cuentan con un porcentaje de nubes menor al 50% de la imagen disminuyendo significativamente la probabilidad que en la zona de estudio no se puedan realizar análisis. Lo anterior se hizo a través del siguiente código:

```
# Colección Sentinel 2
Sen2Collection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
    .filterBounds(zona_estudio) \
    .filterDate('2023-01-01', '2023-12-31')\
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50)) \
    .select(['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B9', 'B11', 'B12'])

# Recortar imágenes
def recortar(img):
    return img.clip(zona_estudio)

aoi_Sen2Collection = Sen2Collection.map(recortar)

# Identificar imágenes y calcular área cubierta
imagenes = aoi_Sen2Collection.toList(aoi_Sen2Collection.size())
area_total = zona_estudio.geometry().area().getInfo()
```

Figura 19: Código de Python para la extracción de imágenes Sentinel y corte en zona de estudio

Al igual que en el ejercicio anterior se desea verificar analizar el comportamiento del promedio del NDVI en la zona de interés, por lo que a través de una función similar a la del anterior ejercicio se revisa la cobertura de las imágenes frente a la zona de estudio con la diferencia que se ordenan de manera descendente de acuerdo a la cobertura, esto se realiza porque en este ejercicio se pretenden usar polígonos segmentados pero se pretende hacer a través de un proceso de clusterizacion de kmeans, por lo que se desea identificar la imagen con mayor cobertura. El principal cambio dentro de este segmento de código es la inclusión de la línea de codigo:

resultados_ordenados = sorted(resultados, key=lambda x: x['porcentaje'], reverse=True)

[8] Repositorio Github: GEE-Remote-Sensing/Notebooks/3_NDVI_Sentinel_calculation.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

```python
# Identificar imágenes y calcular área cubierta
imagenes = aoi_Sen2Collection.toList(aoi_Sen2Collection.size())
area_total = zona_estudio.geometry().area().getInfo()

resultados = []

for i in range(aoi_Sen2Collection.size().getInfo()):
    img = ee.Image(imagenes.get(i))
    img_id = img.id().getInfo()
    mascara_valida = img.select(0).mask().gt(0)
    pix_area = ee.Image.pixelArea().updateMask(mascara_valida)
    area_cubierta = pix_area.reduceRegion(
        reducer=ee.Reducer.sum(),
        geometry=zona_estudio.geometry(),
        scale=10,
        maxPixels=1e10
    ).get('area').getInfo()
    porcentaje_cobertura = (area_cubierta / area_total) * 100 if area_total > 0 else 0
    resultados.append({'id': img_id, 'area_cubierta': area_cubierta, 'porcentaje': porcentaje_cobertura})

# Ordenar por porcentaje de cobertura descendente
resultados_ordenados = sorted(resultados, key=lambda x: x['porcentaje'], reverse=True)

# Mostrar resultados
for r in resultados_ordenados:
    print(f"Imagen: {r['id']}, Área cubierta: {r['area_cubierta']:.2f} m², Porcentaje: {r['porcentaje']:.2f}%")
```

Figura 20: Extracción de imágenes sentinel2 con su respectiva cobertura de nubes

Una vez ejecutado el listado de las imágenes Sentinel2 que se intersecan con la zona de estudio se identifico que se debe realizar una limpieza ya que no todas las imágenes tienen una cobertura aceptable para realizar un análisis significativo.

```
Imagen: 20230128T152639_20230128T152835_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230202T152641_20230202T152642_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230207T152639_20230207T152901_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230403T152641_20230403T152638_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230518T152639_20230518T152642_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230826T152639_20230826T153029_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230503T152641_20230503T153004_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230831T152641_20230831T152943_T18NWK, Área cubierta: 207435705.10 m², Porcentaje: 96.11%
Imagen: 20230826T152639_20230826T153029_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230128T152639_20230128T152835_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230202T152641_20230202T152642_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230227T152639_20230227T152640_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230403T152641_20230403T152638_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230503T152641_20230503T153004_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230518T152639_20230518T152642_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230612T152641_20230612T152642_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230925T152639_20230925T152746_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230930T152641_20230930T152640_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20231129T152641_20231129T152923_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20231229T152651_20231229T152928_T18NWL, Área cubierta: 64947578.19 m², Porcentaje: 30.09%
Imagen: 20230128T152639_20230128T152835_T18NXL, Área cubierta: 25082912.19 m², Porcentaje: 11.62%
Imagen: 20230202T152641_20230202T152642_T18NXL, Área cubierta: 25082912.19 m², Porcentaje: 11.62%
Imagen: 20230217T152639_20230217T152916_T18NXL, Área cubierta: 25082912.19 m², Porcentaje: 11.62%
Imagen: 20230319T152639_20230319T153015_T18NXL, Área cubierta: 25082912.19 m², Porcentaje: 11.62%
Imagen: 20230413T152641_20230413T153029_T18NXL, Área cubierta: 25082912.19 m², Porcentaje: 11.62%
...
Imagen: 20230202T152641_20230202T152642_T18NXK, Área cubierta: 24138550.93 m², Porcentaje: 11.18%
Imagen: 20230413T152641_20230413T153029_T18NXK, Área cubierta: 24138550.93 m², Porcentaje: 11.18%
Imagen: 20230930T152641_20230930T152640_T18NXK, Área cubierta: 24138550.93 m², Porcentaje: 11.18%
Imagen: 20231005T152639_20231005T152742_T18NXK, Área cubierta: 24138550.93 m², Porcentaje: 11.18%
```

Figura 21:Imágenes que se encuentran disponible en la zona de estudio y su respectivo porcentaje de cubrimiento de la zona

Teniendo en cuenta el objetivo del analisis y viendo que a pesar de que se haya creado una funcion para realizar mosaicos de imágenes que contengan la misma fecha la cobertura de estos no eran lo suficientemente altos como para realizar un analisis de estadisiticas zonales principalmente porque estos vacios de informacion generan ruido, se pueden usar pero representaria realizar funciones que se adapten a estos vacios de informacion representando un cambio algo complejo.

```
# Filtrar imágenes con más del 95% de cobertura
imagenes_filtradas = [r for r in resultados if r['porcentaje'] >= 95]
ids_filtradas = [r['id'] for r in imagenes_filtradas]

print("Imágenes con más del 95% de cobertura:")
for r in imagenes_filtradas:
    print(f"Imagen: {r['id']}, Porcentaje: {r['porcentaje']:.2f}%")
```

```
Imágenes con más del 95% de cobertura:
Imagen: 20230128T152639_20230128T152835_T18NWK, Porcentaje: 96.11%
Imagen: 20230202T152641_20230202T152642_T18NWK, Porcentaje: 96.11%
Imagen: 20230207T152639_20230207T152901_T18NWK, Porcentaje: 96.11%
Imagen: 20230403T152641_20230403T152638_T18NWK, Porcentaje: 96.11%
Imagen: 20230503T152641_20230503T153004_T18NWK, Porcentaje: 96.11%
Imagen: 20230518T152639_20230518T152642_T18NWK, Porcentaje: 96.11%
Imagen: 20230826T152639_20230826T153029_T18NWK, Porcentaje: 96.11%
Imagen: 20230831T152641_20230831T152943_T18NWK, Porcentaje: 96.11%
```

