LABORATÓRIOS DIDÁTICOS DE GEOPROCESSAMENTO

Sensoriamento Remoto

Mosaicos e classificação supervisionada de imagens no GEE

Docente: Prof.ª Dr.ª Mariana Abrantes Giannotti

Elaboração do roteiro: Leonardo Alves Godoy

Objetivos

- Remoção de nuvens
- Apresentação de composições e mosaicos de imagens
- Classificação supervisionada no GEE

1 Introdução

Neste laboratório são apresentados como construir mosaicos a partir de diversas imagens e o uso das implementações no ambiente GEE da classificação supervisionada de imagens.

2 Referências teóricas

2.1 Remoção de nuvens em imagens

A ideia da remoção de áreas cobertas por nuvens (e sombras) é tornar cada pixel identificado como parte de uma nuvem - ou sombras de nuvens - transparente, para o processamento, enquanto as outras áreas são mantidas na imagem. Esse processo é feito através da *Image Masking,* onde é utilizada uma imagem chamada de máscara. Uma máscara (*mask*) é uma imagem binária com valores 0 e 1. A máscara exclui da imagem onde ela é aplicada os *pixels* em que ela contém o valor 0 (inválidos) e mantém o que contém o valor 1 (válidos).

2.2 Mosaicos de imagens

Pré-processamento de imagens são operações que objetivam corrigir imagens degradadas ou distorcidas para criar uma representação mais fidedigna da cena original e melhorar a utilidade da imagem para manipulação posterior. Isso tipicamente envolve o processamento inicial de dados de imagens *raw* para eliminar o ruído presente nos dados, calibrá-los radiométricamente, corrigir distorções geométricas e expandir ou contrair a extensão de uma imagem através da operação de *mosaicking* ou *subsetting*.

2.2.1 *Subsetting* (subdividir), *layer stacking* (empilhar imagens) e *mosaicking* (mosaico)

Alguns passos do pré-processamento das imagens que serão utilizadas podem envolver subdividir as imagens para reduzir o volume de dados - o chamado *subsetting* - o *layer stacking* - que é utilizado para combinar múltiplas bandas separadas ou *layers* em uma única imagem, e o *mosaicking* que é utilizado para juntar diversas imagens que cobrem uma área mais ampla do que apenas uma (LILLESAND; KIEFER; CHIPMAN, 2015, cap. 7).

No GEE, os termos composição e mosaico se referem a operações similares. A ideia dessas operações é juntar imagens diferentes em uma só, seja através de operações de agregação de *pixels*, através de operações matemáticas (composição em imagens sobrepostas na mesma região) ou em imagens que não estão sobrepostas ou que possuem descontinuidades dentro de suas áreas (mosaicos). No caso dos mosaicos, as descontinuidades das imagens são preenchidas na ordem em que as imagens estão na coleção conforme *pixels* não “transparentes” são encontrados na sequência. É como se as imagens estivessem em uma pilha, sendo os vazios da imagem do topo preenchidos pelas imagens abaixo, caso não se preencha pela segunda imagem da pilha, a terceira é utilizada e assim sucessivamente até o total preenchimento de todos os *pixels* possíveis (alguns podem não ter nenhuma representação na coleção). O termo composição, nesse contexto, não deve ser confundido com a composição entre bandas para visualização no mapa, conforme visto nos laboratórios anteriores.

2.3 Classificação supervisionada

Ajustar um modelo de classificação supervisionada pode ser entendido como encontrar um modelo ajustado que relaciona uma resposta às entradas, tentando fazer a predição da resposta de forma mais acurada possível para futuros dados de entrada. No caso da classificação supervisionada de imagens de satélite, aqui apresentada, a ideia é associar classes do mundo real aos *pixels* e posteriormente, dado um *pixel* sem a associação, encontrar a classe à qual ele pertence.

2.3.1 Random Forest

Árvores de decisão são modelos utilizados quando as variáveis de entrada podem ser tratadas na forma de estruturas de seleção, as chamadas condicionais (se-então), até se chegar no resultado da predição. Essas estruturas de seleção são as regras que são utilizadas para se classificar um objeto. O *Random Forest* nada mais é do que uma forma mais sofisticada derivada das árvores de decisão.

No *Random Forest* diversas árvores são utilizadas para se construir um modelo preditivo mais robusto. Para isso é utilizada a técnica de *bootstrap*, que é uma ferramenta usada para quantificar a incerteza associada com um determinado método de aprendizagem de máquina (modelo). Ele pode ser usado em situações onde é complicado se calcular o desvio padrão de um conjunto de interesse.

Uma árvore de decisão pode ter alta variância, o que significa que se o conjunto de treinamento for dividido, consequentemente, cada modelo treinado com uma dessas divisões levará a resultados bem diversos. É neste ponto que o *bootstrap* entra em cena, para se reduzir a variância de um método de aprendizado de máquina (procedimento chamado de *bagging*). Para se obter o resultado desejado, um grupo de árvores é construído usando conjuntos de treinamento a partir da amostragem por *bootstrap*. Além disso, o *Random Forest* é capaz de descorrelacionar as árvores geradas. Mais detalhes dessas implementações podem ser encontradas na bibliografia (FRIEDMAN; HASTIE; TIBSHIRANI, 2019).

3. Roteiro prático

3.1 Preparação dos dados

A preparação dos dados para este laboratório tem similaridades com o de classificação não-supervisionada, porém com alguns passos mais sofisticados para se obter uma imagem mais completa para a classificação.

No laboratório anterior, áreas cobertas por nuvens foram tratadas juntamente com a superfície exposta, nesta prática deverão ser utilizadas imagens sem interferência de nuvens. Para isso, as nuvens devem ser removidas da imagem,

entretanto - teríamos nesse caso - o efeito de se ter vazios no meio de uma imagem. Uma forma de se tratar essa característica é capturar uma série de imagens da mesma região, excluir as nuvens de todas elas e ao fim consolidar todas essas imagens como apenas uma, a chamada técnica do mosaico. Essa técnica permite que *pixels* ausentes em uma imagem sejam substituídos pelos de outra que possui valores válidos na mesma posição. O GEE processa as imagens para compor o mosaico na ordem em que elas estão na coleção, ou seja, a prioridade do *pixel* é a da imagem que está mais acima na pilha de imagens da coleção.

