ANEXO I. CÓDIGO

Importación e iniciación de librerías

```
# Importar paquete GEE y GeeMap
import ee
import geemap
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

ee.Authenticate()
ee.Initialize()
```

Cargar puntos de muestreo

```
# Cargamos los datos de muestreo
zona_estudio = ee.FeatureCollection("users/anamari42680/areamuestreo")
puntos = ee.FeatureCollection("users/anamari42680/p2022_2")

# Esta propiedad de la tabla almacena el target (cobertura del suelo).
label = 'TC'

#Creamos el Mapa base
Mapa = geemap.Map(basemap='SATELLITE')
geometria = zona_estudio.geometry()
Mapa.centerObject(geometria,11)
Mapa.addLayer(zona_estudio, {"color" : "red"},name = "Zona de
estudio")
Mapa.addLayer(puntos, {"color" : "green"},name = "Puntos de muestreo")
Mapa
```

Obtención imágenes satélite SENTINEL 2

```
# Función predeterminada para enmascarar las nubes
def maskS2clouds(image):
  qa = image.select('QA60')
  opaque_cloud = 1 << 10
  cirrus_cloud = 1 << 11
  mask = qa.bitwiseAnd(opaque_cloud).eq(0)\
           .And(qa.bitwiseAnd(cirrus_cloud).eq(0))
  clean_image = image.updateMask(mask)
  return clean_image

sen2 = ee.ImageCollection("COPERNICUS/S2_SR")\
        .filterDate('2022-05-08', '2022-05-12')\
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))\
        .map(maskS2clouds)

imagen_sen2 =
sen2.select(['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12'])\
                .mean().divide(10000).clip(geometria)
```

```
# simbología de visualización
viz_sen2 = {
    'bands': ['B4','B3','B2'],
    'min':0.0,
    'max':0.3,
}

# visualizamos la capa
Mapa.addLayer(imagen_sen2, viz_sen2, 'Sentinel2' )


imagen_sen2.bandNames().getInfo()
```

Cálculo de índices

```
# Calcular el NDWI
def calcNDWI(image):
    ndwi = image.expression('float((GREEN - NIR) / (GREEN + NIR))',{
        'NIR': image.select('B8'),
        'GREEN': image.select('B3')})
    return ndwi

NDWI = calcNDWI(imagen_sen2).rename('NDWI')

# Paleta de colores con cadena Hex.
'''palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555',
'FCD163','99B718','74A901', '66A000', '529400', '3E8601', '207401',
'056201','004C00', '023B01', '012E01', '011D01', '011301']'''
palette = ['red','white']

# visualizar la imagen base y las NDVI creado a partir de ella
Mapa.addLayer(NDWI,{'min': 0.2, 'max': 1.0, 'palette': palette},
'NDWI')


# Calcular el NDMI
def calcNDMI(image):
    ndmi = image.expression('float((NIR - SWIR1) / (NIR + SWIR1))',{
        'NIR': image.select('B8'),
        'SWIR1': image.select('B11')})
    return ndmi

NDMI = calcNDMI(imagen_sen2).rename('NDMI')

# Paleta de colores con cadena Hex.
palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
           '74A901', '66A000', '529400', '3E8601', '207401',
           '056201','004C00', '023B01', '012E01', '011D01', '011301']

# visualizar la imagen base y las NDVI creado a partir de ella
Mapa.addLayer(NDMI,{'min': 0, 'max': 1, 'palette': palette}, 'NDMI')
```

Creación de imagen de estudio

```
def maskS2clouds(image):
  qa = image.select('QA60')
  opaque_cloud = 1 << 10
```

```
  cirrus_cloud = 1 << 11
  mask = qa.bitwiseAnd(opaque_cloud).eq(0)\
           .And(qa.bitwiseAnd(cirrus_cloud).eq(0))
  clean_image = image.updateMask(mask)
  return clean_image

sen211 = ee.ImageCollection("COPERNICUS/S2_SR")\
        .filterDate('2022-05-08', '2022-05-12')\
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))\
        .map(maskS2clouds)

imagen_sen211 = sen211.select(['B2','B3','B4','B6','B8','B8A','B12'])\
                .mean().divide(10000).clip(geometria)


imagen = imagen_sen211.addBands([NDMI, NDWI])


bandas=imagen.bandNames().getInfo()
print(bandas)
```

