

## 5. P5 - Beeldclassificatie

### 5.1 Practicum 5: Beeldclassificatie

#### Doelstelling

In dit practicum verdiepen we ons in het proces van de beeldclassificatie, waarbij manieren om remote sensing beelden te vertalen in landgebruikskaarten. Als voorbeeld nemen we een classificatie van mangroves.

#### Beeldclassificatie: introductie

Beeldclassificatie (Engels: '*Image Classification*') is de (complex) procedure waarbij een multiband rasterbeeld wordt ingedeeld in klassen om hieruit spatiale informatie te ontrekken. Het is wellicht de meest belangrijke handeling bij digitale beeldanalyse. Het resulterende beeld kan gebruikt worden om thematische kaarten aan te maken.

Gedurende de classificatie worden pixels ingedeeld in groepen, op basis van hun gemeenschappelijke karakteristieken. Er bestaan twee grote groepen van classificatiemethoden: *Supervised* of gesuperviseerde en *Unsupervised* of niet-gesuperviseerde classificatie.

In een **niet-gesuperviseerde** classificatie deelt de computer pixels in die spectraal gezien gelijkend zijn aan elkaar in een aantal klassen. Dit aantal kan op voorhand worden vastgelegd, of in de classificatieprocedure worden bepaald. Belangrijk is dat de analyse wordt uitgevoerd zonder dat de gebruiker hiervoor voorbeeldklassen of trainingsdata als input moet voorzien. De enige noodzakelijke input door de gebruiker betreft de keuze van het algoritme en eventueel het aantal gewenste resulterende klassen.

In een **gesuperviseerde** classificatie wordt de classifier *getrained* op basis van enkele voorbeeldklassen of *training data*. Nieuwe, ongeklasseerde data wordt dan aan het algoritme voorgelegd, die dit dan op basis van het getrainde algoritme gaat indelen. In een finale stap wordt dan een andere portie van gekende data, de *validatiedata*, gebruikt om de accuraatheid

(*accuracy*) van de classificatie kwantitatief te bepalen. In dit type classificatie is het aantal resulterende klassen eveneens bepaald op basis van het aantal klassen meegegeven tijdens het trainen van de data.

#### Supergeviseerde vs niet-gesuperviseerde classifiers

Niet-gesuperviseerde classificatie	Gesuperviseerde classificatie
+	+
Er is geen voorkennis nodig van het gebied.	De trainingsklassen komen overeen met de eigenlijke landbekking
Minimale input van menselijke fouten	Trainingdata is herbruikbaar, tenzij de landbedekking verandert.
Gemakkelijk uit te voeren	Resultaat is meteen bruikbaar, gezien de klassen gekend zijn.
-	-
De resulterende klassen komen niet per sé overeen met gewenste landbekkingsklassen	De opgegeven klassen komen niet altijd overeen met de spectrale klassen
Spatiale relaties of spatiale context wordt niet meegenomen in de data	Zekerheid/consistentie is niet over alle klassen gelijk
Interpretatie kan moeilijk zijn	Verzamelen van (voldoende) trainingdata kost moeite, geld en tijd

## 5.2 CASE - Mangrove monitoring in Suriname

---

### Over Mangroves

#### Defenitie en verspreiding

Het **Mangrove-ecosysteem** dat kan gezien worden als een soort moerasecosysteem, gedomineerd door mangrovebomen/bossen. Het Mangrove-ecosysteem is de zone die zich doorgaans bevindt het intergetijdenzone tussen zee en land in zowel tropische als subtropische gebieden. Binnen dit Mangrove-ecosysteem bevinden zich de Mangrovebomen zelf ('*True Mangroves*') en de geassocieerde soorten (*Mangrove associates*).

Hoewel er geen vastgelegde definitie bestaat van mangrovebossen, kan deze van Tomilson (2016) worden gezien als de meest geaccepteerde:

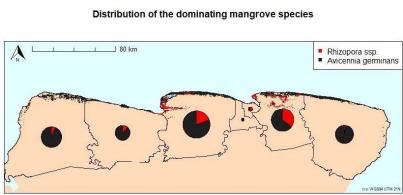
**True Mangroves** are plant species that:

1. Occur only in mangrove forests and are not found in terrestrial communities
2. Play a major role in the structure of the mangrove community, sometimes forming pure stands.
3. Have morphological specialisations to the mangrove environment
4. Have some mechanism for salt exclusion

In totaal bestaan er zo'n 55 mangrovesoorten die aan bovenstaande definitie voldoen, waarvan de hoogste soortendiversiteit zich voordeet rond het westelijk deel van de Stille Oceaan. Globale inschattingen van dit ecosysteem variëren tussen 110.000 tot 240.000 km<sup>2</sup> (Giri, 2016).

#### Mangroves in Suriname

In Suriname bestaan er "slechts" 5 mangrovesoorten waarvan 2 soorten de grootste dominante vertonen: de zwarte mangrove of 'Parwa' (*Avicennia germinans*) die hoofdzakelijk langs de kustlijnen groeit in homogene bestanden en de rode mangrove of 'Mangro' (*Rhizophora mangle*), die landinwaarts en langs rivieroever terug te vinden is.



Verspreiding van de 2 dominerende Mangrovesoorten in 2018 (SBB, 2019).



## Mangrove mapping

### LU/LC klassen

In de voorbeeld voor beeldclassificatie gaan we zelf een LU/LC-kaart maken voor de kustzone ter hoogte van Paramaribo, de hoofdstad van Suriname. Hierin gaan we onderscheid maken tussen volgende klassen, zodat we Mangroves kunnen onderscheiden:

1. Mangrove
2. Ander bos
3. Water
4. Crops/grasvegetatie
5. Urban (stedelijke omgeving)
6. Naakte Bodem (soil)

### Onderscheiden van Mangroves op satellietbeelden

Als laatste voorbereidende stap gaan we na op welke manier we Mangroves (visueel) kunnen onderscheiden van de andere klassen (met in het bijzonder aangrenzend inlands bos).

### Healthy Vegetation Composite

Het aanmaken van verschillende beeldcomposieten helpt om visueel de verschillende landbedekkingsklassen te onderscheiden. Naas de Normale Kleurencomposit (gemakkelijkst interpreerbaar voor het menselijk oog) en de Valse Kleurencomposit (benadrukking verschil vegetatie - andere klassen) kan ook de *healthy vegetation composite* nuttig zijn.

Deze composiet is gemaakt met volgende banden: RGB = NIR, SWIR en Blauw, wat met de resepctievelijke Sentinel-2 banden *B8, B11, B2* kan worden aangemaakt.

In deze composiet kleurt 'gezonde vegetatie' met toetsen van rood, oranje en geel. Doordat de SWIR-band wordt gebruikt, kunnen verschillende fasen van plantengroei en/of stress worden nagegaan. Mangrovebossen, kan op deze composiet worden onderscheiden van andere bossen door hun specifieke watergebonden milieu.

### Aanmaken van Beeldcomposieten

Maak een 'Healthy Vegetation'-composit aan van de Surinaamse kustlijn. Bekijk met welke stretch de mangrovezone het best tot uiting komt.

### Mangrove Vegetation Index (MVI)

Doorheen de jaren werden ook enkele specifieke Mangroves indices ontwikkeld, om de detectie van Mangroves te vergemakkelijken. Een goed voorbeeld is de '**Mangrove Vegetation Index**' of **MVI** ([Baloloy, 2020](#)), dat recentelijk werd ontwikkeld op basis van de Sentinel-2 banden B2, B8, B11, die de specifieke 'groenheid' en vochtomstandigheid die de mangrovebossen typeert in de verf zet, om zo een onderscheid te kunnen maken met de naburige terrestrische bossen en vegetaties.