3.1.1 Carregamento dos dados

A diferença dos dados utilizados neste laboratório em relação ao anterior, está no fato de que os dados utilizados são de uma coleção de imagens e não de apenas uma imagem. A coleção de imagens escolhida é a ***"Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-2A"***, que apresenta a reflectância de superfície (corrigida atmosfericamente) dos dados coletados pelo Sentinel-2 - ao importar a coleção de imagens, a nomeie como ***"colecaoSentinel2"***.

3.1.2 Parâmetros iniciais

A resolução das bandas do visível e do infravermelho próximo da coleção escolhida é de 10 metros e, por esse motivo, no *script* criado como base para este laboratório o valor foi atribuído em uma variável chamada ***"escala"*** para facilitar a codificação (Figura 2). Sendo assim, quando se deseja ajustar a escala para o processamento automaticamente - por exemplo para diminuir a resolução a fim de se trabalhar em uma área maior de forma computacionalmente eficiente - basta ajustar o valor da variável ***"escala"*** que o mesmo valerá para todo o código na sequência. Essa é uma sugestão para se manter um bom padrão de codificação, não um procedimento necessário. Além disso, pode-se definir uma centralização do mapa gerado em São Paulo, a partir da geometria gerada, para conveniência.

Figura 2 - Valor de escala atribuído à variável

```
// Centraliza o mapa em São Paulo.
Map.centerObject(retanguloSelecao, 12);
// Ajusta a escala (resolução) que será utilizada para o processamento.
var escala = 10;
```

Além do ajuste de escala é interessante definir as bandas que serão utilizadas na forma de uma lista de *strings*. Cada *string* representa o nome de uma determinada banda. Essa opção também é indicada para dar mais legibilidade ao código quando for necessário passar um grupo de bandas como parâmetro, como por exemplo no método ***"select()"*** de uma coleção ou imagem. Aqui, duas listas com nomes de bandas serão criadas, uma para as bandas para exclusão das nuvens das imagens e outra para a classificação da imagem resultante da composição das outras imagens sem as nuvens. Na lista ***"bandas"*** (Figura 3) são inseridas as bandas utilizadas na preparação da imagem para classificação, dentre elas as bandas que serão realmente utilizadas para a classificação (***"B2"***, ***"B3"***, ***"B4"*** e ***"B8"***), juntamente de índices de vegetação calculados a partir de algumas delas, e as bandas utilizadas para o processo de exclusão de pixels cobertos por nuvens. Essas são as bandas ***"MSK_CLDPRB"*** e ***"SCL"***. A banda ***"MSK_CLDPRB"*** é gerada por um algoritmo apropriado e contém, em cada *pixel*, o valor da probabilidade dele estar coberto por nuvens. Cabe salientar que a resolução dessa banda é de 20 metros. Já a banda ***"SCL"*** é fruto de um processo de classificação pelo algoritmo de classificação de cena (*Scene Classification*)[1]. Um dos passos da classificação de cena é a detecção de *pixels* cobertos por nuvens e sombras de nuvens. Essas bandas são próprias dos produtos Sentinel-2, para dados coletados por outros satélites existem bandas com funções similares. Uma lista completa das bandas que serão utilizadas para classificação pode ser vista na Figura 4, agora com a banda ***"NDVI"*** que deve ser calculada no item 3.1.7 deste roteiro, como visto no laboratório de classificação não-supervisionada. Para o trabalho prático, além do *NDVI*, devem ser calculados os índices *NDBI - Normalized Difference Built-Up Index* - (OLIVEIRA; CANDEIAS; TAVARES, 2019) e *EVI - Enhanced Vegetation Index* - (GINCIENE; BITENCOURT, 2011) para serem agregados ao procedimento da classificação.

[1] https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm

Figura 3 - Bandas para preparação dos dados

```
8  var bandas = ee.List(['B2', 'B3', 'B4', 'B8', 'MSK_CLDPRB', 'SCL']);
```

```
// Bandas que serão capturadas inicialmente para fazer a composição e calcular o
NDVI.
var bandas = ee.List(['B2', 'B3', 'B4', 'B8', 'MSK_CLDPRB', 'SCL']);
```

Figura 4 - Bandas para classificação

```
10  var bandasClassificacao = ee.List(['B2', 'B3', 'B4', 'B8', 'NDVI']);
```

```
// Bandas que serão utilizadas para as classificações.
var bandasClassificacao = ee.List(['B2', 'B3', 'B4', 'B8', 'NDVI']);
```

3.1.3 Seleção de parte de uma coleção de imagens

A seleção da coleção de imagens a ser utilizada (parte da coleção ***"Sentinel-2 MSI: MultiSpectral Instrument, Level-2A"***) segue um padrão parecido ao visto nos laboratórios onde ocorria a seleção de apenas uma imagem, naqueles laboratórios era utilizado o método ***"first()"*** para se obter a primeira imagem da coleção resultante. Aqui o método ***"first()"*** é removido e o objeto retornado é uma coleção com todas as imagens que obedecem os filtros aplicados. Na Figura 5 é apresentado o código para a seleção que deve ser adaptado para a região de trabalho do estudante. A dica aqui é trabalhar com o intervalo de datas e valor de cobertura por nuvens até se obter uma coleção de tamanho suficiente para a composição da imagem de forma apropriada, processo que será visto mais adiante. Os métodos apresentados na Figura 5 já devem ser familiares ao estudante e não serão aqui discutidos.