Figura 22: Imágenes disponibles que cubren más del 95% de la zona de estudio

En la siguiente sección se crea una colección de imágenes a partir de una lista de imágenes que cumplan la condición de cobertura superior al 95%

```
[6]
# Obtener la lista de imágenes de la colección recortada
imagenes_ee = aoi_Sen2Collection.toList(aoi_Sen2Collection.size())

# Crear una lista de imágenes filtradas usando los índices de las imágenes con >95% cobertura
imagenes_filtradas_ee = [ee.Image(imagenes_ee.get(i)) for i, r in enumerate(resultados) if r['porcentaje'] >= 95]
```

Figura 23: Filtro de la colección de imágenes

El siguientesiguiente paso es mostrar la colección de imágenes con las que se va a trabajar, asegurándose de que cada una cumpla con la condición de cobertura superior al 95%. A continuación, se presenta la información detallada de la primera imagen de la colección:

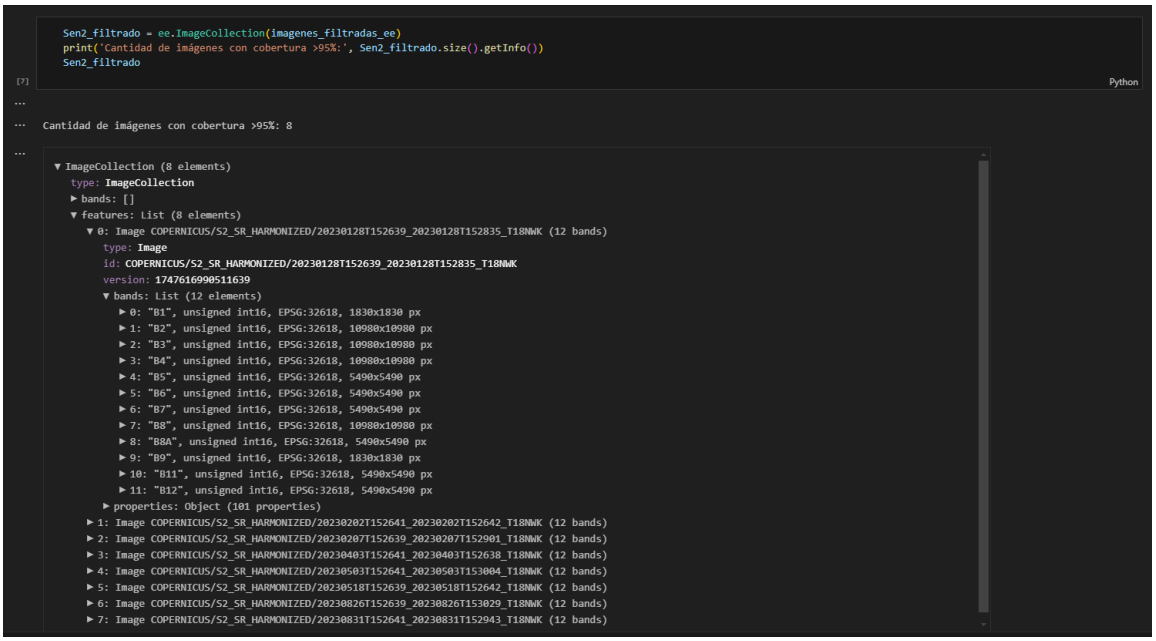

Figura 24: Detalle de colección de imágenes filtradas

Teniendo en cuenta que se va a hacer un análisis de vegetación, y el área es tan grande se realizó un clustering a partir de K-means el cual es un proceso computacional que parte de la clasificación no supervisada el cual consiste en agrupar píxeles de una imagen en función de su similitud espectral. Cada grupo o clúster representa una posible clase espectral en el espacio de características (por ejemplo, tipos de cubierta terrestre con firmas espectrales similares). (Bandyopadhyay & Saha, 2013)

Lo anteriormente mencionado se realizó a través del siguiente código a través de la siguiente manera:

Primero, selecciona la primera imagen de una colección previamente filtrada (Sen2_filtrado) y elige un conjunto específico de bandas espectrales (['B1', 'B2', ..., 'B12']) de esta imagen para el análisis. Luego, define los parámetros para el algoritmo K-means, como el número de clústeres (8) y la cantidad de píxeles para el entrenamiento (5000). Posteriormente, crea un conjunto de datos de entrenamiento muestreando aleatoriamente píxeles de la imagen seleccionada dentro de una zona de estudio definida. Finalmente, entrena el clasificador K-means con estos datos de entrenamiento y lo aplica a la imagen (img_clasif) para generar un resultado donde cada píxel es asignado a uno de los clústeres definidos, agrupando así las áreas con características espectrales similares.

K-means para estadisticas zonales

```
# Seleccion de la primera imagen de la colección filtrada
primera_img = ee.Image(Sen2_filtrado.first())

bandas = ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B9', 'B11', 'B12']

img_clasif = primera_img.select(bandas)

# Parámetros de K-means
num_clusters = 8
num_pixels = 5000
seed = 0

# Crear muestra de entrenamiento
training = img_clasif.sample(
    region=zona_estudio.geometry(),
    scale=10,
    numPixels=num_pixels,
    seed=seed,
    geometries=True
)

# Entrenar el clusterizador K-means
clusterer = ee.Clusterer.wekaKMeans(num_clusters, seed).train(training)
resultado = img_clasif.cluster(clusterer)
```

Figura 25: Clusterización de las coberturas a partir de K-means

Al igual que en los ejercicios previos se realiza un código para visualizar los resultados de la clasificación K-means y la imagen original utilizando la biblioteca geemap. En este caso, se mostrará:

- La imagen original (primera_img), mostrada en una composición de color verdadero (bandas B4, B3, B2).
- El resultado de la clasificación K-means (resultado), donde cada clúster se visualiza con un color diferente según la paleta definida.
- El contorno de la localidad de Usme (zona_estudio) para generar una referencia espacial.