De MVI wordt als volgt geformuleerd:

$$\text{MVI} = \{ \text{NIR} - \text{Green} \over \text{SWIR1} - \text{Green} \}$$

Wat zicht vertaald naar Sentinel-2 banden als:

$$\text{MVI} = \{ \text{B8} - \text{B3} \over \text{B11} - \text{B3} \}$$

## 5.3 Niet-gesuperviseerde classificatie

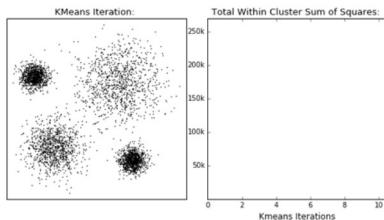
*Niet-gesuperviseerde* classificatie is de procedure waarbij pixel worden ingedeeld volgens spectraal gelijkende klassen zonder dat hierbij trainingsklassen nodig zijn. De resulterende klassen worden dan ook *spectrale klassen* genoemd.

Het is aan de gebruiker om de resulterende klassen te interpreteren en te labelen.

Er bestaan verschillende niet-gesuperviseerde algoritmen, maar de meest gekende groep is deze van de *clustering*. In een clusteranalyse worden pixels met gelijkende spectrale kenmerken tot dezelfde klasse gerekend.

### K-Means Clustering

Een van de meest gebruikte cluster-algoritmen is de "K-means" clustering, waarbij de gebruiker op voorhand een aantal beginclusters opgeeft waarmee het algoritme arbitrair dat aantal clusters in de multi-dimensionale ruimte. Elke pixel wordt dan in een eerste fase toegekend tot de cluster waar de pixels zich gemiddeld het dichtst bij bevindt. Na deze eerste '*run*' worden de clusters herberekend, waarbij de variantie binnen elke cluster wordt geminimaliseerd. Hierna worden de pixels opnieuw toegekend tot de 'best passende' cluster. Deze procedure wordt herhaald (iteraties) totdat er zich geen significante verplaatsing van de clustercentra meer voordoet en de variantie binnen elke cluster dus ook niet meer significant daalt.



Principe van de K-means clustering in een 2-dimensionaal vlak. (Bron: [dashee87.github.io](https://dashee87.github.io))

### Classificatie van de Surinaamse kustzone

In Earth Engine zit de clustering ook vervat in `ee.Clusterer`. We maken in volgend voorbeeld gebruik van de weka k-means cluster.

- Maak een nieuw script aan: P6\_UnsupervisedClass.
- Zoals steeds filteren en reducere we een satellietbeeld, om met een wolkenvrije `image` verder te kunnen werken. We focussen ons in dit voorbeeld op de kustlijn van Suriname, ter hoogte van de hoofdstad: Paramaribo. We focussen hierbij op maanden binnen de grote droge tijd (Augustus - December), aangezien de wolkbedekking dan beperkter zou moeten zijn in vergelijking met de natte tijden.

Maak hiervoor eerst een polygoon aan met de locatie van Paramaribo.

Eventueel kun je hiervoor onderstaande code gebruiken:

```
var Paramaribo =
/* color: #d63000 */
/* shown: false */
/* displayProperties: [
{
  "type": "rectangle"
}
] */
ee.Geometry.Polygon(
  [[[[-55.31615692285674, 6.000339363352038],
  [-55.31615692285674, 5.8043169248564865],
```

```
[-54.91446930078643, 5.8043169248564865],
[-54.91446930078643, 6.000339363352038]]], null, false);
```

Start Daarna met het aanmaken van het S2-beeld

```
//
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
// -----
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
(sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-service-6e5b2cb2ce8a
var getS2_SR_CLOUD_PROBABILITY = function () {
  var innerJoined = ee.Join.inner().apply({
    primary: ee.ImageCollection("COPERNICUS/S2_SR"),
    secondary: ee.ImageCollection("COPERNICUS/
S2_CLOUD_PROBABILITY"),
    condition: ee.Filter.equals({
      leftField: 'system:index',
      rightField: 'system:index'
    })
  });
  var mergeImageBands = function (joinResult) {
    return ee.Image(joinResult.get('primary'))
      .addBands(joinResult.get('secondary'));
  };
  var newCollection = innerJoined.map(mergeImageBands);
  return ee.ImageCollection(newCollection);
}

// Mask out clouds
var maskClouds = function(image) {
  var cloudProbabilityThreshold = 40;
  var cloudMask =
image.select('probability').lt(cloudProbabilityThreshold);
  return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van de kustlijn met
mangroves en de hoofdstad Paramaribo, Suriname

var S2_coll = getS2_SR_CLOUD_PROBABILITY()
  .filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
  .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',50) ////
Voorselectie obv wolken
  .map(maskClouds) //toepassen van de cloudmaskfunctie
  .filterBounds(Paramaribo); //collectie filteren obv de
Kustzonegeometrie

//Omzetten collectie naar een Image, door de.median() reducer toe te
passen. Hierna clippen we ook tot onze ROI
var S2_im = S2_coll.median()
  .clip(Paramaribo) //Bekijk de .clip-eigenschappen
in de Docs
```

### De `.clip()` functie

De `.clip()`-functie wordt toegepast om het resulterende beeld bij te snijden naar de exacte grenzen van de aangemaakte polygoon (ROI).

`.clip()` is enkel toepasbaar op beelden van het `image`-type, maar kan niet worden toegepast op een `ImageCollection`, gezien dit een te grote rekencapaciteit zou vergen. Daarom wordt de functie `.FilterBounds()` gebruikt, waarbij enkel gefilterd wordt op basis van een spatiale feature (punt, lijn of polygoon) maar geen beelden worden bijgesneden.

- De Earth Engine clusterer volgt het algoritme van [Weka](#), een open-source machine learning softwarepakket. In dit algoritme worden eerst pixels uit het beeld 'gesampled', waarop het k-means algoritme wordt losgelaten. Eenmaal het algoritme op punt is, wordt het toegepast op de rest van het beeld. M.a.w. wordt er een niet-gesuperviseerd model getrained op een willekeurige sample van pixels, die representatief wordt geacht voor de rest van het beeld. Het aantal te sampelen pixels moet dus voldoende groot gekozen worden, maar indien het te groot wordt, zal earth engine een foutmelding geven: `clusters: Layer error: Computed value is too large.` De maximale capaciteit van earth engine ligt bij ca. 1 miljoen pixels.

```
// Aanmaken "training" dataset.
var training = S2_im.sample({
  region: Paramaribo,
  scale: 10,
  numPixels: 5000
});
```

- Eenmaal de pixels geselecteerd zijn die gebruikt gaan worden voor het aanmaken van de k clusters, kan het 'K-means cluster-algoritme' worden getraind. Deze bevat enkele parameters die de gebruiker nog kan instellen:

- a. **nClusters**: het eerste en het enige verplicht aan te geven argument dat het aantal gewenste clusters aangeeft.
- b. **init**: de initialisatiemethode. Hiermee kan de manier waarop de initiële clustercentra worden gekozen. De *default*-instelling kiest ad random de beginpunten. Deze zullen we in deze oefening gebruiken.
- c. **distanceFuntion**: welke afstandsberekening moet worden toegepast: *Euclidisch* of *Manhattan*. De euclidische afstand is *default*.
- d. **maxIterations**: het maximale aantal iteraties, indien opgegeven. De clusteranalyse stopt na dit aantal iteraties.

```
// Clusterer opstellen en trainen. Opgeven van 5 klassen, de andere parameters laten we op default.
var Kmeans_cl = ee.Clusterer.wekaKMeans(5).train(training);

// Laat de cluster los op het volledige beeld
var classified = S2_im.cluster(Kmeans_cl);
```

\* Finaal visualiseren we ook het resultaat. Om de bekomen klassen snel een afzonderlijke kleur te geven, maken we gebruik van `.randomVisualizer()`.

```
// Display the clusters with random colors.
Map.addLayer(classified.randomVisualizer(), {}, 'clusters');
```



Voorbeeldresultaat van de K-means clustering.