Figura 5 - Seleção das imagens a serem trabalhadas

```
14  var colecaoSentinel2Selecao = colecaoSentinel2.filterBounds(saoPaulo)
15                                .filterMetadata('CLOUDY_PIXEL_PERCENTAGE', 'less_than', 50)
16                                .filterDate('2019-01-01', '2020-01-01')
17                                .select(bandas);
```

```
// Seleciona apenas as bandas desejadas das imagens da coleção, que cubram o
```

```
ponto saoPaulo,
// sejam capturadas no ano de 2019 e que tenham cobertura por nuvens menor do
que 50%.
var colecaoSentinel2Selecao = colecaoSentinel2.filterBounds(saoPaulo)
                                    .filterDate('2019-01-01', '2020-01-01')
                                    .filterMetadata('CLOUDY_PIXEL_PERCENTAGE',
'less_than', 50)
                                    .select(bandas);
```

3.1.4 Recorte da área de interesse

Como no trabalho a ideia é utilizar uma coleção de imagens, o método ***"clip()"*** não pode ser aplicado diretamente nelas. Para efetuar o procedimento uma função deve ser criada e a partir de um *map* na coleção cada imagem é recortada. Na Figura 6 é apresentada a função que efetua tal procedimento. Essa função retorna a uma nova imagem onde ***"clip()"*** é efetuado na região parametrizada. No caso do trabalho prático, a região deve ser obtida a partir de um *shapefile* que será carregado como um *asset* no GEE. Esse procedimento será apresentado em um tutorial específico e aqui fica como exemplo a mesma região utilizada nos laboratórios didáticos.

Figura 6 - Função que recebe imagem e faz o recorte na geometria dada em *clip()*

```
20 var clipColecao = function (imagem) {
21   return(imagem.clip(retanguloSelecao));
22 };
```

```
// Função para recortar parte da imagem na região desejada.
var clipColecao = function (imagem) {
  return(imagem.clip(retanguloSelecao));
};
```

Para executar a função para toda a coleção, basta utilizar o ***"map()"*** como apresentado na Figura 7. O resultado será o mesmo grupo de imagens, mas somente com os dados da região selecionada.

Figura 7 - Map da função para selecionar apenas uma região da imagem

```
25  colecaoSentinel2Selecao = colecaoSentinel2Selecao.map(clipColecao);
```

```
// Faz o clip da região desejada em toda coleção.
colecaoSentinel2Selecao = colecaoSentinel2Selecao.map(clipColecao);
```

3.1.5 Remoção de áreas cobertas por nuvens

No GEE o método ***"updateMask()"*** é utilizado para se aplicar uma máscara em uma imagem, esse método pode ser aplicado seguidamente, ou seja, a partir do resultado da aplicação de uma máscara, outra pode ser aplicada. Na Figura 8 é apresentada uma função para o procedimento de remoção de nuvens na coleção com reflectância de superfície gerada a partir do Sentinel-2. Aqui, é importante entender os parâmetros passados para as chamadas de ***"updateMask()"***, lembrando que devem ser uma imagem contendo zeros e uns. O primeiro caso usa a banda com probabilidade de cobertura por nuvens, fazendo uma comparação com o método ***"lt()"*** (menor do que). Essa comparação retorna 1 para todo valor menor do que 0,5 na banda ***"MSK_CLDPRB"*** e 0 caso contrário, ou seja a máscara invalida qualquer pixel com probabilidade de cobertura por nuvens maior do que 0,5% (o intervalo dessa banda é entre 0 e 100). Na sequência, as máscaras são criadas a partir da banda ***"SCL"***, validando apenas locais que não são classificados como algum tipo de nuvem ou sombra de nuvem (***"neq()"*** executa a comparação diferente, ou não-igual). Os valores das classes em ***"SCL"*** podem ser vistos nas propriedades da coleção. Essas operações podem ser editadas conforme o desejado para se obter a melhor composição posteriormente.

Figura 8 - Função para remoção de regiões que possivelmente possuem nuvens

```
28  var removeNuvensColecao = function (imagem) {
29    var imagemSemNuvens = imagem.updateMask(imagem.select('MSK_CLDPRB').lt(0.5))
30                                .updateMask(imagem.select('SCL').neq(3))
31                                .updateMask(imagem.select('SCL').neq(7))
32                                .updateMask(imagem.select('SCL').neq(8))
33                                .updateMask(imagem.select('SCL').neq(9))
34                                .updateMask(imagem.select('SCL').neq(10));
35    return imagemSemNuvens;
36  };
```

```
// Faz a remoção de regiões que possivelmente possuem nuvens.
var removeNuvensColecao = function (imagem){
  var imagemSemNuvens = imagem.updateMask(imagem.select('MSK_CLDPRB').lt(0.5))
                              .updateMask(imagem.select('SCL').neq(3))
                              .updateMask(imagem.select('SCL').neq(7))
                              .updateMask(imagem.select('SCL').neq(8))
                              .updateMask(imagem.select('SCL').neq(9))
                              .updateMask(imagem.select('SCL').neq(10));
  return imagemSemNuvens;
};
```

Em seguida, basta executar o *map* da função criada a partir da coleção de imagens como na Figura 9.

Figura 9 - Executa a função de remoção de nuvens na coleção

```
var colecaoSentinel2SelecaoSemNuvens = colecaoSentinel2Selecao.map(removeNuvensColecao);
```

```
// Executa a função de remoção das nuvens na coleção.
var colecaoSentinel2SelecaoSemNuvens =
colecaoSentinel2Selecao.map(removeNuvensColecao);
```

3.1.6 Remoção de valores inválidos

Após a remoção das nuvens e suas sombras, para se dar mais robustez ao conjunto de dados, podem ser eliminados valores discrepantes. No caso, como a coleção de reflectância de superfície do Sentinel-2 apresenta os valores de reflectância entre zero e 10 mil, basta criar uma máscara tornando pixels onde em alguma das bandas com reflectância estão com valores maiores do que 10 mil inválidos. Na Figura 10 é vista uma função para aplicar as máscaras seguidamente para as bandas. As máscaras são geradas a partir do comparador ***"lt()"*** (menor do que) aplicado ao valor 10 mil.