Creación datos de entrenamiento

```
Datos = puntos.randomColumn('random', 2000)

set_datos = imagen.sampleRegions(collection = Datos,
                                 properties = [label,'random'],
                                 scale = 10,
                                 projection = None,
                                 tileScale = 1,
                                 geometries = True)

# Separar datos entrenamiento y validacion
split = 0.7  # Aproximadamente 70% de Entrenamiento, 30% de validacion
training = set_datos.filter(ee.Filter.lt('random', split))
testing = set_datos.filter(ee.Filter.gte('random', split))
```

Parámetro clasificación RandomForest

```
# Crear un clasificador RF con parámetros personalizados.
classifier = ee.Classifier.smileRandomForest(100)

# Entrenar el clasificador
trained = classifier.train(training, label, bandas)

# Clasificar la imagen
classified = imagen.select(bandas).classify(trained)

# simbología de visualización
viz_Clas = {
    'min':1,
    'max':5,
    'palette': ['blue', 'red', 'yellow', 'green', 'white']
}

# visualizamos la capa
Mapa.addLayer(classified, viz_Clas, 'Clasificada' )
```

Parámetro clasificación Arboles de decisión CART

```
Mapa
```

Matriz de confusión y precisión

```
validacion = testing.classify(trained);
errorMatriz = validacion.errorMatrix(label, 'classification')
matriz = errorMatriz.getInfo()
matriz


trainCM = pd.DataFrame(np.asarray(matriz),
                       index=['Eliminar','Agua', 'Carrizo-cañas',
'Cítricos', 'Herbáceas','Suelo sin vegetación'],
                       columns=['Eliminar','Agua', 'Carrizo-cañas',
'Cítricos', 'Herbáceas','Suelo sin vegetación'])
# Eliminar columna por etiqueta
trainCM_1 = trainCM.drop(labels = ["Eliminar"], axis = 1)
trainCM_2 = trainCM_1.drop(labels = ["Eliminar"], axis = 0)
trainCM_2


dataframe = geemap.ee_to_pandas(set_datos)
dataframe.describe()


dataframe.to_csv("Marjal2022.csv")
dataframe.head(5)


dataframe.drop(['random'], axis=1, inplace=True)
correlation=dataframe.corr()
correlation.head(15)


fig, ax = plt.subplots(1, figsize=(6,6))
sns.heatmap(trainCM_2/trainCM_2.sum(axis=1), annot=True)
ax.set_xlabel('Modelo predictivo', fontsize=20)
ax.set_ylabel('Actual', fontsize=20)
plt.title("Matriz de confusión", fontsize=20);
```

Precisión de la clasificación

```
# precisión de la clasificación

print('Exactitud General:', errorMatriz.accuracy().getInfo());
print('Indice Kappa:', errorMatriz.kappa().getInfo());
```

Superficie lámina de agua

```
pixelArea = ee.Image.pixelArea()

mascara_agua = classified.eq(1)  # Devuelve una capa con 0 y 1. Valor
de clase 1 (agua)
```

```
masa_agua = pixelArea.updateMask(mascara_agua)  # Es un extract by
mask

area = masa_agua.reduceRegion(
    reducer = ee.Reducer.sum(),
    geometry = geometria,
    crs = masa_agua.projection(),
    scale = 10,
    maxPixels = 1e9,
)

# visualizamos la capa

Mapa.addLayer(masa_agua, viz_Clas, 'agua' )
Mapa
print(area.get('area').getInfo())
```

Exportar resultados

```
# Exportar shp

# Datos ROI completo
geemap.ee_to_shp(set_datos,
                 filename = 'ROI_completo_modelo.shp')

# Datos ROI entrenamiento
geemap.ee_to_shp(training,
                 filename = 'ROI_entrenamiento_modelo.shp')

# Datos ROI Validacion
geemap.ee_to_shp(testing,
                 filename = 'ROI_validacion_modelo.shp')


# Exportar en google drive. Crear una región
geemap.ee_export_image_to_drive(classified,
                                description="Modelo2022",
                                folder='GEE',
                                region=geometria,
                                scale=10)

Mapa.to_html(filename="TFM_Marjal.html", title='My Map', width='100%',
height='880px')


import os
ruta_archivos = r"C:\Users\anama\Desktop\Universidad\MÁSTER MEMIC
UPV\MARJAL_TFM"
os.chdir(ruta_archivos) # Cambiar ruta trabajo
os.getcwd() # Consultar ruta trabajo
```