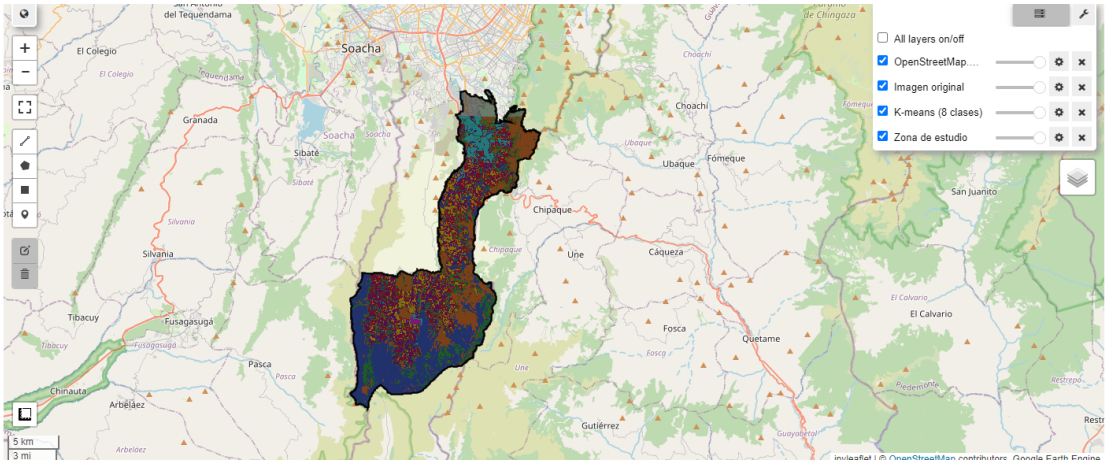

Figura 26: Visualización Clústeres generados a partir del K-means

Al igual que en el ejercicio anterior, se calculan las estadísticas zonales de la colección de imágenes mediante un bucle "for", utilizando como referencia los polígonos generados a partir del K-means. Estos datos se almacenan en un dataframe para su posterior validación.

```
#Calcular NDVI y estadísticas por cluster
for i in range(n_imagenes):
    img = ee.Image(imagenes.get(i))
    img_id = img.id().getInfo()
    fecha = ee.Date(img.get('system:time_start')).format('YYYY-MM-dd').getInfo()

    ndvi = img.normalizedDifference(['B8', 'B4']).rename('NDVI')
    ndvi_cluster = ndvi.addBands(resultado.rename('cluster'))

    stats = ndvi_cluster.reduceRegion(
        reducer=ee.Reducer.mean().group(groupField=1, groupName='cluster'),
        geometry=zona_estudio.geometry(),
        scale=10,
        maxPixels=1e13
    ).getInfo()

    for group in stats['groups']:
        resultados.append({
            'imagen_id': img_id,
            'date': fecha,
            'cluster': group['cluster'],
            'NDVI_mean': group['mean']
        })

df = pd.DataFrame(resultados)
df = df.rename(columns={'mean': 'NDVI_mean'})
df['date'] = pd.to_datetime(df['date'])
```

Figura 27: Calculo de estadísticas zonales de los clústeres generados y almacenamiento de la información en un dataframe

