



Sensoriamento Remoto I

Código GA111

Engenharia Cartográfica

Universidade Federal do Paraná

Departamento de Geomática

**Prática de Docência: Andressa P. Cavassim
2024**



Random Forest (*Floresta aleatória*)

É um algoritmo de aprendizado em máquina e tem como finalidade criar uma floresta de árvores de decisão (*Decision trees*) independentes para ao final combinar suas previsões e criar uma solução otimizada para problemas de regressão e CLASSIFICAÇÃO.

Mercado financeiro

Áreas hospitalares

Sensoriamento remoto

Setor bancário

Comércio eletrônico



Regressão

- Neste caso, o objetivo é calcular uma variável contínua e não uma classe.
- Para selecionar as variáveis mais relevantes em cada nó são usadas medidas estatísticas, como a média e o desvio padrão.
- A árvore de regressão utiliza métricas como o Erro Quadrático Médio para avaliar o quão bem as divisões estão prevendo os valores numéricos.
- O resultado final é um valor numérico.

Decision Tree

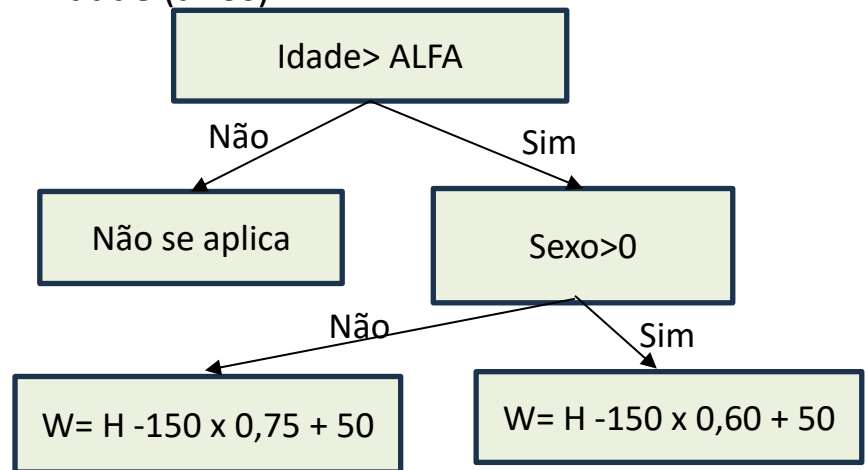
Uma árvore de decisão é uma **estrutura hierárquica** que **armazena uma sequência de decisões** para a solução de um determinado problema. Pode ser usada para regressão ou classificação.

Uma árvore de decisão para um problema de classificação.

Peso ideal:
fórmula de Wan der Vael.

Entradas:

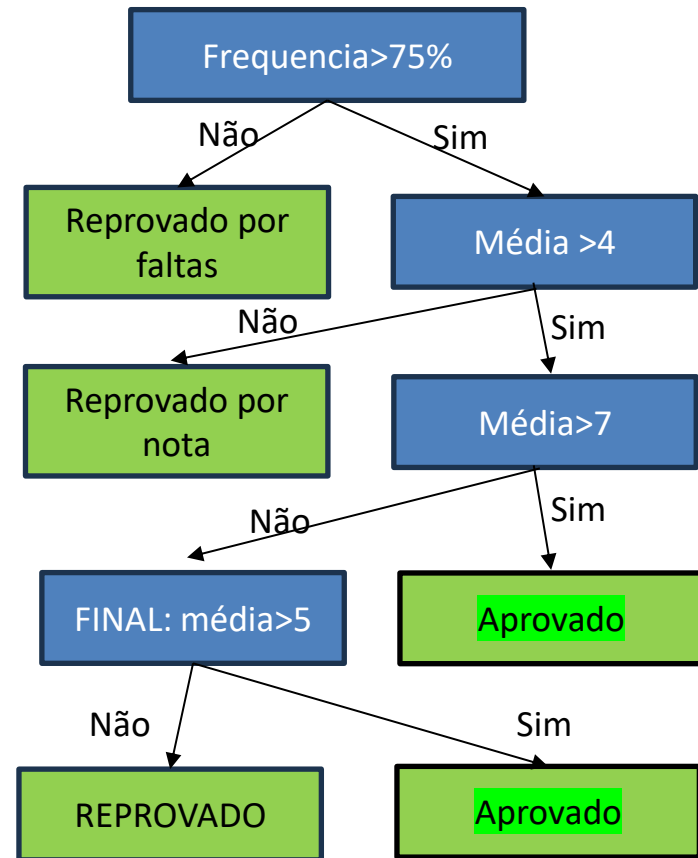
- Sexo (Masc, fem: 0,1),
- altura (H em cm)
- peso (W em Kg),
- idade (anos).



Regressão

Decision Tree

Uma árvore de decisão para um problema de classificação.

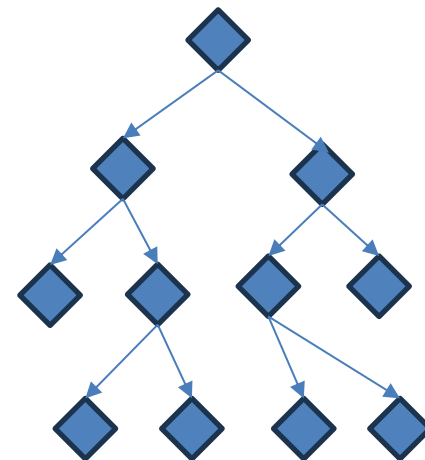


Classificação Hierárquica

Classificação Hierárquica

São métodos de classificação baseados em múltiplos níveis de decisão, que levam em conta a hierarquia entre as classes previamente definidas