### Opdrachten

1. Interpreteer de bekomen klassen en tracht ze te linken aan landbedekkingsklassen. Gebruik hiervoor ook een Normale Kleuren, Valse Kleuren en 'Healthy Vegetation' (RGB = B8,B11,B2) composiet.
2. Herhaal bovenstaande clustering enkele keren, met onderstaande parameters. Vergelijk de resultaten ook steeds met elkaar:
  - Je het aantal clusters optrekt naar 10.
  - Je de parameter `maxIterations` binen de `ee.ClustererwekaKMeans()` functie toevoegd. Test de waarden 1, 10 en 30
  - Je de factor `numPixels` vergroot (naar bv 10000).

## 5.4 Gesuperviseerde classificatie

### Intro

#### Achtergrond

**Gesuperviseerde classificatie** is gebaseerd op een door de gebruiker opgestelde trainingdataset. Deze trainingsdata zijn representatief voor de specifieke gewenste klassen. Het gesuperviseerde algoritme gebruikt dan deze trainingsdata als referentie, om onbekende pixels te classificeren.

Een belangrijk element hierbij is dus voorkennis van het desbetreffende gebied. De *trainingsamples* die worden opgesteld worden ook wel *ground truth data* genoemd en komen overeen met gekende landbekking op locaties.

Er zijn verschillende opties om aan gronddata te komen:

1. Door middel van veldbezoek, waarbij **GPS-data** ter plaatse worden verzameld. Deze methode zorgt voor de meest kwalitatieve data en zekerheid, maar is vanzelfsprekend duur en arbeidintensief om uit te voeren.
2. Door gebruik van **referentiekaarten** of oudere beschikbaar kaartmateriaal of **beeldmateriaal van hogere resolutie**, bijvoorbeeld beelden bekomen gedurende een drone-campagne.
3. Door manuele **interpretatie van de satellietbeelden**, waarbij de gebruiker beschikt over enige expertkennis en voorgaande ervaring.

Een ander aspect is dat de trainingdata zo dicht mogelijk opgenomen wordt bij het tijdstip van opname van het gebruikte remote sensing beeld. Landbedekking kan immers snel veranderen: stadsontwikkeling, ontbossing, seizoенale impact, agrarische veranderingen, kusterosie, ...

In deze oefening van gesuperviseerde classificatie zul je zelf trainingsamples moeten aanmaken, want er zijn geen GPS-punten beschikbaar. Wel beschik je over een beknopt verslag van een veldcampagne, dat je een idee kan geven van de aanwezige landbekkingklassen en aangezien we ons beperken tot enkele brede klassen, zul je tevens gemakkelijk visueel trainingsamples kunnen aanmaken. Hiervoor is het aanmaken en analyseren van verschillende composieten aangewezen.

### Beeldcomposieten aanmaken

- Start opnieuw met het aanmaken van een wolkenvrije dataset, dewelke kan overgenomen worden van vorige oefening. Belangrijk hier is ook om de gewenste banden te selecteren, die we tijdens de classificatie gaan aanmaken.

```
//
-----
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
//
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
(sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-
service-6e5b2cb2ce8a
```

```
var getS2_SR_CLOUD_PROBABILITY = function () {
  var innerJoined = ee.Join.inner().apply({
    primary: ee.ImageCollection("COPERNICUS/S2_SR"),
    secondary: ee.ImageCollection("COPERNICUS/
S2_CLOUD_PROBABILITY"),
    condition: ee.Filter.equals({
      leftField: 'system:index',
      rightField: 'system:index'
    })
  });
  var mergeImageBands = function (joinResult) {
    return ee.Image(joinResult.get('primary'))
      .addBands(joinResult.get('secondary'));
  };
  var newCollection = innerJoined.map(mergeImageBands);
  return ee.ImageCollection(newCollection);
}

// Mask out clouds
var maskClouds = function(image) {
  var cloudProbabilityThreshold = 40;
  var cloudMask =
  image.select('probability').lt(cloudProbabilityThreshold);
  return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van Mangroves
Paramaribo, Suriname
var S2_coll = getS2_SR_CLOUD_PROBABILITY()
  .filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
  .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than', 50) //Voorselectie obv wolken
  .map(maskClouds) //toepassen van de cloudmaskfunctie
  .filterBounds(ROI); //collectie filteren obv de
Kustzonegeometrie
print(S2_coll)

//Omdoen collectie naar een Image, door .median() te nemen. Hierna
clippen we ook tot onze ROI
//Ook selecteren we de banden waarmee we verder willen werken
var bands = ['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12'];

var S2_im = S2_coll.median()
  .select(bands)
  .clip(ROI) //Bekijk de .clip-eigenschappen in de
Docs

Map.centerObject(S2_im, 11)
Map.addLayer(S2_im,{min:50,max:
1800,bands:'B4,B3,B2'},'NormaleKleuren_2020',0)
Map.addLayer(S2_im,{min:700,max:
4500,bands:'B8,B4,B3'},'FalseKleuren_2020',0)
Map.addLayer(S2_im,{min:500,max:
4000,bands:'B8,B11,B2'},'Healthy_Vegetation_2020',0)
```

## Trainingsamples aanmaken

"Whereas the actual classification of multispectral image data is a highly automated process, assembling the training data needed for classification is anything but automatic. In many ways, the training effort required in supervised classification is both an art and a science" **Lillesand & Kiefer** (pg 544)

- Nadat het wolkenvrije Sentinel-2 beeld is ingeladen, kunnen we deze gebruiken om enkele representatieve *samples* te verzamelen van enkele landbedekkingklassen waar we in geïnteresseerd zijn. Er zijn 2 manieren om trainingsdata in Earth Engine op te laden:
  - a. Door ze in te tekenen als polygonen binnen per klasse, zoals we in komend voorbeeld zullen toepassen.
  - b. Door eerder ingetekende trainingssamples of GPS-punten op te laden als een 'Asset'. Dit kunnen *shapefiles*, of .csv-bestanden zijn.
- Hover met je muis over de '*Geometry Imports*' box, dat zich naast de geometrietools bevindt. Klik op \*'+new layer'.
- Elke gewenste landbedekkingsklasse dient als een afzonderlijke laag te worden aangemaakt. Laat ons bijvoorbeeld starten met de eenvoudigste klasse 'water'. Zoom in op het beeld en teken polygonen in over oppervlaktes waar je zeker van bent dat het waterlichamen betreft. Het is goed hierin te variëren binnen verschillende types van zowel zee als rivieren en andere waterlichamen. Teken ca. 10 polygonen in per klasse. Neem hiervoor zeker je tijd, gezien het belangrijk is dit zeer precies te doen. De kwaliteit van de inputdata bepaalt tevens de kwaliteit van de classificatie, oftewel *Garbage in = Garbage out*. Als je een fout gemaakt heb, kun je even op 'exit' duwen en de laats ingetekende polygoon verwijderen.

```
| class | Landbekkingsklasse | :----:|-----:| 1 | Mangrove | | 2 |
OtherForest | | 3 | Water | | 4 | Crop | | 5 | Urban | | 6 | BareSoil |
```



### Polygonen vs puntdata als traingdata

Trainingdata kan bestaan uit puntdata of trainingdata. Beiden hebben hun voor- en nadeel.

**Puntdata** gebruiken als inputdata kan bijvoorbeeld als je beschikt over een grote set GPS-veldpunten van locaties waar je exact weet tot welke landbedekkingsklasse een pixel hoort. Deze pixel wordt dan als referentie aanschouwt. Deze zekerheid van de trainingsdata is dus groot. Een nadeel bij pixels is dat het minder de variëteit van de pixels binnen de klasse opneemt en de totale set aan trainingspixels beperkt blijft.

