MAPBIOMAS
Chaco

Clasificaciones Supervisadas

Contenidos

Incorporación de datos de campo y generación de datos de entrenamiento desde la interfaz de GE. Cálculo de mosaicos de índices del área de estudio. Muestreo a partir de los datos de campo. Separación en training/testing. Algoritmos disponibles para clasificación supervisada. Parametrización de los algoritmos. Ajustes. Validación utilizando matrices de confusión (accuracy, kappa). Aplicación del modelo y mapeo de resultados. Visualizar y exportar los resultados de la clasificación.

Preprocesamiento y generación del conjunto de datos

Algunos de los pasos requeridos para una clasificación ya fueron explicados en tutoriales anteriores.

Estos son:

1. Cargar vector del área de estudio
2. Seleccionar colección de imágenes y filtrar por fecha, nubes, bandas, etc.
3. Reducir colección a imágen (e.g. calcular mediana para cada banda).

Definimos el ImageCollection

Esta ya se la saben :D

Son codigos del Tutorial 1 Seccion 2

```
// 1. area de estudio (de sección anterior)
var geometry = ee.FeatureCollection('ft:1NOdzgdcCiWZ6YcoEh

// 2. Seleccionar colecciones
// Seleccionar producto. Indicar el ImageCollection ID

var producto = ee.ImageCollection('LANDSAT/LC8_L1T_TOA');

// Definir bandas a seleccionar
var bandas = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7'];
```

Aplicamos los filtros

```
// Filtrar colección
var coleccion1 = producto
    // por área de estudio. Debe estar cargada el área en
    .filterBounds(geometry)
    //por rango de fechas
    .filterDate('2016-08-01', '2016-10-31')
    // por cobertura de nubes máxima
    .filterMetadata('CLOUD_COVER','less_than', 40)
    // por bandas (definidas más arriba)
    .select(bandas);

// ver detalles de colección y filtros aplicados
print("Coleccion seleccionada", coleccion1);
```

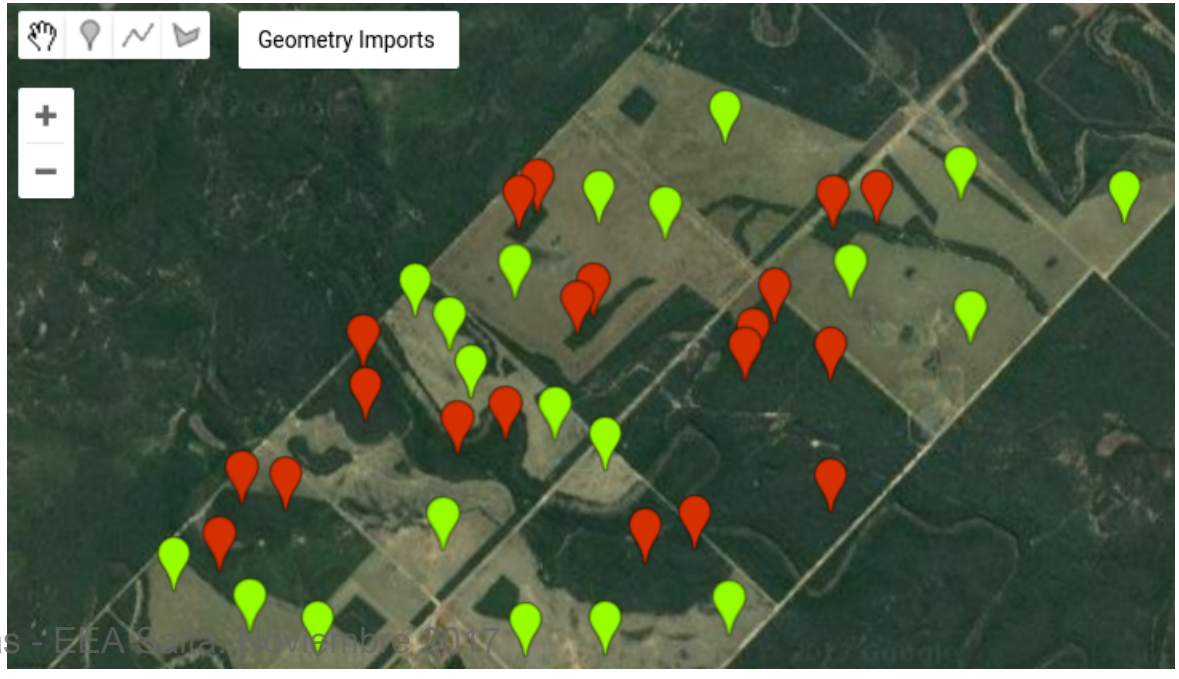
Hacemos el mosaico

```
// 3. Convertir a Imagen. Aplicar reducción de mediana
var stack1 = coleccion1.median();

// Agregar la imagen al mapa con una configuración de
// Falso Color

Map.addLayer (stack1,
        {bands: ['B5', 'B4', 'B3'],
        min: [0,0,0],
        max: [0.6,0.6,0.6] },
        "Landsat 8 B5-B4-B3" );

// Centrar en área de estudio
Map.centerObject(geometry, 8);
```