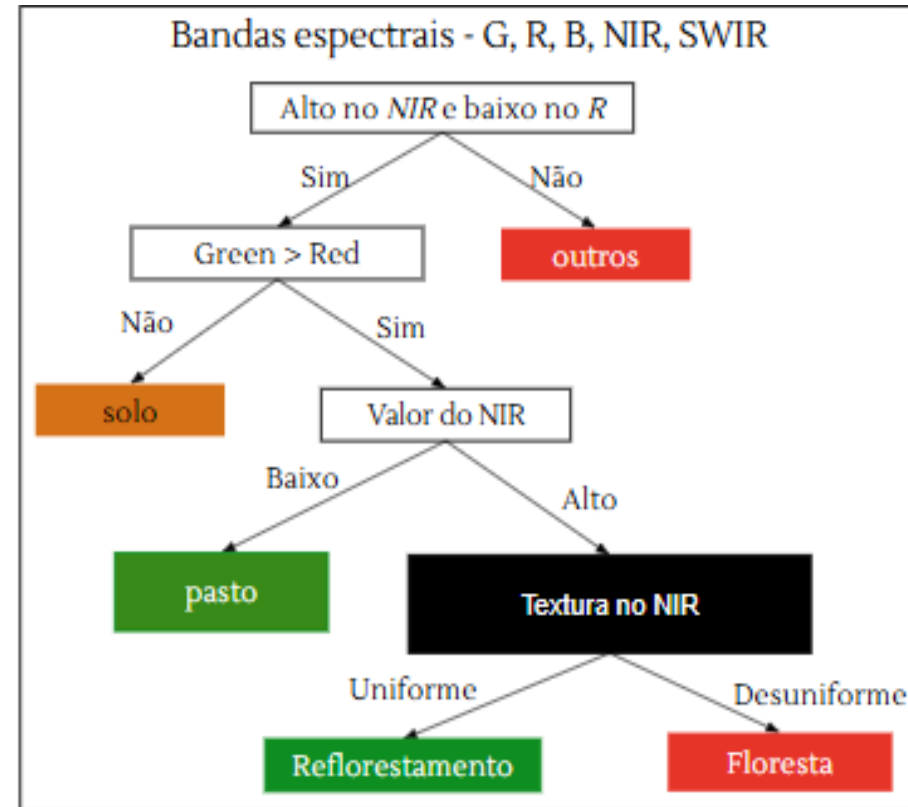
- A hierarquia entre níveis das classes serve para definir uma classe (classificação supervisionada). Ex: cidade - quadra – casa. TOP-DOWN.
- Também se pode definir classes maiores partir da similaridade entre classe (classificação não supervisionada). BOTTOM-UP.



Árvore de decisão

Uma árvore de decisão pode ser usada como:

1. Método de classificação, para a tomada de decisões progressivas em função das variáveis disponíveis.
2. Para detectar as variáveis mais significativas para a solução de um determinado problema (*feature selection*).

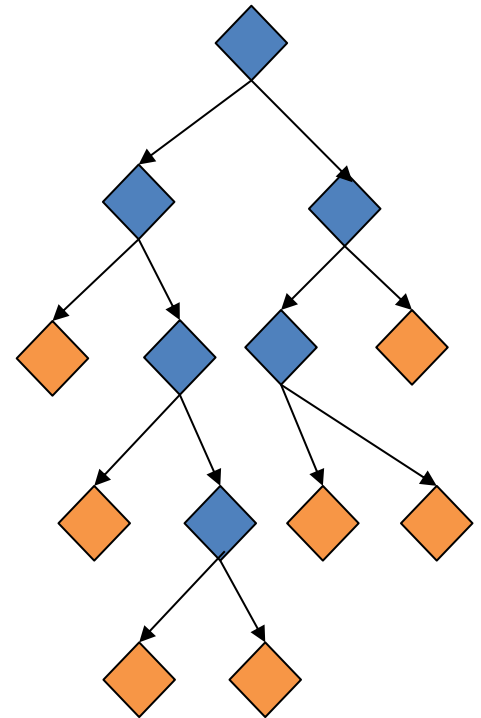


Que variáveis foram usadas para resolver o problema?

Neste algoritmo, várias decisões são tomadas

- Cada “nó” da árvore é um ponto de decisão;
- Após uma decisão, pode-se seguir diferentes caminhos, representados como ‘ramos’.
- Quando não existem mais “ramos”, o nó é considerado um ponto final, uma instância, “uma folha da árvore”.

Para essas tomadas de decisão existem alguns métodos. E seu funcionamento depende da seleção das variáveis a serem usadas em cada nó e o critério aplicado em cada nó, para deixar fluir a informação ao longo da árvore.



Seleção da(s) variável(is)



Na construção de uma árvore de decisão deve-se avaliar, a cada momento, a necessidade de se criar um nó e qual variável usar neste nó. Para isto, pode-se usar:

- **A Entropia**
- **O Índice Gini**



ENTROPIA

$$Entropia(S) = \sum x * \log(x)$$

Medida da variação das variáveis (medida da desordem de um sistema). A entropia varia entre 0 e 1.

- $E=0$ os dados estão mais organizados.
- $E=1$ existe maior desorganização dos dados, ou seja, onde existem mais dados disponíveis.

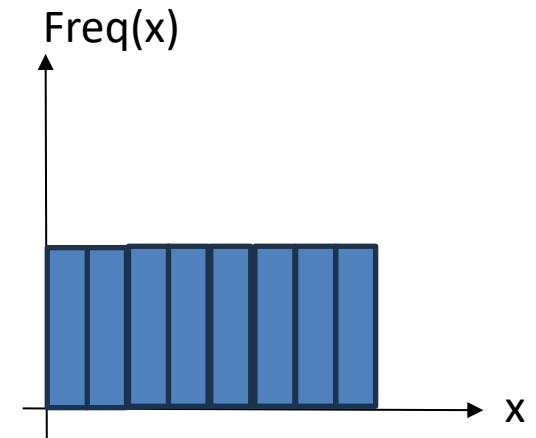
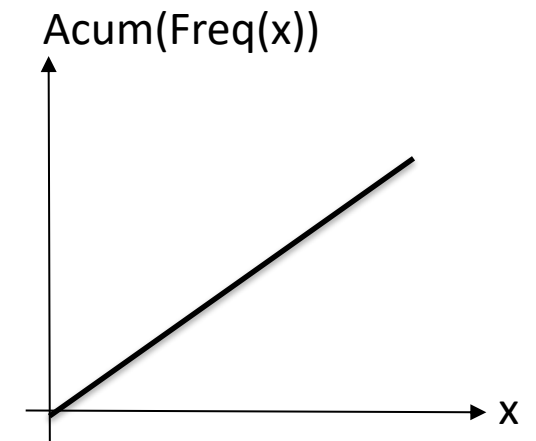
Calcula-se a Entropia de cada variável disponível e seleciona-se aquela associada à maior entropia (pois possibilita maior ganho de informação ao longo da árvore de decisão).

Índice Gini

Irá medir a uniformidade dos dados, onde para uma variável 'x', irá calcular a frequência ou o histograma. Para calcular o acúmulo do histograma:

$$Ac = \text{soma}(i=0...a) = \text{Freq.}(x)$$

Ordenando o acúmulo, é construída uma curva que mostra a uniformidade dos dados. E no caso de um conjunto uniforme é uma reta a 45 graus.



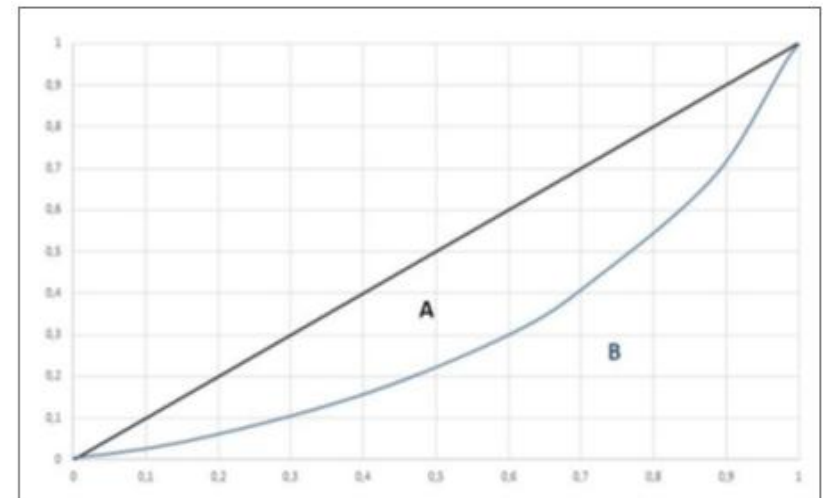
Índice Gini

Um conjunto uniforme produz uma reta. Quanto mais heterogêneo o conjunto, mais a curva acumulada se afasta desta reta.