**Polygonen** worden gebruikt om gebieden in te tekenen voor een bepaalde klasse. In earth engine wordt elke pixel binnen deze polygoon dan gebruikt als inputdata. Dit zorgt ervoor dat de trainingsset groter wordt en de variatie binnen een bepaalde klasse beter wordt omvat. Een nadeel is echter dat zo ook foute pixels kunnen worden meegenomen door het onzorgvuldig intekenen van de polygoon.

- Eenmaal je klaar bent met het intekenen van een klasse, kun je deze import configureren. Klink hiervoor op het tandwiel naast de klasse. Geef het een gepaste naam. Daarna verander je de 'Import as' van *geometry* naar type *FeatureCollection*. Voeg daarna een *property* toe met de naam 'class', door te klikken op '*Add property*'. De eerste klasse geef je waarde 1, de 2e 2, .... Zorg er wel voor dat je goed weet welke waarde je aan welke klasse geeft. Neem hierbij eventueel de *class*-nummering over van bovenstaande tabel.

```
P6_Supervised
  Imports (8 entries)
    var ROI: Polygon, 4 vertices
    var Water: FeatureCollection (10 elements)
    var Mangrove: FeatureCollection (10 elements)
    var OtherForest: FeatureCollection (7 elements)
    var Urban: FeatureCollection (7 elements)
    var BareSoil: FeatureCollection (9 elements)
    var Crop: FeatureCollection (7 elements)
    var Testdata_pol: Table users/jasperfeyen/TELEDETECTIE2020/P6_test...
```

```
1 // Project true <--> Geometries enlege classificatie
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Configure geometry import

Name: Mangrove

Import as: FeatureCollection

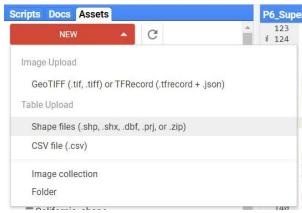
Properties:

Property	Value
class	1

OK Cancel

- Herhaal dit voor elke klasse. Uiteindelijk verschijnt elke klasse als een '*FeatureCollection*' bij de imports-lijst in je script:

```
P6_Supervised
  Imports (7 entries)
    var ROI: Polygon, 4 vertices
    var Water: FeatureCollection (10 elements)
    var Mangrove: FeatureCollection (10 elements)
    var OtherForest: FeatureCollection (7 elements)
    var Urban: FeatureCollection (7 elements)
    var BareSoil: FeatureCollection (9 elements)
    var Crop: FeatureCollection (7 elements)
```



### Voorbeeld trainingsamples voor Mangrove

- Nu de 5 klassen aangemaakt zijn, dienen we ze samen te vatten als een complete trainingsset in earth engine; een gezamenlijke `FeatureCollection`, waarbij de 'class'-*property* wordt overgenomen.

```
//2. Trainingssamples samenvoegen tot 1 ``FeatureCollection``

var classNames =
Mangrove.merge(OtherForest).merge(Water).merge(Crop).merge(Urban).merge(Ba
print(classNames)
```



## De trainingsdata aanmaken

Nu hebben we reeds een `FeatureCollection` met trainingspolygonen, maar deze zeggen nog niks over de trainingspixels. In een volgende stap, extraheren we de trainingspixels per band uit het Sentinel-2 beeld op basis van de aangemaakte polygonen. Dit doen we met de `sampleRegions()` functie. Deze functie extraheert alle pixels binnen opgegeven polygonen, en schrijft elke pixel afzonderlijk naar een nieuw `Feature` binnen een `FeatureCollection`, diewelke ook de class-*property* mee krijgen. Afhankelijk van de grootte van de polygonen, kan deze dataset dus zeer lijvig worden.

```
//Trainingspixels extraheren naar featurecollection = trainingsdata
var traindata = S2_im.sampleRegions({
  collection: classNames, //De trainingspolygonen
  properties: ['class'], //Dit neemt de gewenste eigenschappen van de
  collection over
  scale: 10,
  tileSize:4
});
print('Aantal trainingspixels: ',traindata.size());
//print(traindata) //Niet doen
print(traindata.first()) //Eerste waarde bekijken,
```

### Layer error: Computed value is too Large

Mogelijk heb je bij het gebruik van `sampleRegions()` al onderstaande foutmelding gehad:

### Layer error: Computed value is too large.

Dit krijg je dus als je berekening de maximum-memorycapaciteit overschrijdt. Dit is eenvoudig op te lossen door het toevoegen van een extra parameter `tileScale` aan de `sampleRegions()`-functie, zoals in bovenstaand voorbeeld werd gedaan. Door het verhogen van de `tileScale` zal Earth Engine de opdracht in meerdere stukjes indelen, waardoor de berekening minder gemakkelijk 'out of memory' zal lopen. De bewerking zal hierdoor wel meer tijd in beslag nemen, dus verhoog `tileScale` slechts gelijdelijk.

Zie ook de 'Docs' van `sampleRegions()`

## De classifier trainen

In een volgende stap maken we een classificatiemodel aan en trainen we deze op basis van de traindata. Er bestaan verschillende mogelijke classifiers en 'machine learning'-algoritmen. In Google Earth Engine zitten deze beschikbaar in de `ee.Classifier`-groep. We gebruiken er 3, waarna we kijken welke tot de meest accurate classificatie leidt o.b.v. de validatiedata.

### Minimum Distance Classifier

In een eerste instantie dienen we de classifier te trainen, op basis van de opgestelde trainingsdata. We dienen ook aan te geven welke van de *properties* binnen de trainingssamples de 'class'-bevat, en welke *properties* gebruikt moeten worden om mee te classificeren (de banden).

```
//4. De classifiers trainen en toepassen
// A. Minimum Distance classifier (gebruik van default-waarde
//euclidische afstand)
var MinDist = ee.Classifier.minimumDistance().train({
  features: traindata,
  classProperty: 'class',
  inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
});
```

### CART classifier

```
// B. CART classifier
var Cart = ee.Classifier.smileCart().train({
  features: traindata,
```

```

    classProperty: 'class',
    inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
  );
}

```

## Random Forest classifier

```

// C. Random Forest
var RandomForest = ee.Classifier.smileRandomForest({
  numberofTrees: 60
}).train({
  features: traindata,
  classProperty: 'class',
  inputProperties: bands //verwijzing naar de eerder aangemaakte
bands-lijst
});

```

## Beeld classificeren en visualiseren

Enmaal de classifier(s) opgesteld zijn, kunnen ze worden toegepast op het volledige S2-beeld. Elke pixel wordt dus toegekend tot een klasse, op basis van de kennis opgedaan uit de trainingsdata.

```

// 5. Classifiers toepassen

//MinimumDistance
var classified_MD = S2_im.classify(MinDist)
var classified_CART = S2_im.classify(Cart)
var classified_RF = S2_im.classify(RandomForest)

```

Bij het visualiseren willen we een visueel overzichtelijk resultaat krijgen. Aangezien we eindigen met discrete klassen, stellen we hiervoor een palette op, dat per klasse een kleur aangeeft.