Figura 10 - Função para remoção de valores inválidos

```
var removeValoresInvalidos = function (imagem) {
  var imagemSemValoresInvalidos = imagem.updateMask(imagem.select('B2').lt(10000))
                                        .updateMask(imagem.select('B3').lt(10000))
                                        .updateMask(imagem.select('B4').lt(10000))
                                        .updateMask(imagem.select('B8').lt(10000));
  return imagemSemValoresInvalidos;
};
```

```
// Função para remoção de valores inválidos
var removeValoresInvalidos = function (imagem){
  var imagemSemValoresInvalidos =
imagem.updateMask(imagem.select('B2').lt(10000))

.updateMask(imagem.select('B3').lt(10000))

.updateMask(imagem.select('B4').lt(10000))

.updateMask(imagem.select('B8').lt(10000));
  return imagemSemValoresInvalidos;
};
```

Em seguida, basta executar o ***"map()"*** da função criada a partir da coleção de imagens como na Figura 11.

Figura 11 - Executa a função de remoção de valores inválidos na coleção

```
var colecaoSentinel2SelecaoSemNuvens = colecaoSentinel2SelecaoSemNuvens.map(removeValoresInvalidos);
```

```
// Executa a função de remoção dos valores inválidos na coleção.
var colecaoSentinel2SelecaoSemNuvens =
colecaoSentinel2SelecaoSemNuvens.map(removeValoresInvalidos);
```

3.1.7 Composição e Mosaico

Conforme o código apresentado na Figura 12, para se criar um mosaico a partir de uma coleção de imagens, basta invocar o método ***"mosaic()"*** da coleção instanciada. Já para a criação de uma composição, um método do tipo *reducer* que faz um cálculo *pixel* a *pixel* deve ser chamado a partir do objeto que contém a coleção instanciada. Nesse caso, os *pixels* "transparentes" são ignorados durante os cálculos. No exemplo da Figura 13, a média de cada *pixel* é calculada com o método ***"mean()"***. Os cálculos são feitos de forma separada para cada banda das imagens da coleção, resultando assim em uma imagem com as mesmas bandas da coleção, onde cada *pixel* de cada banda é preenchido por um valor calculado a partir da banda de mesmo nome na coleção. Outros valores podem ser calculados, como o

máximo ou mínimo, ficando a critério do estudante testar e escolher o que achar mais interessante, seja o simples mosaico ou a composição a partir de um cálculo.

Figura 12 - Criação de mosaico a partir de coleção

```
55  var imagem = colecaoSentinel2SelecaoSemNuvens.mosaic();
```

```
// Criação de mosaico a partir da coleção.
var imagem = colecaoSentinel2SelecaoSemNuvens.mosaic();
```

Figura 13 - Criação de composição a partir de coleção

```
57  var imagem = colecaoSentinel2SelecaoSemNuvens.mean();
```

```
// Criação de composição a partir da coleção.
var imagem = colecaoSentinel2SelecaoSemNuvens.mean();
```

3.1.7 Ajuste nos valores da reflectância e índices de vegetação

Para ficar com valor de reflectância entre 0 e 1, basta fazer a multiplicação da imagem pelo valor 0,0001 - Figura 14. Após esse ajuste, conforme visto no laboratório prático de classificação não-supervisionada, deve ser calculado o NDVI. Mas note que agora a banda correspondente ao infravermelho próximo (NIR) é a ***'B8'***. Para o trabalho prático, como dito anteriormente, é necessário fazer o cálculo de mais dois índices, o NDBI e o EVI.

Figura 14 - Ajuste nos valores das imagens

```
60  imagem = imagem.multiply(0.0001);
```

```
// Ajuste nos valores de reflectância das imagens.
imagem = imagem.multiply(0.0001);
```

3.1.8 Seleção de bandas para a classificação

Para finalizar a preparação dos dados para a classificação, deve ser feita a seleção apenas das bandas que servem para o procedimento - Figura 15. Conforme sugerido na seção 3.1.1, os nomes das bandas podem estar em uma lista. Quando for utilizar o *NDBI* e o *EVI*, os nomes dessas bandas devem ser adicionados à lista.

Figura 15 - Seleção das bandas para a classificação

```
80  imagem = imagem.select(bandasClassificacao);
```

```
// Descarta as bandas usadas para se eliminar efeitos de nuvens.
imagem = imagem.select(bandasClassificacao);
```

3.2 Classificação supervisionada no GEE

Para se efetuar a classificação supervisionada no GEE, regiões de onde devem ser coletadas as amostras podem ser definidas no mapa do GEE. Depois, os valores dos *pixels* são extraídos dessas regiões para servir de amostra para os procedimentos de classificação, validação e teste. Após o treinamento, a classificação é feita com o classificador treinado (objeto do tipo ***"ee.Classifier"***) a partir da imagem.

3.2.1 Preparação das amostras

Através da interface do mapa do GEE, devem ser criadas *Feature Collections* contendo polígonos que cobrem regiões de onde as amostras serão retiradas. Uma *Feature Collection* deve ser criada para cada classe desejada - como vegetação, por exemplo. Cada *Feature Collection* deve ter um atributo numérico inteiro identificando a sua classe.

3.2.2 Delimitação das áreas das amostras

Para esse exercício, repetiremos o processo a seguir cinco vezes, coletando cinco tipos de regiões que servirão como amostras para classificação. Os 5 tipos utilizados serão Vegetação Arbórea (nomear a *Feature Collection* ***"amostraVegetacaoAlta"*** *- sem as aspas),* Vegetação rasteira *(**"amostraVegetacaoRasteira"**),* Solo Exposto *(**"amostraSoloExposto"**),* Construções *(**"amostraConstrucoes"**),* locais com água *(**"amostraAgua"**)*. Você pode utilizar o mapa de satélite, que pode ser ativado do lado direito do mapa, para auxiliar na identificação das áreas.

Para criar uma *Feature Collection* com as regiões de onde serão coletadas amostras para uma determinada classe, primeiramente o mouse deve ser movimentado sobre o painel onde está escrito ***"Geometry Imports"*** - Figura 16. Uma lista será aberta (caso já tenha camadas com geometrias, elas são exibidas nesta lista) e nela pode ser visto, na parte inferior, um *link* com a identificação ***"+ new layer"*** - Figura 17. Basta clicar neste *link* para criar uma nova camada com geometrias. Ao clicar no *link* ***"+ new layer"***, uma nova camada é adicionada na lista de camadas de geometrias com um nome gerado automaticamente - Figura 18.