Na figura:

- B: curva da distribuição da variável
- A+B = curva ideal
- A: diferença

$$Gini(V) = 1 - Area(V)$$



Uniforme: $Gini = 1 - 0,5 = 0,5$

Desuniforme: $Gini = 1 - B$, com $B < 0,5$, valores entre 0 e 0,5.



Overfitting

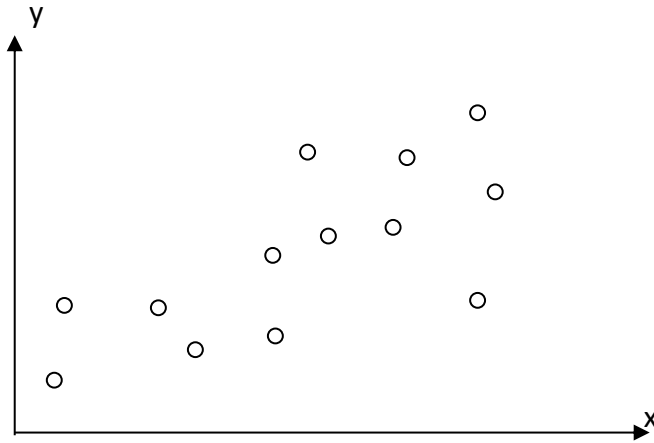
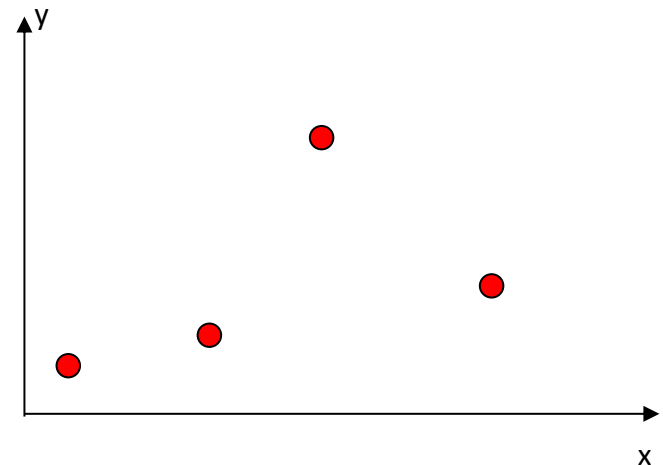
É o ajuste extremo com os dados de treinamento (sobreajuste).

- Neste caso, o modelo criado se adapta excessivamente ou até mesmo de forma precisa aos dados de treinamento (amostras), mas pode falhar para outras amostras.
- Isto conduz o modelo a falsas conclusões e baixa acurácia.

Overfitting

Qual modelo, calculado a partir dos pontos de treinamento (vermelho) é melhor para prever as amostras de treinamento?

E para outras amostras?



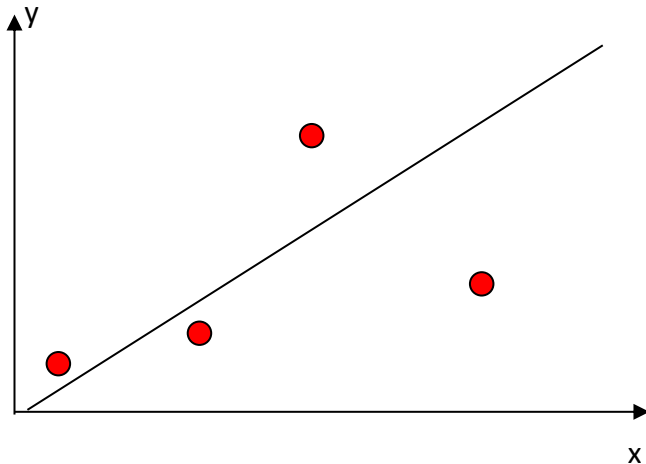
Pontos de treinamento

Overfitting

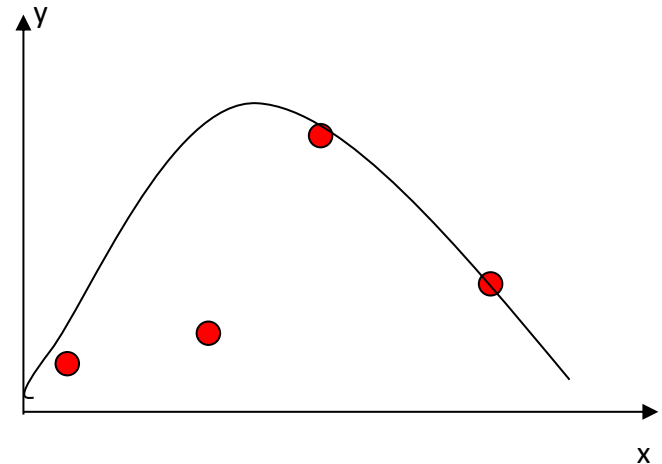
Qual modelo, calculado a partir dos pontos de treinamento (vermelho) é melhor para prever as amostras de treinamento?

E para outras amostras?

Modelo 1

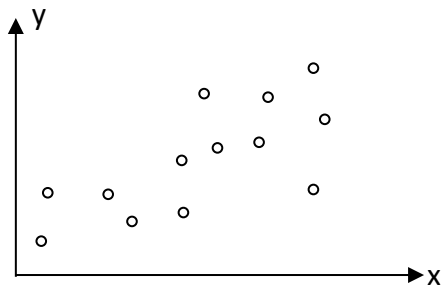
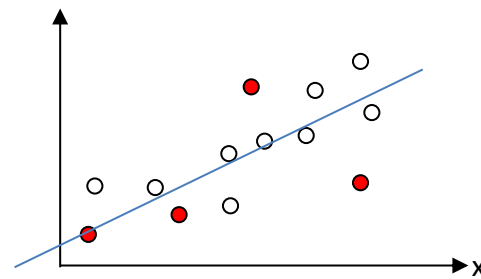


Modelo 2

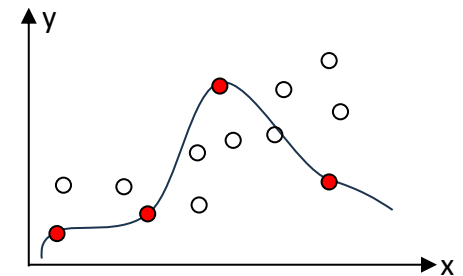


Overfitting

Dado os gráficos selecione e estime o melhor modelo:



Polinômio 1º grau



Polinômio alto grau

Random Forest

O conceito de Random forest (floresta aleatória) foi criado no início dos anos 2000 por Leo Breiman e Adele Cutler, ambos estatísticos com vasta experiência em métodos de análise de dados complexos.