```

var palette = [
  'FF0000', // mangrove (1) // rood
  '7CFC00', // ander bos (2) // lichtgroen
  '1E90FF', //water (3) // blauw
  'FFFFD1', //crop (4) //geel
  '000000', //stad // zwart
  '876829', //BareSoil // bruin
];

var classvis = {min:1, max:6, palette: palette}

Map.addLayer(classified_MD,classvis,'MinimumDistance')
Map.addLayer(classified_CART,classvis,'CART')
Map.addLayer(classified_RF,classvis,'RandomForest')

```

## Accuraatheidsbepaling (Accuracy assessment)

In de *accuracy assessment* toetsen we de bruikbaarheid van het classificatiemodel: welke foutenmarge is er aanwezig? Dit doen we op basis van een foutenmatrix (**error matrix**). Om deze op te stellen hebben we nood aan een set van (onafhankelijke) testdata. We willen met andere woorden weten hoe goed onbekende pixels worden geklassificeerd door de classifier.

### Training- Validation en Testdata

Een veel gebruikte methode bij het opstellen van modellen, is het opsplitsen van de *trainingset*-dataset in training- en validatiedata. Hierbij wordt de *trainingdata* at random gesplits in meestal een 80/20-verhouding. Daarnaast wordt er vaak nog gebruik gemaakt van een derde, volledig afzonderlijke dataset: de **testdataset**:



**Traindata**= de data gebruikt om het model te trainen en dus te *fitten*. Het model bekijkt en leert van deze data.

**Validatiedata**= Het deel van de data dat gebruikt zal worden om na te gaan hoe goed het model werkt op onbekende data. Dit deel zal dus niet gebruikt worden om het model te trainen. Hierdoor kunnen verschillende classificatiemodellen en parameters binnen het model tegenover elkaar worden afgewogen en het model zo worden geperfected. Dit wordt ook wel *parameter tuning* genoemd. Validatiedata wordt dus gebruikt tijdens de ontwikkeling en het zoeken van het beste model. Bij spatiale data echter, dient hier voorzichtig mee te worden omgegaan door het fenomeen van *spatiale autocorrelatie*. Hierbij zijn de validatiepixels veleel buurpixels van de trainpixels. Dit is het geval wanneer er bijvoorbeeld gebruik wordt van polygonen als inputdata.

**Testdata** = Deze afzonderlijke dataset wordt gebruikt om bij een finaal model accuraatheidsmaten van de bekomen classificatie te berekenen. Testdatasets worden meestal ook zeer goed verzorgd en zijn goed verzamelde (veld)datapunten. De *spatiale autocorrelatie* vervalt hier.

### De Error Matrix: interpretatie

Zie ook de foutenmatrix handboek pagina 577

De Error Matrix wordt opgesteld door het vergelijken van de geklassificeerde testdata-waarden en de referentiedata. Onderstaande matrix geeft een voorbeeld van dergelijke matrix:

Classification	Ground reference data						User's accuracy
	Bareland/Constr	Dead Mangrove	Mangrove	Other Forest	Other Vegetation	Water	
Bareland/Constr	40	3	0	0	4	3	50 80.00%
Dead Mangrove	0	44	1	0	1	4	50 88.00%
Mangrove	0	0	125	1	0	1	127 98.43%
Other Forest	1	1	8	97	2	0	109 88.99%
Other Vegetation	2	2	7	12	87	1	111 78.38%
Water	0	2	2	0	1	154	159 96.86%
Total	43	52	143	110	95	163	606 81.37%
Producer's accuracy	93.02%	84.62%	87.41%	88.18%	91.58%	94.48%	
Overall accuracy	90.26%						
Kappa index							

- **Rijen:** resultaat van de classificatie
- **Kolommen:** validatie data
- **Diagonaal:** pixels die goed geclasseerd zijn (validatie data = classificatie)
- **Niet-diagonaal:**
  - **Omissie:** de niet diagonale kolom-elementen. Deze pixels behoren tot een klasse, maar werden ingedeeld in een verkeerde klasse. In het voorbeeld: 18 pixels moesten Mangrove zijn, maar werden ingedeeld onder andere klassen: 1 als 'Dead Mangrove', 8 als Other Forest, 7 als Other Vegetation en 2 als \*Water.
  - **Commissie:** Deze pixels geven aan welke pixels verkeerd werden ingedeeld in deze klasse. Voor Water zijn dit er bijvoorbeeld 2 (moesten 'Dead Mangrove') + 2 (moest Mangrove zijn + 1 (moest Other Vegetation zijn)) = 5 pixels verkeerd als 'Water' ingedeeld.

Op basis van de error matrix kunnen er enkele **accuraatheidsmaten** worden berekend.

- **Overall accuracy:** Dit is de som van de diagonale elementen, gedeeld door het totaal aantal pixels (= "juist ingedeeld"/totaal).
- **Producer accuracy:** het aantal correct ingedeeld pixels in elke klasse, gedeeld door het aantal validatiepixels van die klassen. Het geeft een indicatie hoe goed de validatiepixels geclasseerd werden (Bijvoorbeeld Mangrove =  $(125/143 = 87,41\%)$  werd goed geclasseerd). Deze maat geeft de probabilitet weer dat een pixel die in een bepaalde klasse werd gestopt in werkelijkheid ook tot die klasse behoort.
- **User/consumer accuracy:** Het aantal correct geclasseerde pixels in elke klasse (diagonalelementen), gedeeld door het aantal pixels ingedeeld in die klasse (rijtotaal).
- **Kappa (KHAT) index:** de kappa index is een gecorrigeerde accuraatheidsmaat, die intra- en interobserver agreement in rekening houdt. Het houdt m.a.w. rekening met pixels die per toeval juist geclasseerd zijn. Onderstaande tabel geeft een interpretatie weer van de kappa-waarde.

Values of kappa	Interpretation
< 0	No agreement
0-0.19	Poor agreement
0.20-0.39	Fair agreement
0.40-0.59	Moderate agreement
0.60-0.79	Substantial agreement
0.80-1.00	Almost perfect agreement

## De Error matrix in Earth Engine

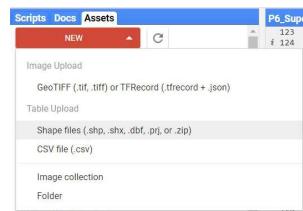
### INLEZEN VAN DE TESTDATASET

In voorliggend voorbeeld maken we gebruik van een extra testdataset voor het opstellen van de error matrix. Gezien we geen optimalisatie van parameters gaan doorvoeren, maken we geen gebruik van validatedata en wordt de volledige trainingscollectie ook trainingsdata.

De gebruikte testdataset bestaat uit kleine polygonen met een diameter van 25m en representeren GPS-punten genomen op veldbezoek. De 25m-buffer rond de GPS-punten werd genomen om voldoende testpixels te weerhouden voor de accuracy assessment.

Je kunt de shape-file hier downloaden: [P6testdata\\_poly.zip](#)

Om deze toe te voegen aan Earth Engine, laad je deze op via de 'Asset'-tab. Daarna kun je het importeren als een `FeatureCollection` in je script. Noem dit 'Testdata\_pol'. Bekijk ook even deze polygonen. Onder welke `property` zitten de klassen hier opgeslagen?



Vervolgens extraheren we de pixelwaarden op basis van deze testpolygons, net zoals we dit gedaan hebben bij de trainingspixels:

```
//Na inlezen van validatedata: maak een testdatacollectie, zoals bij
//het opmaken van traindata
print('Testpolygons',Testdata_pol) //bekijk de properties

//Testpixels extraheren naar featurecollection
var testdata = S2_im.sampleRegions({
  collection: Testdata_pol, //De trainingspolygons
  properties: ['val'], //Dit neemt de gewenste eigenschappen van de
  collection over
  scale: 10
});
print('Aantal testpixels: ',traindata.size());
```

Nu we de testpixelwaarden geëxtraheerd hebben, kunnen we deze vergelijken met de geclasseerde pixels in een **error matrix**. Tevens staat Earth Engine een rechtstreekse berekening van de verschillende accuraatheidsmaten toe. In onderstaand stukje code staat het voorbeeld voor de Minimum Distance classifier. Pas dit toe voor alle classifiers. Welke classifier heeft de grootste algemene accuraatheid (*Overall accuracy*)?