Grafico NDVI en series de tiempo

```
plt.figure(figsize=(12, 6))
for key, grp in df.groupby('cluster'):
    plt.plot(grp['date'], grp['NDVI_mean'], label=f'Clase {key}')
plt.title('NDVI promedio por cluster (Sentinel-2)')
plt.xlabel('Fecha')
plt.ylabel('NDVI promedio')
plt.ylim(0, 1)
plt.legend(title='Cluster', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figura 28: Plotear grafico de series de tiempo con imágenes Sentinel 2

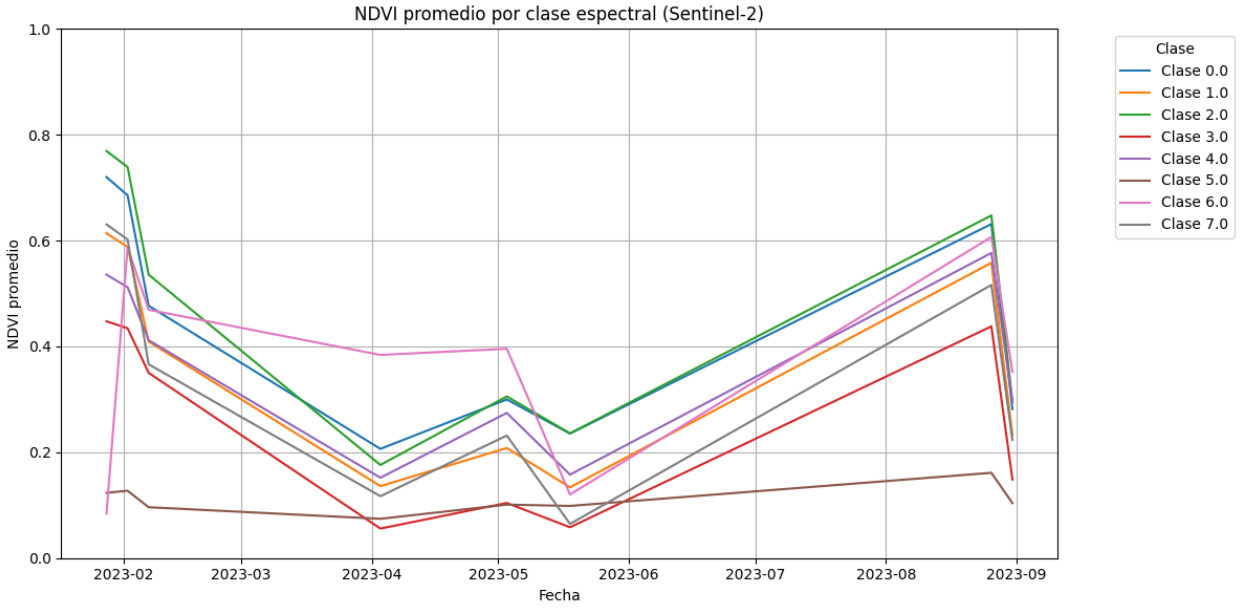

Figura 29: Estadísticas zonales del NDVI en una serie de tiempo (clústeres generados por K-means)

No se explicará el índice de vegetación NDVI en detalle ya que se realizó en el ejercicio pasado. Sin embargo, se intentará proporcionar una explicación detallada de cada clase obtenida a partir de los clústeres generados por el método K-means.

- Clase 5.0 (Marrón)

Presenta consistentemente los valores de NDVI más bajos durante todo el período, manteniéndose mayormente por debajo de 0.15. Adicionalmente muestra muy poca variación, con un ligero aumento hacia el final del periodo. Una posible interpretación: Suelo desnudo, áreas urbanizadas, cuerpos de agua o roca. Estas superficies tienen muy baja reflectancia en el infrarrojo cercano.

- Clase 4.0 (Morado)

Los valores del NDVI son bajos, entre aproximadamente 0.1 y 0.3. Muestra una leve tendencia al alza desde mayo hasta agosto. Una posible interpretación: Vegetación muy escasa, pastizales secos o degradados, o quizás vegetación en etapas muy tempranas de crecimiento.

- Clase 3.0 (Rojo):

Comienza con un NDVI moderado (~0.45), cae drásticamente a valores muy bajos (~0.05) en abril-mayo, y luego se recupera gradualmente hasta ~0.45 en agosto.

Esta gran amplitud sugiere un cambio significativo en la cobertura o vigor de la vegetación. Podría ser un cultivo agrícola con un ciclo de siembra, crecimiento rápido y cosecha, o vegetación que se seca completamente en una temporada y rebrota. El mínimo tan bajo sugiere suelo expuesto.

- Clase 6.0 (Rosa)

Comienza con un NDVI moderado-alto y muestra una curva relativamente más plana comparada con otras, con un mínimo menos pronunciado y un pico alrededor de 0.6. Podría representar vegetación perenne o algún tipo de bosque mixto que mantiene una cobertura vegetal relativamente constante, aunque con fluctuaciones estacionales menos marcadas.

- Clase 7.0 (Gris Claro)

Similar a la Clase 6.0, comienza con NDVI moderado-alto tiene un comportamiento con valles y picos, alcanzando su máximo en agosto. Su comportamiento es un poco más errático que la Clase 6.0. Posible interpretación: Algún tipo de vegetación mixta o matorral denso con una fenología particular.

- Clase 2.0 (Verde)

Comienza como la de mayor NDVI en febrero, sufre la caída más pronunciada hasta ~0.15 en abril-mayo, y luego tiene la recuperación más vigorosa, alcanzando el pico más alto de todas las clases (~0.65) en agosto.

- Clase 0.0 (Azul) y Clase 1.0 (Naranja)

Siguen un patrón muy similar a la Clase 2.0, pero con valores de NDVI ligeramente inferiores en sus picos y mínimos. La Clase 0.0 se mantiene ligeramente por encima de la 1.0 durante la mayor parte del tiempo. Posible interpretación: Estas clases probablemente representan tipos de vegetación con una estacionalidad muy marcada. Podrían ser cultivos de ciclo largo o pastizales productivos. La Clase 2.0 podría ser la vegetación más vigorosa de este grupo.

5. Comparativa respuesta de 3 índices espectrales[9]

Es importante destacar que el NDVI no es el único índice de vegetación disponible para el análisis de la cobertura vegetal. Existen otros índices, como el Índice de Vegetación Mejorado (EVI) y el Índice de Vegetación Verde Normalizado (GNDVI), que proporcionan ventajas adicionales y complementarias en el monitoreo de la vegetación. El EVI, por ejemplo, fue desarrollado para superar algunas limitaciones del NDVI, especialmente en áreas con alta densidad de biomasa. Este índice optimiza la señal de la vegetación y reduciendo las interferencias atmosféricas, lo cual resulta en una mayor precisión en la evaluación de regiones con vegetación densa. Por otro lado, el GNDVI se centra en la reflectancia de la vegetación verde, ofreciendo información específica sobre la salud y vigor de las plantas (Richards, 1999). La manera en la cual se calculan estos índices se pueden visualizar a través de las ecuaciones mostradas a continuación

Index
$\text{NDVI} = \frac{\rho\text{NIR} - \rho\text{RED}}{\rho\text{NIR} + \rho\text{RED}}$
$\text{EVI} = \text{G} \times \frac{\rho\text{NIR} - \rho\text{RED}}{\rho\text{NIR} + \text{C1} \times \rho\text{RED} - \text{C2} \times \rho\text{Blue} + \text{L}} (1 + \text{L})$
$\text{SAVI} = \frac{(\rho\text{NIR} - \rho\text{Red})(1 + \text{L})}{\rho\text{NIR} + \rho\text{Red} + \text{L}}$
$\text{GNDVI} = \frac{\rho NIR - \rho VERDE}{\rho NIR + \rho VERDE}$
$\text{SR} = \frac{\rho\text{NIR}}{\rho\text{RED}}$
$\text{MSI} = \frac{\rho\text{SWIR1}}{\rho NIR}$
$\text{NDWI} = \frac{\rho\text{NIR} - \rho\text{SWIR1}}{\rho\text{NIR} + \rho\text{SWIR1}}$

Ecuación 2: Cálculos para diferentes índices de vegetación

[9] Repositorio Github: GEE-Remote-Sensing/Notebooks/4_Three_vegetation_indices.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

Los valores de EVI suelen oscilar entre -1 y +1, de forma similar al NDVI. Sin embargo, la interpretación de los valores de EVI puede ser ligeramente diferente :

- -1 a 0: Áreas no vegetadas o agua.
- 0 a 0.2: Suelo desnudo o vegetación muy escasa.
- 0.2 a 0.4: Pastizales o cultivos dispersos.
- 0.4 a 0.6: Densidad de vegetación moderada.
- 0.6 a 1: Vegetación densa y sana.

Los valores de GNDVI suelen oscilar entre -1 y +1, de forma similar al NDVI. Sin embargo, la interpretación de los valores de GNDVI puede ser ligeramente diferente:

- -1 a 0: Cuerpos de agua, nieve, hielo o nubes.
- 0 a 0.1: Generalmente corresponden a áreas con muy poca vegetación,
- 0.3 y 0.6: Representan áreas con cobertura vegetal moderada a escasa
- 0.6 a 1.0: Indican vegetación muy densa, sana y vigorosa.

Para mejorar la precisión de los análisis de vegetación, se emplearán los polígonos generados en ejercicios anteriores. Esto permitirá una segmentación más efectiva del área de estudio y facilitará la comparación entre los diferentes índices de vegetación.

El siguiente procedimiento aplica un algoritmo de clustering a la primera imagen de una colección de Sentinel-2, con el objetivo de identificar regiones espectralmente similares. Específicamente, se selecciona el clúster 2, tal como se ha hecho en ejercicios previos. Este clúster se convierte en una máscara y posteriormente se vectoriza para obtener un polígono representativo.

A continuación, para cada imagen en la colección, se calculan los índices de vegetación NDVI, GNDVI y EVI2 dentro de este polígono, utilizando las bandas correspondientes escaladas a reflectancia. Se procede a realizar estadísticas zonales, específicamente el valor promedio de estos índices, y se almacenan junto con la fecha y el ID de la imagen. Esto permite un análisis temporal de la vegetación en el área definida por el clúster, de manera similar a ejercicios pasados, pero con un polígono específico de una categoría seleccionada.