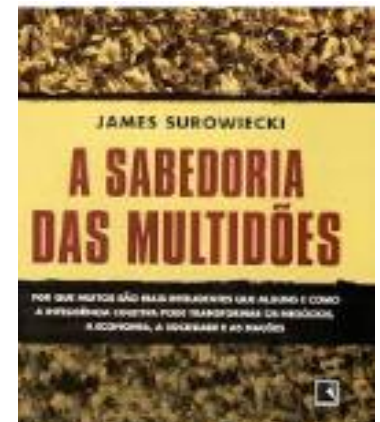
A ideia inicial surgiu porque as árvores de decisão, apesar de serem boas em classificações, têm limitações, como a tendência ao overfitting.



Leo Breiman e Adele Cutler

Random Forest

- Breiman e Cutler perceberam que, ao criar várias árvores de decisão independentes e combinar suas previsões, era possível obter um modelo mais robusto e preciso.
- Esse método é baseado no conceito "sabedoria das multidões" onde cada árvore contribui com uma opinião, e a média ou maioria das opiniões tende a acertar mais do que uma árvore isolada.





Random forest

No método Random Forest (floresta aleatória) são criadas muitas árvores de decisão. A característica destas árvores é que cada uma contém apenas um subconjunto das variáveis e amostras de treinamento.

Assim, cada árvore é otimizada, mas resulta em uma solução diferente, pois não se tem acesso a toda a informação de treinamento.

Após a definição das árvores, elas são combinadas para se calcular um resultado otimizado, em um processo similar a uma votação.

Para isto, é usado o método ensemble.



Ensemble

O método de "ensemble" (conjunto) funciona da seguinte maneira:

- Com o método de bagging são criadas condições para a formação de árvores diferentes;
- O mesmo algoritmo de formação de árvores é aplicado a cada conjunto, produzindo árvores diferentes;
- Com um método de agregação é produzida uma solução otimizada, combinando vários resultados.



Bagging

O Bagging faz uma escolha aleatória de um número menor de amostras para construir a árvore de decisão.

Ele escolhe um conjunto limitado de amostras, sorteando algumas amostras do conjunto total de dados e escolhe uma e, depois, faz outro sorteio e escolhe a segunda; e, dessa forma, ele segue sorteando até atingir a percentagem estipulada de amostras.

Com isso, cada árvore será construída com uma quantidade limitada de amostras de treinamento, selecionadas aleatoriamente.

Obs: algumas amostras podem vir duplicadas ou repetidas (bootstrapping).

Dessa forma são criadas soluções diferentes usando o mesmo algoritmo, pois para cada árvore ele usa um conjunto diferente de amostras.



Resultado

Combinando soluções:

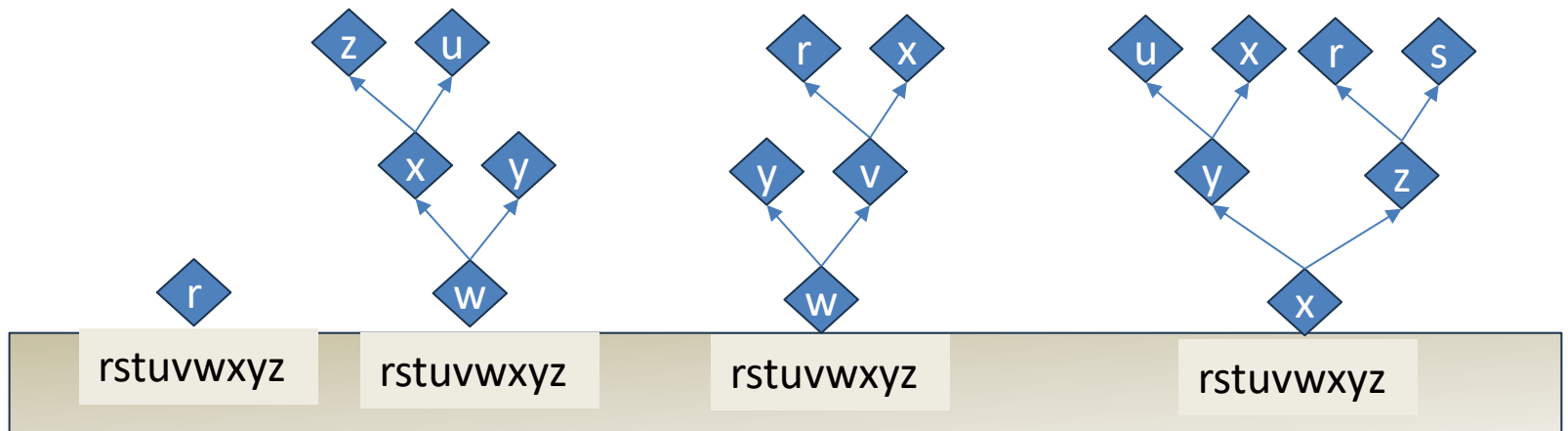
- Cada árvore de decisão apresenta uma solução possível para a tarefa de classificação.
- O conjunto Random Forest combina as previsões das árvores de decisão individuais usando votação por maioria (a classe prevista pela maioria das árvores) para chegar ao modelo final.

Com o método ensemble serão criados vários modelos diferentes a partir de um algoritmo, e uma solução ótima é obtida combinando estas soluções parciais.

Estratégia

- Criam-se diferentes subconjuntos de variáveis e cada um deles é usado para criar uma árvore (solução).
- Algumas árvores tem baixo desempenho, outras são melhores para a solução do problema.
- Quanto mais árvores forem criadas, maior a chance de achar a solução com melhor desempenho

Conjunto original: rstuvwxyz. ... mas com diferentes amostras (bagging)