Figura 16 - Painel onde são adicionadas as camadas das Features Collections criadas

Figura 17 - Link para se criar um novo layer

Figura 18 - Nova camada de geometria criada

Para se alterar as configurações da nova camada criada, basta clicar no ícone em forma de engrenagem na linha da camada - Figura 18. Ao fazer isso, uma nova janela será aberta (Figura 19) e é possível renomear o nome para um mais adequado (seguindo o mesmo padrão de nomes de variáveis, como ***"amostraVegetacao"*** para áreas com vegetação). Na mesma janela, deve ser alterado o tipo do objeto para importação na lista abaixo de ***"Import as"*** para ***"FeatureCollection"*** (a ideia é colocar diversas *Features* para cada classe, todas

em uma única *Feature Collection*). Também deve ser adicionada uma propriedade para identificar a classe. Isso é feito com um clique sobre o botão ***"+ Property"*** e em seguida na nova propriedade adicionada na lista de propriedades (***"Properties"***) introduzindo-se um nome em ***"Property"*** (seguindo o padrão de nomes de variáveis) e um número em ***"Value"*** para identificar a classe representada na *Feature Collection*. O nome da propriedade deve ser o mesmo para todas as *Features Collections* (no exemplo apresentado sempre será ***"classe"***), já o número atribuído deve ser único para cada classe, pois é esse valor que o GEE usa para diferenciar as classes (pode ser usado o padrão começando em 0 e seguindo para 1, 2 e assim por diante). Finalmente, a cor pode ser configurada do lado direito. É interessante aproveitar e copiar o código hexadecimal da cor (sem o carácter de cerquilha) juntamente com a classe à qual pertence para uso posterior. Para salvar as alterações, basta clicar no botão ***"OK"***. A *Feature Collection* na lista deve estar atualizada, como na Figura 20.

Figura 19 - Configurações da camada

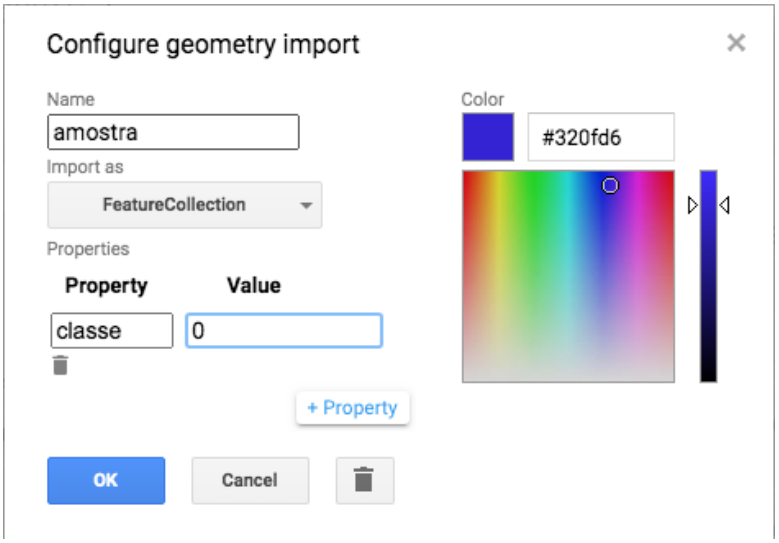
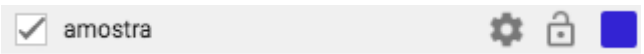


Figura 20 - Camada com as configurações alteradas na lista



Para se iniciar o desenho das regiões de amostra no mapa, primeiro é preciso clicar sobre a *Feature Collection* que se deseja alterar na lista de ***"Geometry Imports"***. Ao fazer isso, um novo painel será aberto ao lado do das ***"Geometry Imports"*** como o da Figura 21. Caso ele não apresente o título ***"Polygon drawing"***,

não estará apto a desenhar polígonos (o tipo padrão de geometria é de pontos). Para alterar o tipo de geometria, basta clicar no ícone para se desenhar polígonos no painel do lado esquerdo do ***"Geometry Imports"***, que tem o desenho da Figura 22. Ao selecionar esse tipo de geometria, o painel do lado direito deve ficar como o da Figura 21, apresentando o título ***"Polygon drawing"***.