```
// Validatie met de testdata
var val_MinDist = testdata.classify(MinDist);
var ErrorMatrix_MinDist = val_MinDist.errorMatrix('val',
'classification')

print('MinDist Validation error matrix: ',
ErrorMatrix_MinDist.array().transpose());
print('MinDist Validation overall accuracy: ',
ErrorMatrix_MinDist.accuracy());
print('MinDist Producer Accuracy: ',
ErrorMatrix_MinDist.producersAccuracy());
print('MinDist User/Consumer Accuracy: ',
ErrorMatrix_MinDist.consumersAccuracy());
print('Kappa index: ', ErrorMatrix_MinDist.kappa());

//Omvatten naar een Feature + transpose
var ErrorMatrix_MinDist = ee.Feature(null, {matrix:
ErrorMatrix_MinDist.array().transpose()});
```

## De Error matrix in Earth Engine

In Earth Engine wordt de error matrix opgeroepen met de `ee.ConfusionMatrix()`-functie. Resulterend is een lijst met 7 elementen (de rijen), waarbij elke rij op zijn beurt bestaat uit 7 elementen (de kolommen). In Earth Engine corresponderen de rijen met de referentiedata en de kolommen met de geclasseerde data. Met de `.array()`-functie kunnen we deze matrix transponeren, zodat deze overeenkomst met de matrix in het voorbeeld (kolommen: referentie, rijen: geclasseerd), wat de standaard weergave is.

```
errorMatrix(actual, predicted, order)
Computes a 2D error matrix for a collection by comparing two columns of a collection: one containing the actual values, and one containing predicted values. The values are expected to be small contiguous integers, starting from 0.
Axis 0 (the rows) of the matrix correspond to the actual values, and Axis 1 (the columns) to the predicted values.
```

Daarnaast wordt ook klasse '0' meegerekend in de berekening. Gezien deze in ons voorbeeld niet bestaat, zal de 1e rij/kolom enkel 0-waarden bevatten.

## Voorbeeld: interpretatie error matrix

In ons voorbeeldje werd onderstaande ErrorMatrix verkregen voor de MinDist classifier. Door de `.transpose()`-functie komt de referentiedata terecht in de kolommen, terwijl de classificatiedata zich in de rijen bevindt.

```
MinDist Validation error matrix: JSON
▼ List (7 elements) JSON
  ▶ 0: [0,0,0,0,0,0,0]
  ▶ 1: [0,179,71,0,0,0,0]
  ▶ 2: [0,59,257,0,15,22,0]
  ▶ 3: [0,3,0,438,0,0,0]
  ▶ 4: [0,0,12,0,147,75,41]
  ▶ 5: [0,0,0,0,0,193,18]
  ▶ 6: [0,0,0,0,0,171,19]
```

Na wat opschoning in excel, ziet de error matrix van dit voorbeeld er uit als volgt:

Classified data	Reference data							Consumer accuracy
	Mangrove	OtherForest	Water	Crop	Urban	Soil	Total	
Mangrove	179	71	0	15	22	0	250	71.60%
OtherForest	59	257	0	0	0	0	353	72.80%
Water	3	0	438	0	0	0	441	99.32%
Crop	0	12	0	147	75	41	275	53.45%
Urban	0	0	0	0	193	18	211	91.47%
Soil	0	0	0	0	171	19	190	10.00%
	241	340	438	162	461	78	1720	
Producer's accuracy	74.27%	75.59%	100.00%	90.74%	41.87%	24.36%		
Overall accuracy	71.69%							
Kappa index	65.50%							

Welke klassen scoren goed? Welke zijn minder accuraat? Aan de *overall accuracy* en de *Kappa index* kan worden afgeleid dat de classificatie reeds een goede indicatie heeft, maar nog voor verbetering vatbaar is. Hoofdzakelijk de klassen 'water' en 'crop' scoren goed, terwijl 'Soil' moeilijker te onderscheiden valt van 'crop' en 'urban'. Betere trainingsdata is hier dus de boodschap!

## Extra: Exporteren van de Error Matrix

In Google Earth Engine is de weergave van de error matrix niet zo handig. Om verdere accuraatheidsmaten uit te rekenen en een betere interpretatie te kunnen uitvoeren kan het handig zijn om de error matrix te exporteren als een .csv-bestand, dewelke in andere software (zoals excel) geopend kan worden.

Met de `Export.table.toDrive()`-functie kunnen we de matrix exporteren naar onze Google Drive. Hiervoor dienen we dit eerst om te zetten naar een *feature*.

```
//Omzetten naar een Feature
var ErrorMatrix_MinDist = ee.Feature(null, {matrix: ErrorMatrix_MinDist.array()});

//Exporteren van de errormatrix
Export.table.toDrive({
  collection: ee.FeatureCollection(ErrorMatrix_MinDist),
  description: 'F6_Errormatrix',
  fileFormat: 'CSV',
  folder: 'TELEDETECTIE_2021'
});
```

## Volledig script

Via deze link: <https://code.earthengine.google.com/665cbaa3a887d685d4d07c6081902f19>

## 5.5 Verbeteren van de classificatie

---

In de gesuperviseerde classificaties testten we reeds 3 'classifiers', waarvan de *Random Forest* classifier leidde tot een iets betere algemene accuraatheid. Desondanks ze al tot acceptabele percentages leidden, bestaan er enkele opties om de classificatie nog te verbeteren.

### Aanpassen van het aantal trainingssamples en aanpassen *sampling*-strategie

---

De opgemaakte trainingssamples kunnen worden verbeterd door het aantal samples omhoog te trekken, of door de *sampling*-strategie aan te passen. In dit voorbeeld tekenden we zelf polygonen in op basis van visuele inspectie, waardoor we enkel de pixels werden aangeduid waar we relatief zeker waren van de desbetreffende klasse.

Een betere techniek zou erin bestaan om op voorhand willekeurige locaties door de computer aan te laten duiden, eventueel gestratificeerd per landbedekkingsklasse ('*stratified sampling*').

### Toevoegen van Indices

---

Ook door de input-banden aan te passen of extra banden toe te voegen, kan er voor zorgen dat de onderscheiding van de klassen wordt verbeterd. Zo kan:

- er hoogtedata (zgn. *Digital Elevation Model* of DEM) worden toegevoegd, waaruit bv de hellingsgraad van het terrein kan worden berekend. Zo kunnen land*features* met een specifieke topografie zoals basins, kanalen, toppen, dalen, hellingen gemakkelijker worden geïdentificeerd.
- er informatie uit berekende indices helpen voor versterking van verschillen tussen klassen, zoals NDVI, MNDWI of in de context van mangrove-classificatie de MVI.

### Opdracht

---

Tracht je classificatie te verbeteren door:

- enkele indices toe aan het Sentinel-2 beeld van vorige oefening, zoals de MVI, NDVI en NDWI.
- Textuurmatrices toe te voegen. Zie hiervoor [P5 - Textuur](#) Gebruik deze keer enkel de Random Forest classifier. Bekijk de bekomen accuraatheid. Merk je verbeteringen op?

## 5.6 Oppervlaktebepaling

### Oppervlaktebepaling

Nu de classificatie is uitgevoerd, kunnen we per finale landbedekkingsklasse ook de oppervlakte bepalen. Gezien het bekomen classificatieleresultaat een rasterbeeld met pixels is, zal de oppervlakte een klasse gelijk zijn aan de som van de oppervlakte van elke pixel binnen de klasse.

- `.area()` : wordt gebruikt om de oppervlakte van `Features` te bepalen.
- `ee.Image.pixelArea()` : wordt gebruikt om de oppervlakte van pixels binnen een rasterbeeld te bepalen. Het resultaat wordt weergegeven in  $m^2$ .

In Google Earth Engine zijn hier verschillende methodes voor de oppervlakteberekening.

- **Stap 1** - afzonderen van de klasse met een `.eq()`-functie. Het resultaat is een binair beeld, waarbij de '1'waarde de geselecteerde klasse(n) voorsteld, waarde '0' de overige pixels.