```python
# 4. Estadísticas solo sobre el polígono del cluster 2
imagenes = Sen2_filtrado.toList(Sen2_filtrado.size())
n_imagenes = Sen2_filtrado.size().getInfo()
resultados_indices = []

for i in range(n_imagenes):
    img = ee.Image(imagenes.get(i))
    img_id = img.id().getInfo()
    fecha = ee.Date(img.get('system:time_start')).format('YYYY-MM-dd').getInfo()

    # Escala a reflectancia
    nir = img.select('B8').divide(10000)
    red = img.select('B4').divide(10000)
    green = img.select('B3').divide(10000)

    ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI')
    gndvi = nir.subtract(green).divide(nir.add(green)).rename('GNDVI')
    evi2 = nir.subtract(red).multiply(2.5).divide(nir.add(red.multiply(2.4)).add(1)).rename('EVI2')

    indices = ndvi.addBands([gndvi, evi2])
    stats = indices.reduceRegion(
        reducer=ee.Reducer.mean(),
        geometry=cluster2_vector.geometry(),
        scale=10,
        maxPixels=1e13
    ).getInfo()

    resultados_indices.append({
        'imagen_id': img_id,
        'date': fecha,
        'NDVI_mean': stats.get('NDVI'),
        'GNDVI_mean': stats.get('GNDVI'),
        'EVI2_mean': stats.get('EVI2')
    })
```

Figura 30: Calculo de índices de vegetación

```python
# Convertir a DataFrame y graficar
df = pd.DataFrame(resultados_indices)
df['date'] = pd.to_datetime(df['date'])

plt.figure(figsize=(12, 6))
plt.plot(df['date'], df['NDVI_mean'], label='NDVI')
plt.plot(df['date'], df['GNDVI_mean'], label='GNDVI')
plt.plot(df['date'], df['EVI2_mean'], label='EVI2')
plt.title('Índices de vegetación promedio para cluster 2 (Sentinel-2)')
plt.xlabel('Fecha')
plt.ylabel('Valor promedio')
plt.ylim(0, 1)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figura 31: Visualización de series de tiempo de clúster 2

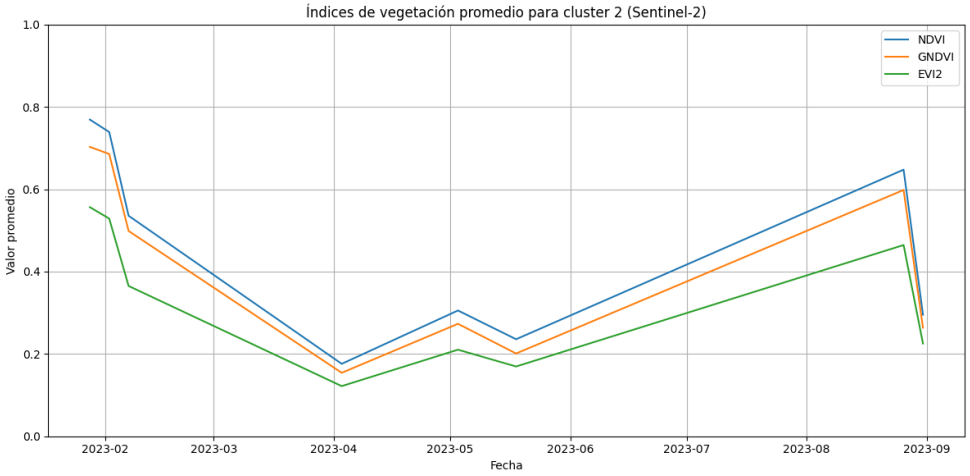

Figura 32: NDVI vs EVI vs GNDVI

El gráfico muestra la evolución temporal de los índices NDVI, GNDVI y EVI2 sobre una misma cobertura vegetal clasificada como clúster 2 a partir de imágenes Sentinel-2 entre febrero y septiembre de 2023. Se observa una disminución progresiva de los valores desde febrero hasta alcanzar un mínimo en abril, seguida de una recuperación constante hasta agosto, lo que sugiere un ciclo fenológico típico de la vegetación, posiblemente asociado a un periodo de cosecha o estrés seguido por un nuevo crecimiento.

6. Extracción de modelos digital de elevación y convertirlos en polígonos[10]

Un Modelo Digital de Elevación (DEM, por sus siglas en inglés) es una representación tridimensional digital que describe la elevación de la superficie terrestre. Este modelo se organiza como una cuadrícula regular, en la cual cada celda contiene un valor numérico que especifica la altitud en una ubicación georreferenciada concreta. Esta estructura permite modelar con precisión el relieve del terreno, incluyendo montañas, valles, llanuras y otras formas geográficas.

Google Earth Engine (GEE) ofrece acceso a una amplia variedad de Modelos Digitales de Elevación (DEM) que permiten analizar y visualizar el relieve terrestre a escala global. Entre estos, destaca la colección 'USGS/GMTED2010_FULL', que proporciona datos de elevación globales a una resolución de 7.5 segundos de arco (aproximadamente 250 metros). Este conjunto de datos fue desarrollado por el Servicio Geológico de los Estados Unidos (USGS) en colaboración con la Agencia Nacional de Inteligencia Geoespacial (NGA) y reemplaza al modelo GTOPO30, ofreciendo mejoras significativas en precisión y cobertura[11]

[10] Repositorio Github: GEE-Remote-Sensing/Notebooks/6_Raster_to_polygons_Elevation.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

[11] Documentación GEE DEM 'USGS/GMTED2010_FULL GMTED2010: Global Multi-resolution Terrain Elevation Data 2010 | Earth Engine Data Catalog | Google for Developers