Figura 21 - Edição da camada ativada

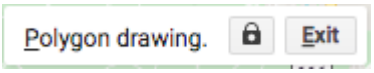
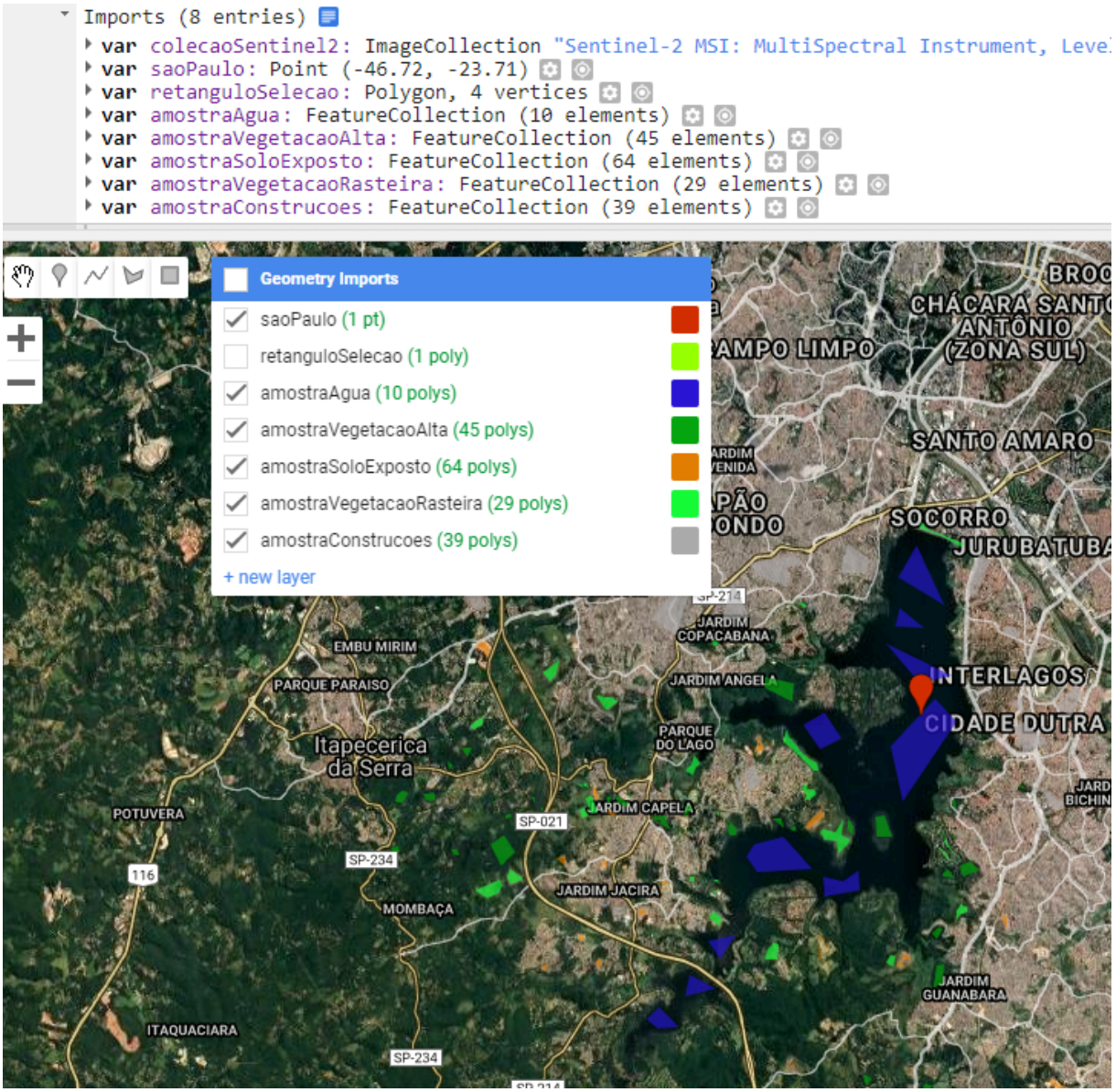


Figura 22 - Ícone para ativação do desenho de polígonos

Com o modo de desenho ativado apropriadamente, basta procurar regiões da classe que está sendo colhida e desenhar livremente diversos polígonos, desde que dentro da região de estudos delimitada - Figura 22. Observe que a composição (ou mosaico), feita a partir das imagens sem nuvens deve ser adicionada para se encontrar as regiões com as classes desejadas. É um trabalho manual e fica a critério do estudante a quantidade de polígonos a serem desenhados. Idealmente, para cada classe, devem estar bem espalhados pela região toda que será classificada. Para finalizar a classe atual, basta clicar em ***"Exit"*** no painel ao lado direito do painel ***"Geometry Imports"*** - Figura 21.

Ao final de todo o processo de criação das amostras de treinamento, deveremos ter algo como na Figuras 23.

Figura 23 - Exemplo de polígono desenhado no mapa



3.2.3 Gerando as amostras automaticamente no GEE

O procedimento feito na seção anterior apenas delimitou as regiões de onde se deseja coletar amostra, associando-as com suas respectivas classes, e não colheu nenhuma amostra propriamente. Depois da delimitação das regiões das amostras para cada classe, é feita a coleta dos valores de *pixels* nas regiões, devendo os mesmos carregarem a propriedade que identifica suas respectivas classes. Ao fim do processo, os pixels amostrados de todas as *Feature Collections* das regiões definidas como áreas de amostras são consolidadas em uma única *Feature Collection* que é o conjunto das amostras. Esse conjunto é formado por *Features* em que as propriedades são os valores numéricos do *pixel* de onde foi feita

a coleta da amostra mais um representando a classe do *pixel*. Nas próximas seções serão apresentadas duas formas de se obter amostras com os valores dos *pixels* nas regiões desenhadas no mapa. Para o procedimento de classificação deve se optar por apenas uma das seleções de amostras.

3.2.4 Amostragem com todos os *pixels* nas áreas

A primeira opção aqui apresentada busca coletar todos os *pixels* que estão nas regiões desenhadas pelo usuário. Para esse procedimento, primeiro todas as *Feature Collections* com as regiões das classes devem ser juntadas em uma só com código similar ao da Figura 24. O método ***"merge()"*** funde duas coleções em uma, a que é usada para chamar o método (lado esquerdo) e a que é passada como parâmetro para o método (entre parênteses). O resultado é uma *Feature Collection* com todos os elementos das duas coleções. Isso é feito progressivamente, se encadeando o resultado com um novo ***"merge()"*** para todas as *Feature Collections* das regiões de amostra. Ao final, todas as *Features* dentro das *Features Collections*, desenhadas pelo usuário, estarão em apenas uma *Feature Collection* com suas respectivas geometrias e propriedade identificando a classe.