Cargar o generar puntos (o polígonos) de entrenamiento y validación

Para el punto 4 se pueden generar nuevos puntos, cargarlos a partir de un vector disponible previamente u obtenerlos de puntos ya generados en la plataforma GEE.

https://code.earthengine.google.com/ce97d96621040d67a1ec08ffc0df322c



Ahora vamos a combinar las dos variables **clase0** y **clase1** en un único FeatureCollection:

```
// Unir muestras por clase en un único FeatureCollection
var samples = clase0.merge( clase1 );

// ver caracteristicas de FeatureCollection
print ("muestras", samples);
```

Clasificaciones Supervisadas

Las clasificaciones supervisadas se realizan cuando se tienen definidas las clases y se dispone de información de casos correspondientes a esas clases (información de campo). Los distintos clasificadores se entrenan con la información de campo disponible a partir del comportamiento en las distintas bandas consideradas. La evaluación del resultado de una clasificación requiere disponer de datos independientes a los utilizados para el entrenamiento.

Separación en train/testing

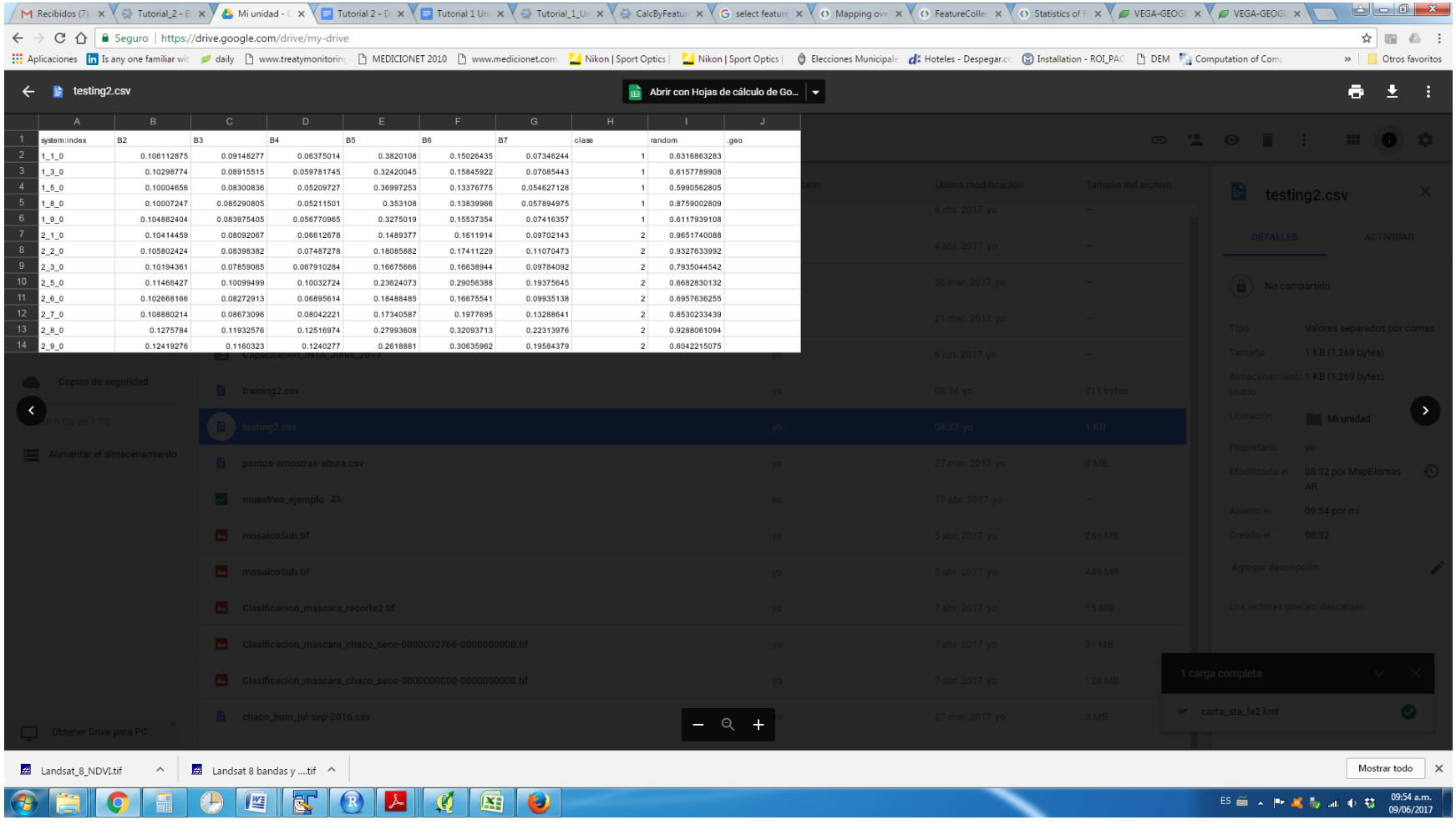
Separación del conjunto de datos para entrenamiento y validación. La plataforma permite generar atributos (llamado “random” en este caso) con números al azar mediante la función randomColumn y agregarlos al FeatureCollection que contiene el set de datos. Genera valores entre 0 y 1. Esta requiere indicar el FeatureCollection y un valor inicial para generar los números al azar (“seed”):

```
// Separacion de set de datos (polígonos) para entrenamien
// generación de atributo con números al azar (columna "ra
// para hacer muestreo

var seed = 2015;
samples = samples.randomColumn('random', seed);
```