```
// Mangroveklasse afzonderen uit het geclasseerd beeld
var Mangrove_class = classified_RF.eq(1)

// Visualiseren van het beeld
Map.addLayer(Mangrove_class, {min:0, max:1, palette: ['white',
'green']}, 'Mangrove Cover')
```

- **Stap 2** - pixelwaarde vervangen door de oppervlakte van de pixel.

```
//Pixelwaarden (=1) vermenigvuldigen met de pixeloppervlakte.
var areaImage = Mangrove_class.multiply(ee.Image.pixelArea())
```

- **Stap 3** - Nu elke pixel van de Mangrove-klasse een waarde heeft gelijk aan de oppervlakte, kunnen we de som nemen van alle pixels in de regio om de totale oppervlakte te verkrijgen. Dit doen we met een Reducer (zie ook P4)

```
var area_Mangrove = areaImage.reduceRegion({
reducer: ee.Reducer.sum(),
geometry: ROI,
scale: 10, //minimumpixelgrootte zorgt voor meest accurate waarde.
maxPixels: 1e10 //Vergroten rekencapaciteit
})
```

- **Stap 4** - Het resultaat is een dictionary, waarbij de enige 'waarde' de oppervlakte is (check dit door het resultaat naar de Console te prullen). Print nu de oppervlakte naar de console. Deze is momenteel uitgedrukt in  $m^2$

```
var MangroveAreaHa =
ee.Number(area.get('classification')).divide(1e4).round()
print('Oppervlakte Mangrove: ', MangroveAreaHa)
```

### Opdracht

Bereken en print ook de oppervlakte van de klasse 'Ander bos' binnen het studiegebied.

### EXTRA: Oppervlaktehistogram

Hierboven werd een standaardmethode behandeld om de oppervlakte per klasse in een geclasseerd beeld afzonderlijk te berekenen. Met onderstaande code kun je meteen de oppervlakte per klasse gezamenlijk berekenen en via de `ui.Chart.image.byClass()` functie een histogram visualiseren die de oppervlaktes weergeeft.

```
var areaImageHa =
ee.Image.pixelArea().divide(1e4).addBands(classified_RF);

// Berekenen oppervlakte per klasse met een reduceRegion()-functie
// We voeren m.a.w. een reductie uit, waarbij ons
// classificatieleresultaat als regio's gelden, waarbinnen de soms wordt
// berekend. de .group()-functie maakt dit mogelijk
var areas = areaImageHa.reduceRegion({
    reducer: ee.Reducer.sum().group({
        groupField: 1,
        groupName: 'classification', //De bandnaam van het
        //geclasseerd beeld is standaard 'classification'
    }),
    geometry: ROI,
    scale: 10,
    tileSize: 6,
    maxPixels: 1e10
});

var classAreas = ee.List(areas.get('groups'))
print(classAreas)

var areaChart = ui.Chart.image.byClass({
    image: areaImageHa,
    classBand: 'classification',
    region: ROI,
    scale: 20, //Op 20m pixelgrootte berekenen voor beperken
    rekencapaciteit
    reducer: ee.Reducer.sum(),
    classLabels: ['', 'Mangrove', 'Other Forest', 'Water',
    'Crop', 'Urban', 'Bare Soil'], // ''0' is een lege klasse
}).setOptions({
    hAxis: {title: 'Classes'},
    vAxis: {title: 'Area Km2'},
    title: 'Area by class',
    series: [
        0: {color: 'red'},
        1: {color: 'green'},
        2: {color: 'blue'},
        3: {color: 'yellow'},
        4: {color: 'black'},
        5: {color: 'brown'}
    ]
});
print(areaChart);
```

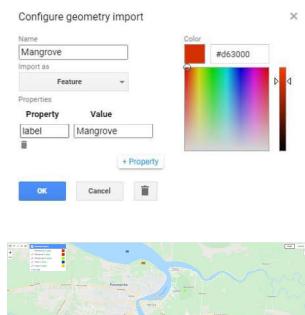
## 5.7 Spectrale responsiecurves

Als laatste onderdeel van dit practicum, kijken we even naar de spectrale signaturen van onze klassen. Uit deze curves kunnen we volgende zaken afleiden (zie ook Practicum 1):

- Welke banden/indices zorgen voor onderscheid tussen klassen?
- Welk spectrale curves heeft elk van onze klassen? Kunnen we deze curves ook verklaren?

### Aanmaken voorbeeldFeatures

Om spectrale curves te maken, hebben we eveneens voorbeeld-samples nodig van elk van de klassen. Dit kan door opnieuw nieuwe *features* in te tekenen (bv één punt voor 1 klasse). We doen dit bijvoorbeeld voor de klassen Mangrove, OtherForest, Water en Urban. Zoek één representatief punt voor elk van de klassen. Maak deze aan als het type *Feature* en geef het een *property* 'label', met een bijpassende naam



Eenmaal de VoorbeeldFeatures zijn aangemaakt kun je ze samenvoegen in het script:

```
var vbPol = ee.FeatureCollection([Mangrove,OtherForest,Water,Urban]);
```

### S2-beeld aanmaken

Maak ook opnieuw het Sentinel-2 beeld uit vorige oefening aan, met de indices:

```
//
-----
// STAP 1 - Inladen en klaarzetten van S2-beeld. Met extra cloud-
masking
// -----
//Cloudprobability functie:
// Functie die nieuwe CloudProbability collectie samenvoegt met S2
// (sen2cloudless)
// meer info: https://medium.com/sentinel-hub/cloud-masks-at-your-
// service-6e5b2cb2ce8a
var getS2_SR_CLOUD_PROBABILITY = function () {
    var innerJoined = ee.Join.inner().apply({
        primary: ee.ImageCollection("COPERNICUS/S2_SR"),
        secondary: ee.ImageCollection("COPERNICUS/
S2_CLOUD_PROBABILITY"),
        condition: ee.Filter.equals({
            leftField: 'system:index',
            rightField: 'system:index'
        })
    });
    var mergeImageBands = function (joinResult) {
        return ee.Image(joinResult.get('primary'))
            .addBands(joinResult.get('secondary'));
    };
    var newCollection = innerJoined.map(mergeImageBands);
    return ee.ImageCollection(newCollection);
};

// Mask out clouds
```

```
var maskClouds = function(image) {
    var cloudProbabilityThreshold = 40;
    var cloudMask =
    image.select('probability').lt(cloudProbabilityThreshold);
    return image.updateMask(cloudMask);
};

//Aanmaken van een ImageCollection ter hoogte van Mangroves
//Paramaribo, Suriname
var S2_coll = getS2_SR_CLOUD_PROBABILITY()
.filterDate('2019-08-01','2019-10-30')// Filteren voor het
jaar 2020, droge tijd
.filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',50) ////
Voorselectie obv wolken
.map(maskClouds) //toepassen van de cloudmaskfunctie
.filterBounds(ROI); //collectie filteren obv de
Kustzonegeometrie

// -----
// TOEVOEGEN INDICES
// -----
var addIndices = function (image) {
    var mvi = image.expression('(B8-B3)/(B11-B3)', {
        'B8' : image.select('B8'),
        'B3' : image.select('B3'),
        'B11' : image.select('B11')
    }).float().rename('MVI');
    var ndvi = image.normalizedDifference(['B8',
'B4']).rename('NDVI');
    var ndwi = image.normalizedDifference(['B3',
'B12']).rename('NDWI');
    return image.addBands(mvi).addBands(ndvi).addBands(ndwi);
};

//Toepassen indices + medianreducer + clippen
var S2_im = S2_coll.map(addIndices).median().clip(Paramaribo);
```

### Cloud mask methode

In voorgaande code wordt opnieuw gebruik gemaakt van de extra S2-cloudmask methode. Je kunt evengoed gebruik maken van de andere strategiën, zoals gezien in "Cloud Masking" van Practicum 4.

### Spectrale responsiecurve aanmaken

Vervolgens kunnen we de Chart aanmaken. Tevens linken we de overeenkomstige golflengtes aan de banden, om zo een spectrum te krijgen met de golflengte in de X-as.