Figura 24 - Juntando diversas Features Collection em uma só

```
83  var amostraCompletaGeometrias = amostraAgua.merge(amostraVegetacaoAlta)
84                                               .merge(amostraSoloExposto)
85                                               .merge(amostraVegetacaoRasteira)
86                                               .merge(amostraConstrucoes);
```

```
// Agrega todas as amostras coletadas pelo usuário em uma FeatureCollection.
var amostraCompletaGeometrias = amostraAgua.merge(amostraVegetacaoAlta)
                                           .merge(amostraSoloExposto)
                                           .merge(amostraVegetacaoRasteira)
                                           .merge(amostraConstrucoes);
```

Com as regiões de amostras fundidas em apenas uma *Feature Collection*, o processo para se obter todos os *pixels* dentro das regiões é feito com o método ***"sampleRegions()"*** invocado a partir da imagem que será classificada e com a *Feature Collection* obtida como parâmetro para as regiões de buscas dos *pixels*. A

Figura 25 apresenta como deve ser feita essa chamada do método ***"sampleRegions()"***. Além da chave ***"collection"*** no dicionário de dados passado como parâmetro para o método, a escala é definida em ***"scale"*** (caso queira trabalhar com uma região muito grande, a escala pode ser alterada para se utilizar *pixels* de maior resolução e o GEE não ter problemas na computação do algoritmo). A chave *properties* é muito importante, pois é uma lista que diz quais propriedades devem ser copiadas junto dos valores numéricos das bandas de cada *pixel*. Aqui o interesse é sobre a identificação da classe, para que o algoritmo possa saber a que classe aquele *pixel* pertence. O resultado de ***"sampleRegions()"*** é uma nova *Feature Collection* que contém uma propriedade para cada banda da imagem de entrada, que contém o valor numérico do *pixel* naquela banda, acrescido das propriedades que devem ser copiadas.

Figura 25 - Gera a amostra

```
89  // Extrai o valor de todos os pixels nas regiões amostradas.
90  var amostraTreinamento = imagem.sampleRegions({
91    collection: amostraCompletaGeometrias,
92    properties: ['classe'],
93    scale: escala
94  });
```


```
// Extrai o valor de todos os pixels nas regiões amostradas.
var amostraTreinamento = imagem.sampleRegions({
  collection: amostraCompletaGeometrias,
  properties: ['classe'],
  scale: escala
});
```

3.2.5 Treinamento e classificação

O passo inicial do processo é treinar o modelo desejado e em seguida classificar a imagem.

3.2.5.1 Treinamento e matriz de confusão

Para se treinar o modelo, basta instanciar um classificador escolhido com os parâmetros desejados e utilizar o método ***"train()"***. Na Figura 26 é apresentado o código para o algoritmo *Random Forest*. O primeiro parâmetro de ***"train()"*** é a amostra a ser utilizada para treinar o modelo, o segundo é a propriedade que contém o número que identifica a classe e o último, a lista com as bandas que serão utilizadas para classificação. Na instanciação do classificador, ***"smileRandomForest()"***, o parâmetro é a quantidade de árvores de decisão que devem ser criadas. Esse valor pode, e deve, ser ajustado de acordo com os resultados no conjunto de validação. Outros parâmetros podem ser encontrados na documentação do GEE e a explicação da utilidade dos mesmos em literatura apropriada.

Figura 26 - Execução do Random Forest no GEE

```
161 var classificadorTreinado = ee.Classifier.smileRandomForest(50)
162                                          .train(amostraTreinamento, 'classe', bandasClassificacao);
```

```
// Instancia um classificador na memória com os parâmetros dados e treinando no
conjunto de treinamento.
var classificadorTreinado = ee.Classifier.smileRandomForest(50)
                                         .train(amostraTreinamento, 'classe',
bandasClassificacao);
```

Um primeiro teste pode ser feito com a própria amostra de treinamento, com a criação da matriz de confusão a partir do método ***"confusionMatrix()"*** - Figura 27. A acurácia provavelmente será altíssima, visto que foi o próprio conjunto utilizado no treinamento que foi avaliado. Caso o desempenho tenha sido ruim sobre o conjunto de treinamento, é um provável caso de *underfitting*. O que significa que o treinamento não foi suficiente para se identificar as classes. Isso pode ocorrer devido

aos parâmetros do modelo, insuficiência ou má distribuição das amostras. Deve ser ajustado antes de se partir para uma validação.

Figura 27 - Matriz de confusão

```
165  var matrizConfusaoAmostraTreinamento = classificadorTreinado.confusionMatrix();
166  print('Matriz de confusão: ', matrizConfusaoAmostraTreinamento);
167  print('Acurácia no conjunto de treinamento: ', matrizConfusaoAmostraTreinamento.accuracy());
```

```
// Matriz de confusão do conjunto de treinamento e acurácia.
var matrizConfusaoAmostraTreinamento = classificadorTreinado.confusionMatrix();
print('Matriz de confusão: ', matrizConfusaoAmostraTreinamento);
print('Acurácia no conjunto de treinamento: ',
matrizConfusaoAmostraTreinamento.accuracy());
```

3.2.5.2 Classificação da imagem

Para fazer a classificação da imagem, a partir do treinamento, invoca-se o método ***"classify()"*** a partir da imagem e com o classificador treinado escolhido como parâmetro - Figura 28. O resultado é um objeto do tipo imagem com uma única banda em que os *pixels* armazenam o valor relativo às classes que foram atribuídas no momento do desenho das regiões das amostras em tela. As bandas são comparadas pelo nome com o classificador criado.

Figura 28 - Classificação da imagem

```
186  var imagemClassificada = imagem.classify(classificadorTreinado);
```

```
// Classifica a imagem com o classificador treinado e com os parâmetros
definidos.
var imagemClassificada = imagem.classify(classificadorTreinado);
```

3.3 Visualização da classificação

Para visualizar a imagem, primeiro uma paleta de cores deve ser definida. Essa paleta deve ser feita como uma string em que cada cor é uma sequência de seis dígitos em hexadecimal separados por vírgula - Figura 29. A ordem em que as cores estão na *string* representa o número inteiro associado com a classe a partir de 0. As cores podem ser escolhidas à gosto, mas pode ser interessante usar as cores

já definidas nas regiões amostradas na tela. Esse valor está nas propriedades da *Feature Collection*, como na Figura 19 mostrada anteriormente.

```
189 var paletaCoresClasses = "2915d6," + // Água.
190                          "08a910," + // Vegetação alta.
191                          "e17f04," + // Solo exposto.
192                          "14fb3b," + // Vegetação rasteira.
193                          "aaaaaa";   // Construção.
```

Figura 29 - Paleta de cores

```
// Paleta de cores para adicionar a imagem classificada ao mapa.
var paletaCoresClasses = "2915d6," + // Água.
                         "08a910," + // Vegetação alta.
                         "e17f04," + // Solo exposto.
                         "14fb3b," + // Vegetação rasteira.
                         "aaaaaa";   // Construção.
```

A criação de uma camada com a imagem é feita com a função ***"Map.addLayer()"*** - Figura 30. A diferença do uso dessa função nos laboratórios da disciplina, está na adição da paleta (***"pallete"***) e do uso de ***"min"*** e ***"max"*** para representar o valor mínimo e máximo do atributo que identifica as classes. O formato pode ser ajustado para png via chave ***"format"***.