Esto nos permitirá hacer un muestreo, seleccionando filas que contengan cierto rango de números generados al azar (Mayores o menores a cierto valor umbral). Para entrenar el algoritmo de clasificación se requiere extraer información de las imágenes para los polígonos seleccionados (similar a la etracción de información visto en secciones anteriores), incluyendo en las salidas los atributos clase y “random”:

```
// extraccion de información incluyendo atributos clase y
var set_datos = stack1.sampleRegions({
  collection: samples,
  properties: ['clase','random'],
  scale: 30
});
```

El resultado de la extracción es una FeatureCollection que contiene información de cada banda y cada polígono:



testing2.csv

	system:index	B2	B3	B4	B5	B6	B7	clase	random	.geo
2	1_1_0	0.106112875	0.09148277	0.06375014	0.3820108	0.15026435	0.07346244	1	0.6316863283	
3	1_3_0	0.10298774	0.08915515	0.059781745	0.32420045	0.15845922	0.07085443	1	0.6157789908	
4	1_5_0	0.10004656	0.08300836	0.05209727	0.36997253	0.13376775	0.054627128	1	0.5990562805	
5	1_8_0	0.10007247	0.085290805	0.05211501	0.353108	0.13839966	0.057894975	1	0.8759002809	
6	1_9_0	0.104882404	0.083975405	0.056770965	0.3275019	0.15537354	0.07416357	1	0.6117939108	
7	2_1_0	0.10414459	0.08092067	0.06612678	0.1489377	0.1611914	0.09702143	2	0.9651740088	
8	2_2_0	0.105802424	0.08398382	0.07487278	0.18085882	0.17411229	0.11070473	2	0.9327633992	
9	2_3_0	0.10194361	0.07859065	0.067910284	0.16675666	0.16638944	0.09784092	2	0.7935044542	
10	2_5_0	0.11466427	0.10099499	0.10032724	0.23624073	0.29056388	0.19375645	2	0.6682830132	
11	2_6_0	0.102668166	0.08272913	0.06895614	0.18488485	0.16675541	0.09935138	2	0.6957636255	
12	2_7_0	0.108880214	0.08673096	0.08042221	0.17340587	0.1977695	0.13288641	2	0.8530233439	
13	2_8_0	0.1275784	0.11932576	0.12516974	0.27993608	0.32093713	0.22313976	2	0.9288061094	
14	2_9_0	0.12419276	0.1160323	0.1240277	0.2618881	0.30635962	0.19584379	2	0.6042215075	