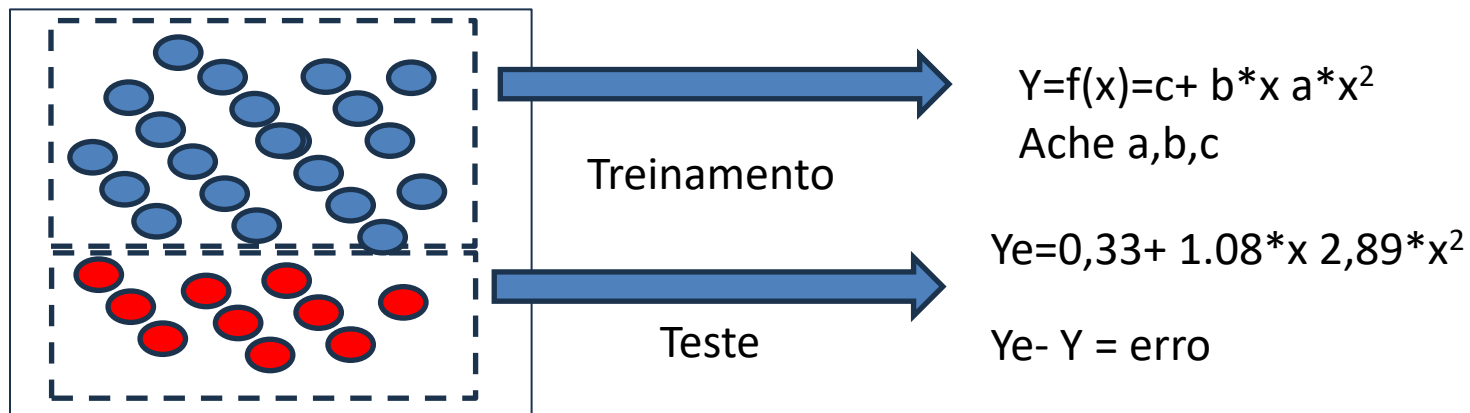


Treinamento/verificação

O algoritmo divide o conjunto de dados em dois subconjuntos:

- Um para treinamento (aprendizado) e
- Outro de teste (aplicação);

As amostras de treinamento servem para achar o modelo de classificação. E a adequação deste modelo é avaliada com novas amostras (conjunto de teste).





Acurácia

Para avaliar a acurácia de um classificador, pode-se usar o índice kappa. O índice kappa é um avaliador estatístico do nível de concordância entre dois resultados (originalmente, dois avaliadores). Aqui é tratada a concordância entre a verdade de campo e o resultado obtido pelo classificador.

Pode ser calculado pela equação:

$$k = \frac{\hat{p}_0 - \hat{p}_e}{1 - \hat{p}_e}$$

Onde:

p_0 é a taxa de aceitação relativa

p_e é a taxa hipotética de aceitação

Obs: quando a concordância é total entre os dois conjuntos de dados $k = 1,0$

Lembrando...

- Matriz de confusão

classificação

verdade

	c_1	c_2	...	c_r
c_1	n_{11}	n_{12}	...	n_{1r}
c_2	n_{21}	n_{22}	...	n_{2r}
...	...	...	...	...
c_r	n_{r1}	n_{r2}	...	n_{rr}

Onde:

- r número de classes
- n número de amostras
- n_{ij} número de amostras classificadas como i que na verdade são da classe j
- $n_{.i}$ quantidade de amostras da classe i na referência
- n_i quantidade de amostras classificadas na classe i

Kappa

- O coeficiente de concordância Kappa, segundo Cohen 1960:

$$k = \frac{\hat{p}_0 - \hat{p}_e}{1 - \hat{p}_e}$$

Onde

$$\hat{p}_0 = \sum_{i=1}^r \frac{n_{ii}}{n}$$

e

$$\hat{p}_e = \sum_{i=1}^r \frac{n_i * n_j}{n^2}$$

n_i e n_j são as somas ao longo de cada linha e cada coluna.

n_{ii} é a soma da diagonal

* $\hat{p}_0$ representa a acurácia global ou a taxa de elementos corretamente classificados

* $\hat{p}_e$ considera os elementos fora da diagonal



Distância de Mahalanobis

É uma medida de distância estatística que calcula a distância entre um ponto e uma distribuição de dados, levando em consideração a variabilidade e a correlação entre as variáveis.

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

Onde:

- $\mathbf{x}$ é o vetor de características do ponto de interesse (neste caso, um pixel ou amostra).
- $\boldsymbol{\mu}$ é o vetor da média das variáveis.
- $\boldsymbol{\Sigma}$ é a matriz de covariância dos dados, que descreve como as variáveis estão relacionadas (correlação e variabilidade).
- $\boldsymbol{\Sigma}^{-1}$ é a matriz inversa de covariância, que ajusta a distância para a variabilidade das variáveis.



Classificador em Distância Mínima

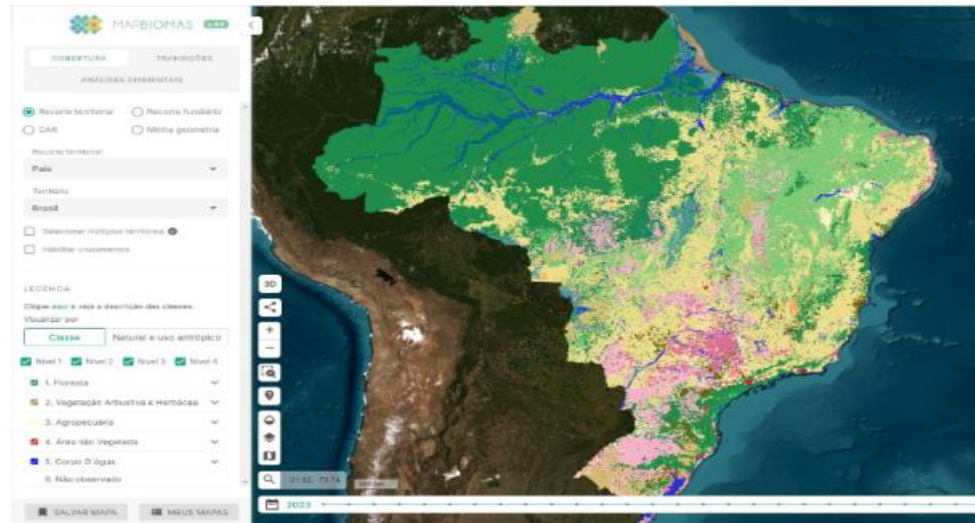
ee.Classifier.minimumDistance()

O tipo de distância a ser calculada entre os pontos de treinamento e as amostras. No GEE, podemos usar três distâncias:

- **Euclidean** - distância Euclidiana.
- **Mahalanobis** - distância de Mahalanobis.
- **SquaredEuclidean** - distância Euclidiana quadrada.

Neste caso, usaremos a distância de Mahalanobis para a atividade prática.

Exemplo de uso



MapBiomas (<https://brasil.mapbiomas.org/>) é uma iniciativa colaborativa com dados publicamente disponíveis que utiliza imagens de satélite (Landsat) e algoritmos de inteligência artificial para mapear e monitorar a cobertura e o uso da terra no Brasil ao longo dos anos.

MAPBIOMAS/GGE

MAPBIOMAS usa o classificador Random Forest (RF) dentro da nuvem de processadores da Google Earth Engine (GEE).

- As amostras necessárias são obtidas por meio de mapas de referência, mapas de classes estáveis das séries anteriores e por coleta direta por interpretação visual das imagens Landsat.
- A classificação é feita para cada um dos anos da série (1985 e 2023).