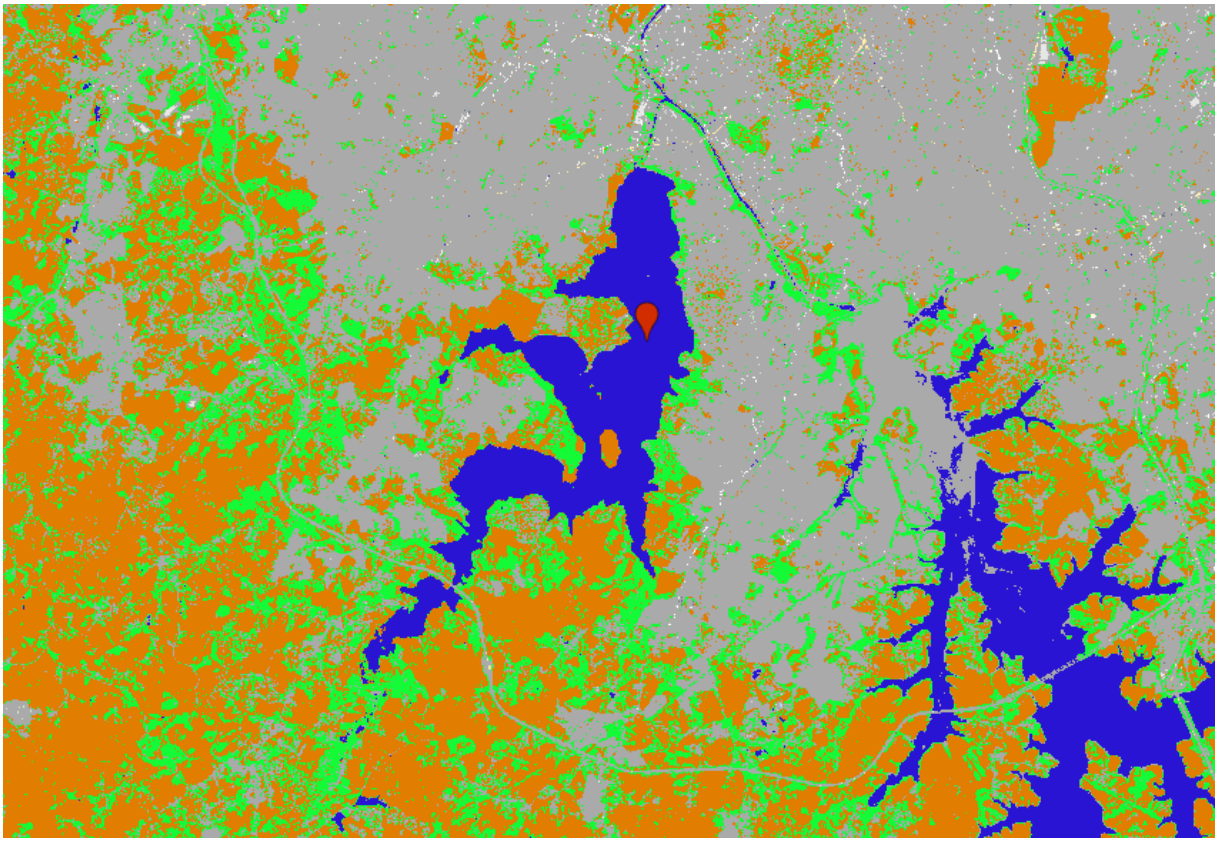
Figura 30 - Criação de camada com a imagem classificada

```
196 Map.addLayer(imagemClassificada.clip(retanguloSelecao), {
197       "min": 0,
198       "max": 4,
199       "palette": paletaCoresClasses,
200     "format": "png"
201 },  'Imagem Classificada',true);
```

```
// Adiciona o layer com o resultado da classificação.
Map.addLayer(imagemClassificada.clip(retanguloSelecao), {
      "min": 0,
      "max": 4,
      "palette": paletaCoresClasses,
    "format": "png"
},  'Imagem Classificada',true);
```

Ao final de todo o código e depois de rodar o classificador o mapa final será algo como o da Figura 31.

Figura 31 - Mapa Final



4. Referências bibliográficas

BOEHMKE, B.; GREENWELL, B. M. Hands-on machine learning with R. CRC Press, 2019. Disponível em: https://bradleyboehmke.github.io/HOML/. Acesso em: 24 set. 2020.

CRÓSTA, A. P. Processamento Digital de Imagens de Sensoriamento Remoto. Campinas: Ed. Rev. Campinas, UNICAMP. 1992.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. The elements of statistical learning. New York: Springer series in statistics, 2001.

GORELICK, N. et al. Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment, [s. l.], v. 202, p. 18-27. 2017.

GARETH, J. et al. An introduction to statistical learning: with applications in R. Spinger, 2013.

GINCIENE, B. R.; BITENCOURT, M. D. Utilização do EVI (Enhanced Vegetation Index) para maior sensibilidade na detecção de mudanças temporais em fragmentos

de floresta estacional semidecidual. Simpósio Brasileiro de Sensoriamento Remoto, Curitiba, PR, 2011.

LILLESAND, T.; KIEFER, R. W.; CHIPMAN, J. Remote sensing and image interpretation. John Wiley & Sons, 2015.

OLIVEIRA, C. B. de S.; CANDEIAS, A. L. B.; TAVARES, J. R.. . Revista Brasileira de Geografia Física, v. 12, n. 03, p. 1039-1053, 2019.