Previamente asegurarse de tener instaladoas las librerias earthengine-api y folium en el enviroment de trabajo

Cargar las librerías necesarias y activar la plataforma de GEE en la que previamente el usuario deberá haberse registrado.

```
import folium
from folium import plugins
import ee
ee.Authenticate()
ee.Initialize()

# Función de EE predefinida para añadir capas con Folium

def add_ee_layer (self, ee_imagen, parametros, nombre):
    map_id = ee.Image(ee_imagen).getMapId(parametros)
    folium.raster_layers.TileLayer(
        tiles=map_id['tile_fetcher'].url_format,
        attr = 'Alumno: Tecnologías de la Información Geográfica',
        name = nombre,
        overlay = True, # Permite la superposición
        control = True # Permite que aparezca la capa o no en el layer
control
        ).add_to(self)

# Con esto creamos un método propio llamado addLayer que llamara a la
función add_ee_layer
folium.Map.addLayer = add_ee_layer
```

$$dNBR = \left(NBR_{pre} - NBR_{post}\right) * 1000$$

```
# Colección de imágenes de Sentinel con una nubosidad inferior al 20%
y la media calculada de los pixels
sentinel_pre= (ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                  .filterDate('2018-07-03', '2018-08-03')
                  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20)).
mean()) #el filtro de nubosidad varía según imagen

sentinel_post= (ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                  .filterDate('2018-08-15', '2018-09-15')
                  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20)).
mean())

# Funciones de calculo NBR
```

```
def NBR(image):
    NBR = image.expression('float(NIR-SWIR2)/(NIR+SWIR2)',{
        'NIR' : image.select("B8").multiply(0.0001),
        'SWIR2' : image.select("B12").multiply(0.0001),
    })
    return NBR

# Parámetros de visualización de la imágen
vis_NBR ={
    'max': 1,
    'min': -0.1,
    # Paleta de colores standar de GEE
    'palette' : ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163',
'99B718','74A901', '66A000', '529400', '3E8601', '207401', '056201',
               '004C00', '023B01', '012E01', '011D01', '011301']}

# Llamada a las funciones
NBR_pre = NBR(sentinel_pre)
NBR_post = NBR(sentinel_post)


# Exportar imagen a Google Drive. Configuración parámetros de salida
geometry = ee.Geometry.Rectangle([-0.40, 38.9, -0.20, 39.1]) #
Coordenadas geográficas minima y maxima de la imagen a exportar
task_config = {
        'region': geometry,
        'fileFormat': 'GeoTIFF',
        'fileNamePrefix': 'Marti_NBR_post',
        'folder':'gee-datos',
        'scale': 20,      # Resolución en metros del pixel
        'crs': 'EPSG:4326',
        'image': NBR_post,
        'description': 'Imagen procesada NBRpost'
    }
ee.batch.Export.image.toDrive(**task_config).start()

# Crear y añadir elementos al mapa
mapa_NBR = folium.Map(location=[38.9,-0.30], zoom_start = 11) #Las
coordenadas son en Latitud y Longitud
mapa_NBR.addLayer(NBR_pre,vis_NBR, "NBR_pre")
mapa_NBR.addLayer(NBR_post,vis_NBR, "NBR_post")

folium.LayerControl().add_to(mapa_NBR)

mapa_NBR
```

$$dNBR = \left(NBR_{pre} - NBR_{post}\right) * 1000$$

```
# Funciones de calculo dNBR
def dNBR(pre,post):
    dNBR = pre.subtract(post).multiply(1000)
    return dNBR

# Parámetros de visualización de la imágen
vis_dNBR ={
    'max': 1000,
    'min': 100,
    # Paleta de colores standar de GEE
    'palette' : ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163',
'99B718','74A901', '66A000', '529400', '3E8601', '207401', '056201',
               '004C00', '023B01', '012E01', '011D01', '011301']}
# Llamada a la funcion
imagen_dNBR= dNBR(NBR_pre,NBR_post)


mapa_dNBR = folium.Map(location=[39,-0.30], zoom_start = 12)
mapa_dNBR.addLayer(imagen_dNBR,vis_dNBR, "dNBR")
folium.LayerControl().add_to(mapa_dNBR)

mapa_dNBR
```

	Severidad	dNBR
1	No quemado	<100
2	Baja	100-270
3	Media baja	270-440
4	Media alta	440-660
5	Alta	>= 660