Laboratório

- Iniciaremos aplicando o código descrito à seguir para classificar uma área utilizando Random Forest e usaremos Kappa para avaliar o classificador Random Forest.
- Em seguida, faremos uma classificação utilizando um classificador em distância mínima:
ee.Classifier.minimumDistance() e vamos comparar os valores.

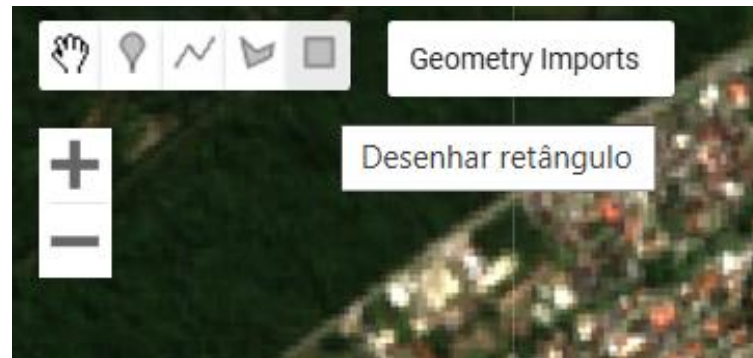


Prática

- GEE/RF – Faremos uma Classificação utilizando Random Forest, no GEE, seguindo os seguintes passos:
 - Definir o local de análise
 - Selecionar e visualizar imagem
 - Coletar amostras para treinar o classificador
 - Aplicar o classificador
 - Avaliar a classificação (Kappa)

Definir área de estudo

- Definir uma "*Geometria*", neste caso, um retângulo.
- Para isto use a opção "*desenhar retângulo*":



- Com isto, apenas esta região será processada.
- As coordenadas da área serão disponibilizadas sob forma de *Geometry* na área de código
- Por isso, a primeira linha do programa é:
`var aoi = geometry // aoi= area of interest`

Visualize a imagem Sentinel

```
var aoi = geometry
```

```
// Copernicus= base de dados de imagens Sentinel
```

```
var s2 = ee.ImageCollection("COPERNICUS/S2_SR")
```

```
.filterDate ('2023-06-01','2023-09-30') // Filtra pela data
```

```
// Filtra imagens com menos de 3% de nuvens
```

```
.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 3))
```

```
.filterBounds(aoi) // Filtra pela área de interesse
```

```
.mean() // Calcula a média da coleção
```

```
.multiply(0.0001) // Ajusta os valores para a escala 0-1 (normalizar)
```

```
// Recortar a imagem com a geometria da área de interesse (AOI)
```

```
var clippedImage = s2.clip(aoi);
```

```
// Selecionar as bandas RGB (Red = B4, Green = B3, Blue = B2 no Sentinel-2)
```

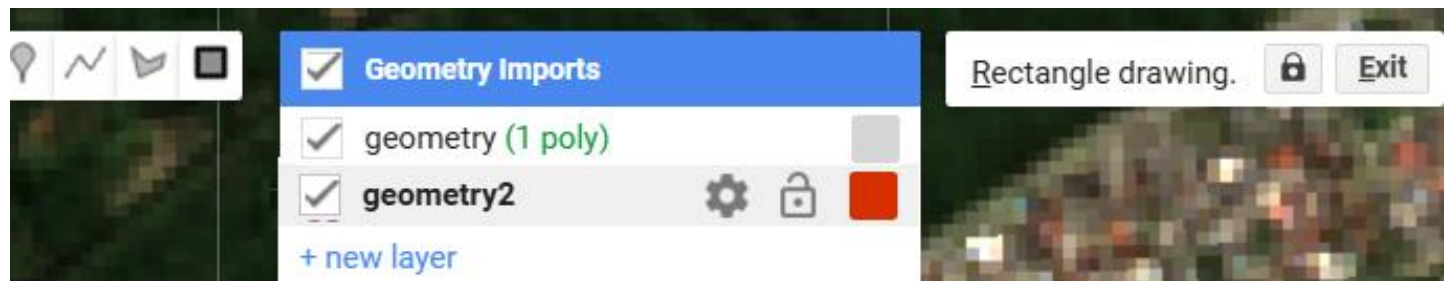
```
var rgbS2 = clippedImage.select(['B4', 'B3', 'B2']);
```

```
// Exibir a imagem RGB no mapa
```

```
Map.addLayer(rgbS2, {min: 0.1, max: 0.3}, 'Composição RGB Sentinel-2');
```

Coletar amostras

- Usando a ferramenta de "*geometry imports*" adicione uma nova classe (*new layer*).
- Por default, apresenta uma geometria padrão (*geometry*) que deve ser editada usando a roda dentada:



Editar a geometria

- **Name:** Nome da classe
- **Import as:** FeatureCollection (isto adiciona a área para uma coleção de amostras)
- **Properties**
 - **classe** (define que a região é uma "classe" em nosso modelo)
 - **Value:** número da classe (1,2,3,...)
 - **rotulo** (na legenda): "nome na legenda"
- Pode selecionar a cor
- OK para salvar.

- Colete amostras para várias classes que serão classificadas.

Pode selecionar várias áreas para a mesma classe, usando o mesmo "nome"!

Configure geometry import

Name

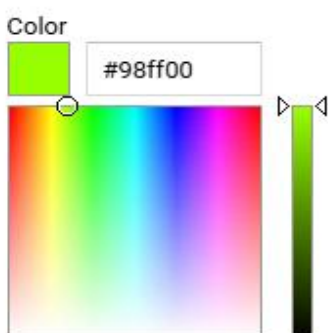
Import as

Properties

Property	Value
<input type="text" value="classe"/>	<input type="text" value="01"/>
<input type="text" value="rotulo"/>	<input type="text" value="floresta"/>

[+ Property](#)

Color



OK Cancel 

Código

- Após a seleção, o GEE mostra a coleção de amostras no início do script, como *FeatureCollection*, para cada classe.

Imports (7 entries) 

```
▸ var geometry: Polygon, 4 vertices    
▸ var floresta: FeatureCollection (41 elements)    
▸ var oceano: FeatureCollection (41 elements)    
▸ var urbana: FeatureCollection (41 elements)    
▸ var industria: FeatureCollection (41 elements)    
▸ var restinga: FeatureCollection (41 elements)    
▸ var areia: FeatureCollection (41 elements)  
```

```
1 var aoi = geometry
```



Preparação

Para iniciar, devemos combinar todas as amostras das classes desejadas em um conjunto (merge):

// Combinar todas as amostras em uma única coleção

```
var amostras =  
    floresta.merge(oceano).merge(urbana).merge(industria).merge(rest  
    inga).merge(areia);
```

E podemos selecionar as bandas que usaremos:

// Selecionar as bandas

```
var bandas = [ 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12'];
```