El siguiente codigo carga los límites de Colombia y un raster de elevación correspondiente al modelo de USGS, se clasifica la elevación en intervalos de 800 metros, recorta los datos a Colombia, y finalmente convierte esas zonas en polígonos vectoriales.

```python
with open(os.path.join(os.getcwd(), '..', 'data', 'geoBoundaries-COL-ADM0.geojson')) as geojson:
    data = json.load(geojson)

colombia = ee.FeatureCollection(data)

# Cargar el raster de elevación y el vector de Colombia
elevation = ee.Image('USGS/GMTED2010_FULL').select('mea').rename('elevation')

#Elegir intervalo de elvación

zones = elevation.divide(750).floor().toInt().rename('zone')

# Enmascara valores menores o iguales 0
zones = zones.updateMask(elevation.gt(0))

# Recorte Colombia
zones_clipped = zones.clip(colombia)

# Convertir raster a polígonos
elevation_vector = zones_clipped.reduceToVectors(
    geometry=colombia,
    crs=elevation.projection(),
    scale=1000,
    geometryType='polygon',
    eightConnected=False,
    labelProperty='zone',
    bestEffort=False,
    maxPixels=1e13,
    tileScale=3
)
```

Figura 33: Extracción DEM para Colombia en GEE

Figura 34: Extracción DEM en Colombia

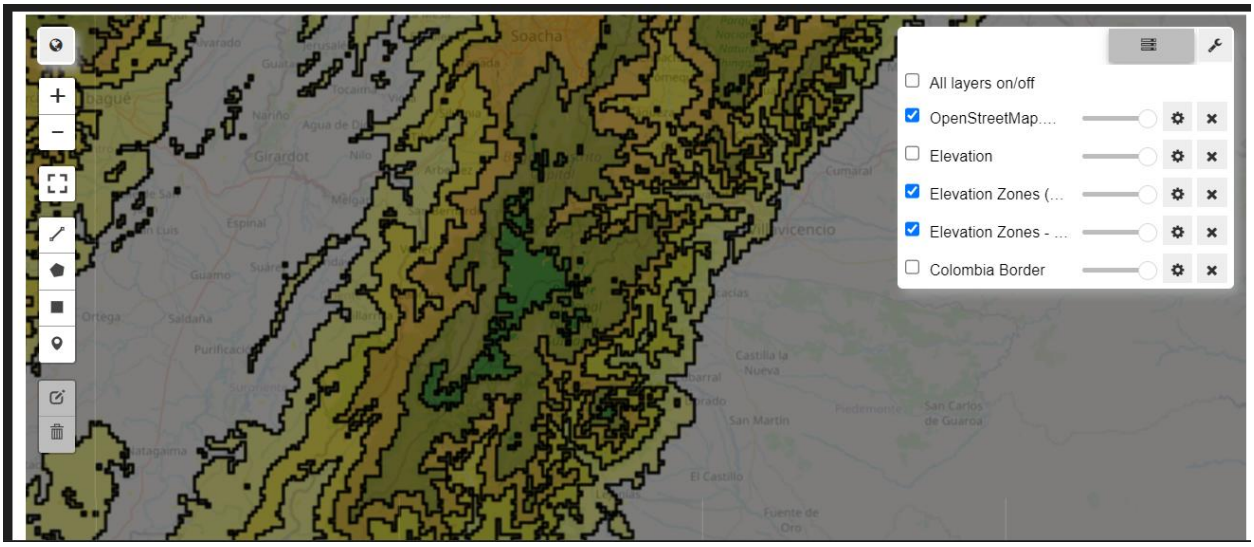

Figura 35: Poligonización de la capa raster proveniente del DEM

7. Detección de cambios en vegetación mediante imágenes Landast[12]

El Índice Normalizado de Quemaduras (NBR) es un índice espectral altamente especializado diseñado específicamente para identificar áreas que han sido afectadas por el fuego y para proporcionar una medida cuantitativa de la severidad de la quemadura. Es particularmente efectivo para delinear y resaltar zonas afectadas por el fuego dentro de paisajes extensos, ofreciendo una demarcación más clara de las áreas quemadas de lo que sería posible con las imágenes de banda visible tradicionales por sí solas. El índice, a veces también denominado Índice Normalizado de Quemaduras (NBI) (Hislop et al., 2018)

La fórmula del NBR está matemáticamente construida para amplificar esta diferencia espectral crítica, proporcionando un valor normalizado que resalta la extensión de la quemadura:

$$\mathrm{NBR} = (\mathrm{NIR} - \mathrm{SWIR}) / (\mathrm{NIR} + \mathrm{SWIR})$$

Ecuación 3: Calculo del NBR

Los números de banda específicos correspondientes a las longitudes de onda NIR y SWIR varían según el sensor satelital. Para Landsat más nuevas, Landsat 8 el NBR se calcula utilizando la Banda 5 para NIR y la Banda 7 para SWIR. La fórmula se convierte en:

$$\mathrm{NBR} = (\mathrm{Banda\ 5} - \mathrm{Banda\ 7}) / (\mathrm{Banda\ 5} + \mathrm{Banda\ 7}).$$

Ecuación 4: Ecuación 3: Calculo del NBR

[12]Repositorio Github: GEE-Remote-Sensing/Notebooks/5_Change_Detection.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

Primero, se autentica el acceso a GEE y se define un punto de interés en el Amazonas Colombiano, para centrar el análisis. Luego, se filtran las colecciones de imágenes Landsat 8 para obtener dos imágenes representativas: con dos temporalidades diferentes, se seleccionan las bandas relevantes para el cálculo de índices espectrales (NBR). Se calcula el índice NBR (Normalized Burn Ratio) para ambas fechas, que es útil para detectar cambios en la vegetación, especialmente los relacionados con incendios.

```
# Crear el mapa
Map = geemap.Map()
point = ee.Geometry.Point([-70.1,-3.4])
Map.centerObject(point, 11)

# Colección Landsat 8 con bandas seleccionadas
landsat8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
    .select(
        ['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7'],
        ['blue', 'green', 'red', 'nir', 'swir1', 'swir2']
    )

# Filtrar imágenes por fecha y nubes
pre_image = landsat8 \
    .filterBounds(point) \
    .filterDate('2015-01-01', '2015-06-30') \
    .sort('CLOUD_COVER') \
    .first()

post_image = landsat8 \
    .filterBounds(point) \
    .filterDate('2020-06-01', '2020-06-30') \
    .sort('CLOUD_COVER') \
    .first()
```