Posteriormente se subdivide el FeatureCollection con el set de datos en Entrenamiento (“training”) y Validación (“testing”). Se selecciona un umbral de separación de los valores al azar generados entre 0 y 1 (atributo “random”). En este caso, se seleccionan los valores mayores o iguales a 0.6 para entrenamiento y los menores a 0.6 para validación. Se pueden ver en consola los nuevos set de datos generados:

```
// Separación entre Entrenamiento y validación. Identifica
var training = set_datos.filterMetadata('random', 'not_les
var testing = set_datos.filterMetadata('random', 'less_tha

print ("Set de datos entrenamiento", training);
print ("Set de datos validación", testing);
```

Entrenamiento

Aquí debemos seleccionar el algoritmo de clasificación, el set de datos de entrenamiento (“training”), el atributo de separación en clases (“clase”) y las bandas seleccionadas. En este caso usamos el algoritmo Random Forest:

```
// Entrenamiento
var bandas_sel = ['B7','B6','B5', 'B4', 'B3'];
var trained = ee.Classifier.randomForest().train(training,
```

Aplicar el modelo

Una vez entrenado el modelo, se lo aplica a una imagen y se genera la clasificación.

```
// clasificación con el modelo entrenado
var classified = stack1.select(bandas)
        .classify(trained)
        .clip(geometry);

Map.addLayer(classified,
        {min:0,
         max:1,
         palette: ['339820', 'e6f0c2']},
         'Clasificacion');

print(classified);
```

Exportar la clasificación a Drive

La clasificación puede ser exportada como imagen GeoTiff

```
// Exportar imagen de clasificación
Export.image.toDrive({
  image:classified,
  description: 'clasificacion',
  scale: 30,
  region: geometry
});
```

Evaluación del modelos

Generación de Matriz de Confusión y resultados. La herramienta permite calcular la matriz de confusión, y estimar exactitud general, de usuario y de productor e indice Kappa.

```
// Generación de matriz de confusión y resultados
var validation = testing.classify(trained);
var errorMatrix = validation.errorMatrix('clase', 'classif

print('Matriz de Confusión:', errorMatrix);
print('Exactitud General:', errorMatrix.accuracy());
print('Indice Kappa:', errorMatrix.kappa());
print('Exactitudes de Usuario:', errorMatrix.consumersAccu
print('Exactitudes de Productor:', errorMatrix.producersAc

// exportar matriz de confusion como csv
var err = ee.FeatureCollection( ee.Feature(null, {
  'matrix': errorMatrix.getInfo()
}));
print (err);
```

Exportar la matriz de confusión

```
Export.table.toDrive({
    'collection': err,
    'description': 'CM_tutorial_2',
    'fileNamePrefix': 'CM_tutorial_2',
    'fileFormat': 'CSV'}
);

// convertir matriz de confusion a array
var cfmt2x2 =  errorMatrix.array().toList().map(
  function(item){
        item = ee.List(item);
        return ee.Feature(null, {Bosque: item.get(0), NoBo

  });

print(cfmt2x2);
```

Ejemplo de clase

Actividad Propuesta

Modifique el script para que funcione con CART y almacene el árbol generado en una nueva variable utilizando la función **explain**.

Luego realice una nueva clasificación con el árbol ya generado y utilizando la funcionalidad provista por ee.Classifier.decisionTree en la región cubierta por el siguiente polígono:

```
var nueva_zona = ee.Geometry.Rectangle([-64.456, -24.417,
```

1. ¿Qué pasa si incremento el número de árboles de un RandomForest?
2. ¿Qué pasa si limito la profundidad en CART?

Muestreo para la clasificación no supervisada

```
// Seleccionamos los pixeles para correr el agrupamiento
var datos_km = stack1.clip(area_estudio).sample({
    region: area_estudio,
    scale: 30,
    numPixels: 5000,
    seed: 10
  });
```

Configuramos el algoritmo wekaXMeans

```
var TOTALK = 10; // Máxima cantidad de clusters
var clasificadorNoSup = ee.Clusterer.wekaXMeans(
  {minClusters: 2,  // Mínima cantidad de clusters
   maxClusters: TOTALK,
   maxIterations: 100,
   distanceFunction: "Euclidean",
   seed: 123});

// Entreno con las muestras
clasificadorNoSup = clasificadorNoSup.train(datos_km);
```

Ejemplo completo

Aquí está el ejemplo completo de la clasificación supervisada y no supervisada.