Bandas do Sentinel-2

Resolução	Nr da Banda	Nome da Banda	Comprimento de Onda Central (nanômetro)	Combinações de Bandas
10 m	B02	Blue (Azul)	490	Cor Verdadeira RGB 04/03/02 Falsa Cor 1 e 2 RGB 08/04/03 e 04/08/03
	B03	Green (Verde)	560	
	B04	Red (Vermelho)	665	
	B08	NIR (Infravermelho Próximo)	842	
20 m	B05	Red Edge 1	705	SWIR 1 RGB 12/11/8A
	B06	Red Edge 2	740	
	B07	Red Edge 3	783	
	B08A	Red Edge 4	865	
	B11	SWIR 1	1610	
	B12	SWIR 2	2190	
60 m	B01	Aerossol	443	—
	B09	Water Vapor	940	
	B10	Cirrus	1375	



Ler valores das amostras

// Ler valores das Amostras para o treinamento

```
var treinamento =
```

```
clippedImage.select(bandas).sampleRegions({
```

```
  collection: amostras,
```

```
  properties: ['classe'], // A propriedade 'classe' contém as  
  etiquetas de classe
```

```
  scale: 10 // Escala das imagens
```

```
});
```



Criar classificador (Modelo)

```
var classificador_RF = ee.Classifier.smileRandomForest({  
  numberOfTrees: 150,  
  seed: 42  
}).train({  
  features: treinamento,  
  classProperty: 'classe'  
})
```

- **Defina o número de árvores**
- **Seed= inicializador de série aleatória**



Classificar

```
//Classificar pelo Classificador Random Forest definido
```

```
var classificacao = clippedImage.classify(classificador_RF)
```

```
// Visualizar o resultado
```

```
Map.addLayer(classificacao, {min:1, max:6, palette:  
  ['green', 'blue', 'red', 'magenta', 'yellow', 'cyan']},  
  'Classificacao')
```

Obs: seis classes necessitam seis cores na paleta e assim para todas as classes criadas.

Cores disponíveis para o GEE

Colors	Cores	Colors	Cores
Brown	Marrom	Lime	Lima
Beige	Bege	Magenta	Magenta
Blue	Azul	MediumBlue	Azul médio
BlueViolet	Azul Violeta	Olive	Verde azeitona
Chocolate	Marrom escuro	Orange	Laranja
Coral	Coral	PaleGreen	Verde pastel
ForestGreen	Floresta Verde	Purple	Roxo
Fuchsia	Fúcsia	Violet	Violeta

Calcular Kappa

Para calcular o Kappa, primeiramente construiremos uma matriz de confusão, com os seguinte passos:

// Obter as amostras de classificação (aquelas que foram usadas para treinamento)

```
var amostrasClassificadas =  
    clippedImage.select(bandas).sampleRegions({  
        collection: amostras,  
        properties: ['classe'], // Classe verdadeira  
        scale: 10  
    });
```

// Obter a classificação prevista para cada amostra

```
var amostrasComClassePreditada =  
    amostrasClassificadas.classify(classificador_RF);
```



Calcular Kappa

//Gerar a Matriz de Confusão

```
var matrizConfusao = amostrasComClassePreditas.errorMatrix('classe',  
    'classification');
```

```
print('Matriz de Confusão Random Forest:', matrizConfusao); // Exibir a  
    Matriz de Confusão
```

// Calcular o Índice Kappa

```
var kappa = matrizConfusao.kappa();  
print('Índice Kappa Random Forest:', kappa);
```



Opcional: Acurácia Geral

Para calcular a acurácia geral, faremos os seguintes passos:

```
// Opcional: Calcular a acurácia geral
```

```
var acuraciaGeral = matrizConfusao.accuracy();
```

```
print('Acurácia Geral:', acuraciaGeral);
```



VERIFICAÇÃO

Colete novas amostras para cada classe. Mantenha o número de amostras parecido para cada classe: Amostras de VERIFICACAO

- Florestav
- Oceanov
- Urbana
- Industriav
- Restingav
- Areiav

// Combinar todas as amostras de verificação em uma única coleção

var AmostrasTeste =

```
    florestav.merge(oceanov).merge(urbanav).merge(industriav).merge  
    (restingav).merge(areiav);
```



Amostras para teste

Repita o processo de coleta de amostras, mas desta vez, crie um conjunto separado, para teste. Com estes dados pode calcular a matriz de confusão.

// Ler valores das Amostras para o teste

```
var AmostrasTesteClassificadas =  
clippedImage.select(bandas).sampleRegions({  
  collection: AmostrasTeste,  
  properties: ['classe'], // A propriedade 'classe' contém as etiquetas de  
  classe  
  scale: 10 // Escala das imagens  
});
```

// Obter a classificação prevista para cada amostra

```
var TesteCom Classe = AmostrasTesteClassificadas.classify(classificador_RF);
```



Kappa para verificação

//Gerar a Matriz de Confusão

```
var matrizConfusaoV = TesteCom Classe.errorMatrix('classe',  
    'classification');
```

// Exibir a Matriz de Confusão

```
print('Matriz de Confusão Random Forest:', matrizConfusaoV);
```

// Calcular o Índice Kappa

```
var kappa = matrizConfusaoV.kappa();  
print('Índice Kappa Random Forest:', kappa);
```

// Calcular a acurácia geral

```
var acuraciaGeral = matrizConfusaoV.accuracy();  
print('Acurácia Geral:', acuraciaGeral);
```



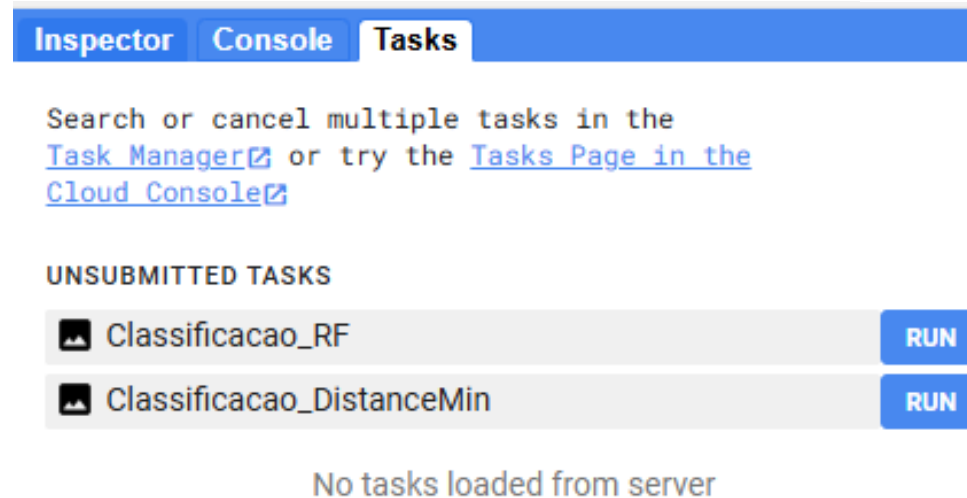
Exportar resultados para Google Drive

Exportar a imagem classificada para o Google Drive:

```
Export.image.toDrive({  
  image: classificacao, // A imagem classificada que você deseja exportar  
  description: 'Classificacao_RF', // Nome do arquivo exportado  
  scale: 10, // Resolução espacial (em metros)  
  region: aoi, // A região de interesse (a geometria definida)  
  fileFormat: 'GeoTIFF', // Formato de saída  
  crs: 'EPSG:4326', // Sistema de referência de coordenadas (CRS) - 4326  
    (WGS84) é padrão para dados geoespaciais  
  maxPixels: 1e8 // Limite para o número máximo de pixels exportados  
    (ajuste conforme necessário)  
});
```