Figura 36: Extracción de imágenes para el NBR

Posteriormente, el código calcula la diferencia entre los valores NBR de ambas fechas para identificar áreas de cambio. Se visualiza este cambio en el mapa usando una paleta de colores personalizada. Además, se clasifica el cambio en tres categorías: ganancia, pérdida o sin cambio, según umbrales definidos para la diferencia de NBR. Finalmente, se visualiza esta clasificación en el mapa, permitiendo identificar fácilmente las zonas afectadas por el evento analizado. Todo el proceso se realiza de manera interactiva en un mapa generado con geemap, facilitando la interpretación espacial de los resultados.

```
# Visualización del cambio NBR
palette = [
    '011959', '0E365E', '1D5561', '3E6C55', '687B3E',
    '9B882E', 'D59448', 'F9A380', 'FDB7BD', 'FACCFA'
]

# Calcular NBR
nbr_pre = pre_image.normalizedDifference(['nir', 'swir2']).rename('nbr_pre')
nbr_post = post_image.normalizedDifference(['nir', 'swir2']).rename('nbr_post')

diff = nbr_post.subtract(nbr_pre).rename('change')

vis_params = {
    'min': -0.2,
    'max': 0.2,
    'palette': palette
}
Map.addLayer(diff, vis_params, 'NBR Change')

# Clasificación según umbrales
threshold_gain = 0.10
threshold_loss = -0.10

diff_classified = ee.Image(0) \
    .where(diff.lte(threshold_loss), 2) \
    .where(diff.gte(threshold_gain), 1)

# Visualización de la clasificación
change_vis = {
    'min': 0,
    'max': 2,
    'palette': ['fcffc8', '2659eb', 'fa1373']
}
Map.addLayer(diff_classified.selfMask(), change_vis, 'Change Classified')

# Mostrar mapa
Map
```

Figura 37: Código para el cálculo y la visualización de NBR

Interpretación de los Valores NBR

Los valores NBR están normalizados y típicamente oscilan entre -1 y 1. La interpretación de estos valores proporciona información directa sobre el estado de la superficie terrestre:

- Valores NBR altos (cercanos a 1): Estos valores son indicativos de vegetación sana y vigorosa. Dichas áreas exhiben una fuerte reflectancia NIR y una débil reflectancia SWIR, características de una biomasa densa y verde.

- Valores NBR bajos (cercanos a -1 o cerca de 0): Estos valores suelen significar suelo desnudo, áreas recientemente quemadas o regiones con vegetación sana mínima o nula. Específicamente, las áreas gravemente quemadas se caracterizan por una baja reflectancia NIR y una reflectancia SWIR relativamente alta. Los valores cercanos o por debajo de 0 suelen ser indicativos de vegetación quemada o improductiva.

- Valores cercanos a cero: A menudo representan áreas no quemadas, cobertura mixta o áreas con vegetación escasa donde las reflectancias NIR y SWIR están más equilibradas. Los cuerpos de agua, las nubes o la nieve también pueden producir valores NBR negativos.

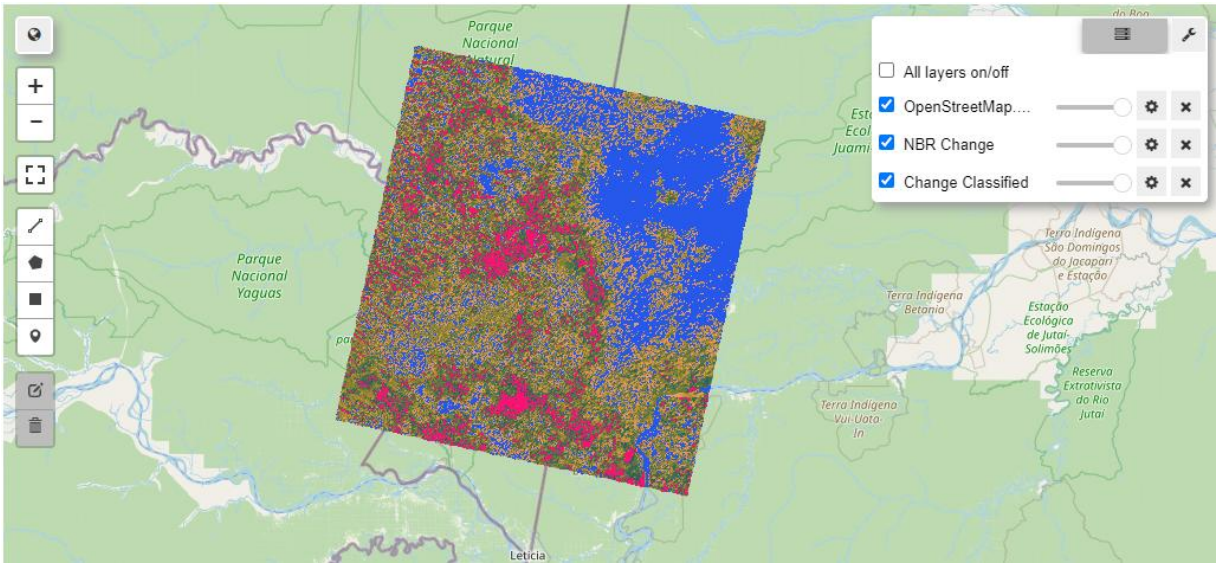

Figura 38: Visualización de resultados NBR

8. Generación de un GIF de imágenes satelitales[13]

La generación y análisis de datos espaciales, junto con su interpretación visual, son herramientas fundamentales para el monitoreo y gestión de las coberturas terrestres. Estos datos permiten una comprensión detallada de cómo se distribuyen y cambian las distintas coberturas del suelo, como bosques, cuerpos de agua, áreas urbanas y agrícolas, a lo largo del tiempo.

La interpretación visual de imágenes satelitales facilita la identificación de patrones y tendencias en el uso del suelo. Por ejemplo, al analizar una serie temporal de imágenes, es posible detectar la expansión urbana, la deforestación o la conversión de tierras agrícolas en áreas residenciales. Esta capacidad de observación es crucial para la planificación territorial, ya que proporciona información actualizada y precisa que respalda la toma de decisiones informadas.

[13] Repositorio Github: GEE-Remote-Sensing/Notebooks/7_generation_of_a_multitemporal_gif.ipynb at main · RicardoMartinezS/GEE-Remote-Sensing

Además, los datos espaciales son esenciales para la conservación ambiental. Permiten monitorear áreas protegidas, evaluar la efectividad de políticas de conservación y detectar actividades ilegales, como la tala no autorizada. La integración de estos datos en sistemas de información geográfica (SIG) y su análisis mediante técnicas de teledetección potencian la capacidad de respuesta ante cambios ambientales y contribuyen a la sostenibilidad de los recursos naturales.