```
// Banden S2 aangeven (hier: volledig, inclusief B1 en B9 (worden
weggelaten in classificatie wegens onbruikbaar)
var wavelengths
=[ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A','B9','B11','B12']

//De overeenkomstige golflengte per band aangeven (zie bandenverdeling
//Sentinel-2).
var wavelengths =[443.9,496.6,559.664.5,703.9,740.2,782.5,835.1,864.8,
945,1613.7,2202.4]

//Aanmaken Chart
var Chart = ui.Chart.image.regions({
    image: S2_im.select(bands),
    regions: vbPol,
```

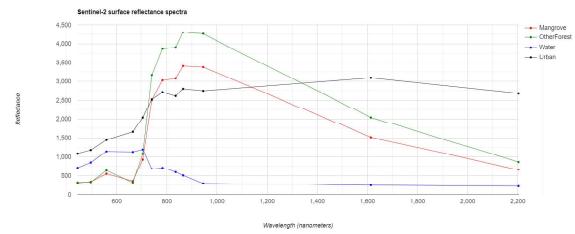
```

reducer: ee.Reducer.mean(),
scale: 10,
seriesProperty: 'label',
xLabels: wavelengths
})

var plotOptions = {
  title: 'Sentinel-2 surface reflectance spectra',
  hAxis: {title: 'Wavelength (nanometers)'},
  vAxis: {title: 'Reflectance'},
  lineWidth: 1,
  pointSize: 4,
  series: [
    0: {color: 'red'}, // Mangrove
    1: {color: 'green'}, // Forest
    2: {color: 'blue'}, // water
    3: {color: 'black'}, //crops/grass
  ],
};

```

```
print(Chart.setOptions(plotOptions));
```



## 5.8 Oefeningen

### OEF 5.1 - Classificatie van België

#### Gegeven

In volgende oefening maken we een landclassificatie van België op basis van een Landsat-8 beeldcomposit. Om de grenzen van België te bekomen, maken we gebruik van volgende dataset: [https://developers.google.com/earth-engine/datasets/catalog/USDOS\\_LSIB\\_SIMPLE\\_2017](https://developers.google.com/earth-engine/datasets/catalog/USDOS_LSIB_SIMPLE_2017)

Om hieruit België te filteren, maak je gebruik van onderstaande code:

```
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
var bel = countries.filterMetadata('country_na', 'equals', 'Belgium');
```

#### Opdracht

- Maak één wolkenvrij Landsat-8 beeld aan voor 2019. Weerhoud enkel de bruikbare banden voor classificatie: voor Landsat-8 dit zijn volgende banden: blauw, groen, rood, nir, swirl1, swirl2.
- Clip je resulterende beelden op basis van de Belgische grens.
- Voer een classificatie uit met volgende klassen: *bos, water, grasland, landbouw, urban*. Maak hiervoor je eigen trainingssamples aan. Vergeet ze niet samen te voegen tot één `FeatureCollection`.
- Kies 2 classifiers naar keuze.
- Visualiseer je resultaat naar eigen keuze. Evaluere het visueel: is je classificatie geslaagd? Welke classifier lijkt beter?
- Bereken de *Overall Accuracy* van je classificatie. Maak gebruik van deze validatieset: [P6\\_Oef1\\_validation.zip](#). Hierbij maak je gebruik van de eigenschap 'val', waarbij volgend schema geldt:

val	Landbekkingsklasse
1	Bos
2	Water
3	Grasland
4	Landbouw
5	Urbaan

#### Oplossing Oef 5.1

Voor deze oefening kun je zowel Landsat Collectie 2 (de allernieuwste) of Landsat Collectie 1 (toekomstig niet meer gebruikt, maar momenteel nog in een overgangsfase) gebruiken. Het verschil is verwaarloosbaar.

Oplossing Landsat 8 Collectie 1: <https://code.earthengine.google.com/b80d9ab8e0be2e309dbd64fe3f7a0f81> Oplossing Landsat 8 Collectie 2: <https://code.earthengine.google.com/021d9c2592813d2779aff69722edf08a>

Opgelet: het spreekt voor zich dat de gebruikte trainingsvectoren in dit voorbeeld zeer rudimentair zijn, met een ruwe classificatie tot gevolg. Daarbij werden de testpunten ook gehaald uit een bestaande landcover classificatie van België, maar met een resolutie van 100m, waardoor de kwaliteit van de testdata niet perfect is. Verder is een verbetering van het classificatieresultaat ook mogelijk door het gebruik maken van Multitemporale beelden: hiermee wordt het sezonaal karakter van de o.a. de landbouwvelden mee in rekening gebracht, waardoor een beter onderscheid tussen o.a. grasland en landbouw (sterker temporeel karakter) behaald kan worden.

### OEF 5.2 - Eyjafjallajökull



De gletsjer Eyjafjallajökull is een van de kleinere gletsjers op IJsland en heeft een oppervlakte van ongeveer 100 km<sup>2</sup>. De Eyjafjallajökull ligt ten noorden van het plaatsje Skógar. Op de oostflank van de vulkaan, nabij de bergpas Fimmvörðuháls, vond op 20 maart 2010 nieuwe vulkanische activiteit plaats. Een tweede explosievere uitbarsting in de hoofdkrater van de Eyjafjallajökull, begon op 14 april 2010. In grote delen van Europa werd het vliegverkeer dagenlang volledig stilgelegd vanwege de aswolken die de vliegtuigen kunnen beschadigen.

#### Gegeven

Maak gebruik van volgend Sentinel-2 beeld en ROI, genomen in 2019:

```
var S2 = ee.Image('COPERNICUS/S2_SR/20190810T125311_20190810T125306_T27VWL');
var ROI = ee.Geometry.Polygon([
  [[[-19.967455239503007, 63.845568279400595],
    [-19.967455239503007, 63.398959439658746],
    [-18.824877114503007, 63.398959439658746],
    [-18.824877114503007, 63.845568279400595]]], null, false);
```

```
var S2 = ee.Image('COPERNICUS/S2_SR/
20190810T125311_20190810T125306_T27VWL').clip(ROI)
```

- In deze oefening gaan we geen cloud mask toevoegen, maar de wolken en wolkenschaduwen opnemen in de classificatie.
- Training data: Chinese experten digitaliseerden verschillende polygonen in het gebied rond de vulkaan. Hierbij werd onderscheid gemaakt in 5 klassen:
  - Gletsjer
  - Schaduw
  - Bodem
  - Vegetatie
  - Water
  - Wolken

De Trainingfiles werden reeds ondergebracht in een `FeatureCollection` en kunnen via deze link worden ingelezen:

```
var traindata = ee.FeatureCollection("users/jasperfeyen/
TELEDETECTIE2020/P6_oef2_training");
```

- **Referentie data:** tijdens een veldcampagne in 2019 werd het gebied rond de vulkaan intensief bemonsterd. Honderden pixels werden op het terrein bezocht en de landbedekking werd geregistreerd.

Je kunt de shape-file hier downloaden: [P6\\_oef2\\_val.zip](#)

### Gevraagd

Classificeer het 2019 beeld met behulp van 2 supervised classifiers naar keuze. Voor de classificatie transformeer je de data, zodat je slechts 3 getransformeerde banden overhoudt die de meeste informatie bevatten.

### Oplossing

Via volgend script: <https://code.earthengine.google.com/38fd0ac0f35c8a4175afb7273d8eae4b>