Tabla 1: Clasificación de la severidad según el dNBR
Fuente: Key & Benson, 2006

```
import branca
# Ejemplo aplicación de intervalos a imagenes.
sld_intervalos = """
    <RasterSymbolizer>
        <ColorMap type="intervals" extended="false" >
            <ColorMapEntry color="#ffffff" quantity="100" label="Zonas
No quemado"/>
            <ColorMapEntry color="#00ff00" quantity="270"
label="Gravedad baja" />
            <ColorMapEntry color="#007f30" quantity="440"
label="Gravedad Media baja" />
            <ColorMapEntry color="#FFA500" quantity="660"
label="Gravedad Media Alta" />
            <ColorMapEntry color="#ff0000" quantity="1000"
label="Gravedad Alta" />
        </ColorMap>
    </RasterSymbolizer>"""

# Leyenda
```

```
legend_html = """
{% macro html(this, kwargs) %}
<div style="
    position: fixed;
    bottom: 50px;
    left: 50px;
    width: 250px;
    height: 250px;
    z-index:9999;
    font-size:14px;
    ">
    <p><a
style="color:#ffffff;font-size:150%;margin-left:20px;">■</a> Zona
s No quemado
    <p><a
style="color:#00ff00;font-size:150%;margin-left:20px;">■</a> Grav
edad baja
    <p><a
style="color:#007f30;font-size:150%;margin-left:20px;">■</a> Grav
edad media baja
    <p><a
style="color:#FFA500;font-size:150%;margin-left:20px;">■</a> Grav
edad media Alta
    <p><a
style="color:#ff0000;font-size:150%;margin-left:20px;">■</a> Grav
edad Alta
</div>

{% endmacro %}
"""

legend = branca.element.MacroElement()
legend._template = branca.element.Template(legend_html)


# Crear y añadir elementos al mapa
mapa_intervalo = folium.Map(location=[39,-0.30], zoom_start=12)
mapa_intervalo.addLayer(imagen_dNBR.sldStyle(sld_intervalos),None,'dNB
R intervalos')
folium.LayerControl().add_to(mapa_intervalo)
mapa_intervalo.add_child(legend)
mapa_intervalo
```

Este plugin nos permite en una misma ventana Html tener 2 mapas. Crear un mapa Dual con los 2 mapas creados en la Tarea 1 y 2

```
# mapa dual
mapa_dual = folium.plugins.DualMap(location=[38.9,-0.30], tiles=None,
zoom_start=12)
```

```
# Mapas Base
folium.TileLayer('OpenStreetMap').add_to(mapa_dual.m1)
folium.TileLayer('CartoDB Positron').add_to(mapa_dual.m2)

# Añadir capas a visualizar
mapa_dual.m1.addLayer(NBR_pre,vis_NBR, "NBR_pre")
mapa_dual.m2.addLayer(NBR_post,vis_NBR, "NBR_post")

# Añadir control de mapas
folium.LayerControl().add_to(mapa_dual)

# Guardar mapas en web
mapa_dual.save("Medio_Ambiente.html")

mapa_dual
```

Catálogo de recursos GEE

Enlace al catálogo de datos ofical de GEE https://developers.google.com/earth-engine/datasets

```
          <ul type ='1'>
               <li> 1. Fechas Pre: '2023-09-15', '2023-10-15' </li>
               <li> 2. Fechas Post: '2023-11-15', '2023-12-15' </li>
               <li> 3. Zona a Exportar: Punto mínimo: [38.8,-0.4]
Punto máximo [39.0,-0.15] </li>
               <li> 4. Punto Central : [38.9,-0.28]     </li>
          </ul>
```

El método Índice de vegetación ajustado al suelo (SAVI) es un índice de vegetación que intenta minimizar las influencias del brillo del suelo utilizando un factor de corrección de brillo del suelo. Esto con frecuencia se utiliza en regiones áridas en donde la cubierta de vegetación es baja y genera valores entre -1,0 y 1,0, siendo la vegetación sana 0 a 1.
Su creador, Huete, agregó un factor de ajuste del suelo L a la ecuación de NDVI para corregir los efectos del ruido del suelo (color del suelo, humedad del suelo, variabilidad del suelo a través de la región, etc.), que tienden a afectar a los resultados.

L = 0.428

$$EVI = G * \frac{NIR - RED}{NIR + (C1 * RED) - (C2 * BLUE) + L}$$

EVI es un índice que trabaja las bandas rojo y azul del visible para corregir simultaneamente el efecto de la atmósfera y junto al factor L corregir la influencia del suelo. Además del factor L emplea dos parámetros adicionales constantes C (C1 = 6.0, C2 = 7.5) y G = 2.5 Los valores para EVI varían del -1 a 1, siendo la vegetación sana valores entre 0.2 y 0.8

L = 1.0

$$\boldsymbol{EVI} = G * \frac{NIR - RED}{NIR + (C1 * RED) - (C2 * BLUE) + L}$$

Salvar el mapa como: Medio_Ambiente_'Apellidos alumnos'.html