Para poder realizar esto se realizó un código que genera un GIF multitemporal utilizando imágenes satelitales Sentinel-2 para visualizar cambios en una región específica durante el año 2020. Primero, se autentica el acceso a Google Earth Engine (GEE) y se define la zona de estudio mediante un rectángulo geográfico. Luego, se filtra la colección de imágenes Sentinel-2 para esa zona y periodo, descartando aquellas con más de un 10% de nubosidad. A continuación, se crea una función que genera compuestos mensuales, calculando la mediana de las imágenes de cada mes para reducir el efecto de nubes y anomalías.

```python
def create_monthly_composites(start_year, end_year):
    composites = []
    for year in range(start_year, end_year + 1):
        for month in range(1, 13):
            start_date = ee.Date.fromYMD(year, month, 1)
            end_date = start_date.advance(1, 'month')
            filtered = (ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                        .filterBounds(geometry)
                        .filterDate(start_date, end_date)
                        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10)))
            composite = filtered.median().clip(geometry)
            composites.append(composite)
    return ee.ImageCollection(composites)

# Crear la colección de compuestos mensuales para 2020
monthly_collection = create_monthly_composites(2020, 2020)


# Visualizar las imágenes en color natural
def visualize_image(image):
    return image.visualize(bands=['B4', 'B3', 'B2'], min=0, max=3000)

rgb_collection = monthly_collection.map(visualize_image)

# Parámetros para la animación
video_args = {
    'dimensions': 720,
    'region': geometry,
    'framesPerSecond': 2,
    'crs': 'EPSG:4326'
}
```

Figura 39: Código para generar un GIF multitemporal

Después, cada compuesto mensual se visualiza en color natural (bandas rojo, verde y azul) y se agrupan en una colección lista para animación. Se definen los parámetros del video, como el tamaño, la región, la velocidad de los cuadros y el sistema de referencia espacial. Finalmente, se descarga la animación como un archivo GIF en la carpeta especificada y se verifica si el archivo

se guardó correctamente, mostrando el GIF si todo salió bien. Este proceso permite observar de manera dinámica los cambios temporales en la zona de interés a lo largo del año. A continuación, se mostraran algunas imágenes que se encuentran en el GIF.

Figura 40: Imagen en el GIF - Maicao 1

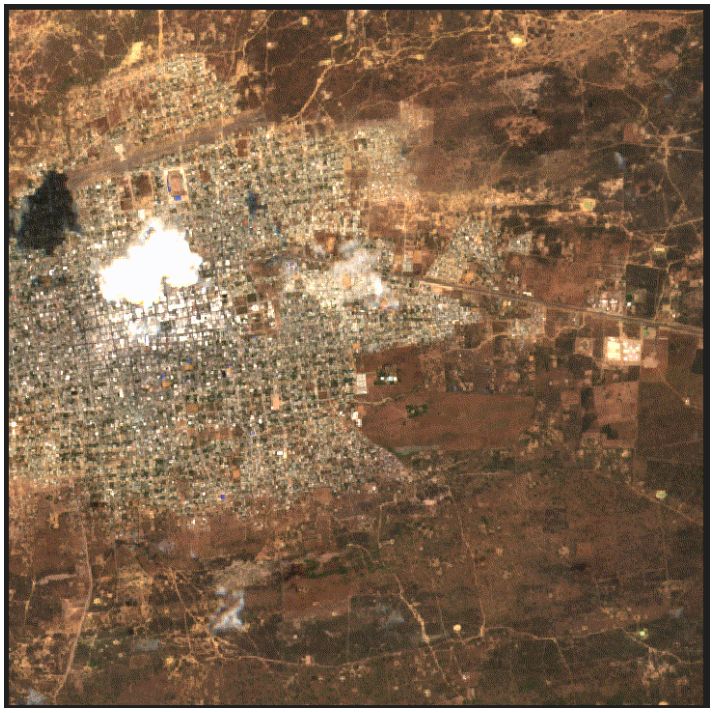

Figura 41: Imagen en el GIF - Maicao 2

Figura 42: Imagen en el GIF - Maicao 3

Conclusión

Google Earth Engine (GEE) se ha consolidado como una herramienta integral y versátil en el campo de la percepción remota. Su infraestructura basada en la nube permite a los usuarios acceder y procesar grandes volúmenes de datos geoespaciales sin necesidad de recursos computacionales locales de alto rendimiento, ofreciendo una notable flexibilidad y adaptabilidad a diversas necesidades. El extenso catálogo de GEE incluye desde imágenes satelitales históricas y actuales hasta modelos digitales de elevación y datos climáticos, lo que facilita abordar una amplia variedad de proyectos, desde el monitoreo ambiental hasta la planificación urbana. Además, la integración de herramientas avanzadas de análisis, como algoritmos de aprendizaje automático y procesamiento de imágenes, permite extraer información significativa y detallada. En síntesis, GEE es una plataforma poderosa y adaptable que maximiza el potencial de la percepción remota, facilitando la toma de decisiones informadas en múltiples disciplinas

Bibliografía

Bandyopadhyay, S., & Saha, S. (2013). Unsupervised classification: Similarity measures, classical and metaheuristic approaches, and applications. In *Unsupervised Classification: Similarity Measures, Classical and Metaheuristic Approaches, and Applications* (Vol. 9783642324512). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32451-2

Banik, A., Das, A. K., Das, S., Maheshwari, A., & Sarvottamananda. (2021). Voronoi game on polygons. *Theoretical Computer Science*, *882*, 125–142. https://doi.org/10.1016/j.tcs.2021.06.023

Chuvieco, E. (1996). *Fundamentos de Teledetección Espacial*. Rialp. https://books.google.co.cr/books?id=HwkQNQAACAAJ

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/J.RSE.2017.06.031

Hislop, S., Jones, S., Soto-Berelov, M., Skidmore, A., Haywood, A., & Nguyen, T. H. (2018). Using landsat spectral indices in time-series to assess wildfire disturbance and recovery. *Remote Sensing*, *10*(3). https://doi.org/10.3390/rs10030460

Joyner Elizabeth. (2025, March). *Access Landsat 8 Surface Reflectance Point Data with AppEEARS.*

Richards, J. A. (1999). *Remote Sensing Digital Image Analysis* (Fifth Edition). https://doi.org/10.1007/978-3-642-30062-2