Exportar resultados para Google Drive

Após a task ser gerada, iremos realizar o download em 'run':



Exportar resultados para Google Drive

Realizar o download no drive:

Task: Initiate image export

Task name (no spaces) *

Classificacao_RF

Coordinate Reference System (CRS)

EPSG:4326

Scale (m/px)

10

DRIVE

CLOUD STORAGE

EE ASSET

Drive folder

Drive folder name or blank for root

Filename *

Classificacao_RF

File format *

GEO_TIFF

CANCEL

RUN



Criar classificador Distância mínima

// Treinando o classificador "minimumDistance" com a
métrica de distância mahalanobis

```
var classificador2 = ee.Classifier.minimumDistance({  
  metric: 'mahalanobis', // Tipo de métrica  
}).train({  
  features: treinamento,  
  classProperty: 'classe'  
});
```

- **Métrica - Usaremos a distância de Mahalanobis.**



Criar classificador Distância mínima

// Classificar a imagem com o classificador treinado

```
var classificacao2 = clippedImage.classify(classificador2);
```

// Visualizar o resultado

```
Map.addLayer(classificacao2, {  
  min: 1,  
  max: 6,  
  palette: ['green', 'blue', 'red', 'magenta', 'PaleGreen',  
    'yellow']  
}, 'Classificação Mínima Distância');
```



Calcular Kappa para Classificação em distância mínima

Para calcular o Kappa, primeiramente construiremos uma matriz de confusão, com os seguinte passos:

// Obter as amostras de classificação (aquelas que foram usadas para treinamento)

```
var amostrasClassificadas2 =  
    clippedImage.select(bandas).sampleRegions({  
        collection: amostras,  
        properties: ['classe'], // Classe verdadeira  
        scale: 10  
    });
```

// Obter a classificação prevista para cada amostra

```
var amostrasComClassePreditada2 =  
    amostrasClassificadas2.classify(classificador2);
```



Calcular Kappa para Classificação em distância mínima

Para calcular o Kappa, primeiramente construiremos uma matriz de confusão, com os seguinte passos:

// Gerar a Matriz de Confusão

```
var matrizConfusao2 =  
    amostrasComClassePedita2.errorMatrix('classe',  
    'classification');  
print('Matriz de Confusão Distância Mínima:', matrizConfusao2); //  
    Exibir a Matriz de Confusão
```

// Calcular o Índice Kappa

```
var kappa = matrizConfusao2.kappa();  
print('Índice Kappa Distância Mínima:', kappa);
```

// Opcional: Calcular a acurácia geral

```
var acuraciaGeral = matrizConfusao2.accuracy();  
print('Acurácia Geral Distância Mínima:', acuraciaGeral);
```

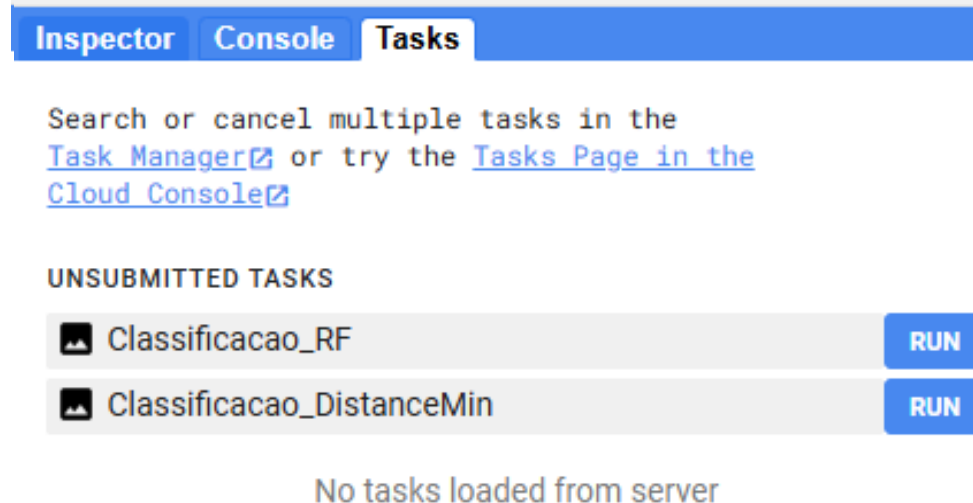
Exportar resultados para Google Drive

Exportar a imagem classificada para o Google Drive:

```
Export.image.toDrive({  
  image: classificacao2, // A imagem classificada que você deseja  
                        exportar  
  description: 'Classificacao_DistanceMin', // Nome do arquivo  
                        exportado  
  scale: 10, // Resolução espacial (em metros)  
  region: aoi, // A região de interesse (a geometria definida)  
  fileFormat: 'GeoTIFF', // Formato de saída  
  crs: 'EPSG:4326', // Sistema de referência de coordenadas (CRS) - 4326  
                (WGS84) é padrão para dados geoespaciais  
  maxPixels: 1e8 // Limite para o número máximo de pixels exportados  
                (ajuste conforme necessário)  
});
```

Exportar resultados para Google Drive

Após a task ser gerada, iremos realizar o download em 'run':



The screenshot shows the 'Tasks' tab in the Google Cloud console. It includes a search bar, a list of 'UNSUBMITTED TASKS', and a 'No tasks loaded from server' message.

Task Name	Action
Classificacao_RF	RUN
Classificacao_DistanceMin	RUN

No tasks loaded from server

Exportar resultados para Google Drive

Realizar o download no drive:

Task: Initiate image export

Task name (no spaces) *

Classificacao_DistanceMin

Coordinate Reference System (CRS)

EPSG:4326

Scale (m/px)

10

DRIVE

CLOUD STORAGE

EE ASSET

Drive folder

Drive folder name or blank for root

Filename *

Classificacao_DistanceMin

File format *

GEO_TIFF

CANCEL

RUN

TAREFA

- Comparar o resultado (acurácia medida pelo índice Kappa) de dois classificadores (Distância mínima e RF) da região de pontal do Paraná.
- Imagem: Sentinel-2B de Junho de 2023, com resolução espacial de 10 m.

O código completo pode ser consultado nesse link:

<https://docs.google.com/document/d/1pXOF9sZj2YPsk28CUc8vkjhBonDJ69u8Yoh9aMjOuRE/edit?usp=sharing>