Cvičení v prostředí Google Earth Engine

Klasifikace

Dálkový průzkum Země

OBSAH

1 TEORETICKÁ ČÁST 3

1.1 ANALÝZA HLAVNÍCH KOMPONENT 3

1.2 KLASIFIKACE 3

Neřízená klasifikace 3

Řízená klasifikace 3

2 DATA 5

2.1 LANDSAT-9 5

3 PRAKTICKÁ ČÁST 6

3.1 NEŘÍZENÁ KLASIFIKACE 6

1. Import potřebných dat 6
2. Příprava dat 7
3. Vizualizace snímku 7
4. Analýza hlavních komponent 8
5. K-Means algoritmus 11
6. X-means algoritmus 14

3.2 ŘÍZENÁ KLASIFIKACE 16

1. Příprava dat 16
2. Tvorba trénovací množiny 19
3. Trénink klasifikátoru 23
4. Klasifikace snímku 23
5. Odhad přesnosti klasifikace 24

3.3 POROVNÁNÍ VÝSLEDKŮ 25

1 TEORETICKÁ ČÁST

1.1 Analýza hlavních komponent

Analýza hlavních komponent (Principal Components Analysis) je statistická metoda, která se využívá ke snížení dimenze dat. Původní pásma přepočítá na soustavu nových pásem, která jsou informačně bohatá a lépe vysvětlují variabilitu dat. Základní vlastností každé komponenty je její míra variability. Hlavní komponenty jsou uspořádány podle rozptylu od největšího k nejmenšímu. Platí, že první komponenta obsahuje, co nejvíce informací a poslední komponenta naopak obsahuje informací, co nejméně. Výsledky analýzy hlavních komponent mohou posloužit ke zlepšení výsledků následné klasifikace.

1.2 Klasifikace

Klasifikace dává našemu snímku význam, protože poskytuje informace o tom, co se na snímku nachází. Využívá tzv. klasifikátory, které na základě pravidel zařazují jednotlivé pixely obrazu do tříd.

Neřízená klasifikace

Neřízená klasifikace využívá shlukovou analýzu, kdy jsou jednotlivé pixely zařazeny do spektrálních tříd na základě podobných vlastností. Spoléháme na to, že se jednotlivé pixely přirozeně přiřadí do odpovídajících tříd. Uživatel pak jednotlivým spektrálním třídám přiřadí význam a provede reklasifikaci. Výsledkem jsou informační třídy.

Algoritmy pro neřízenou klasifikaci:

- K-means algoritmus
- X-means algoritmus
- ISODATA algoritmus
- LVQ algoritmus

Řízená klasifikace

Při řízené klasifikaci specifikujeme tzv. trénovací množinu. Trénovací množina obsahuje trénovací plochy, které jsou příkladem cílových tříd. Následně vybereme vhodný klasifikátor, který natrénujeme na trénovacích datech a poté natrénovaný klasifikátor

použijeme pro klasifikaci družicového snímku. Průběh řízené klasifikace tedy lze rozdělit do dvou fází, a to na tréninkovou a klasifikační část. Výsledkem celého procesu jsou opět informační třídy.

Klasifikátory pro řízenou klasifikaci:

- CART klasifikátor
- Random Forest klasifikátor
- Klasifikátor rozhodovacích stromů
- Klasifikátor nejbližšího souseda

2 DATA

V rámci dnešního cvičení budeme pracovat s daty z družice Landsat-9.

2.1 Landsat-9

Landsat-9 je mise NASA, která byla vypuštěna na oběžnou dráhu v září 2021.

Družice nese dva senzory, a to OLI-2 a TIRS-2. OLI-2 (Operational Land Imager) poskytuje panchromatická a multispektrální data. TIRS-2 (Thermal InfraRed Sensor 2) získává tepelná data. Družice Landsat-9 poskytuje data v 11 spektrálních pásmech.

Prostorové rozlišení snímků Landsat-9 je 15 m pro panchromatické snímky, 30 m pro multispektrální snímky a 100 m pro termální snímky.

Časové rozlišení Landsat-9 je přibližně 16 dní.

Band Number	Description	Wavelength (µm)	Spatial Resolution (m)	Radiometric Resolution (bits)
Band 1	Coastal / Aerosol	0.43 - 0.45	30	14
Band 2	Visible blue	0.45 - 0.51	30	14
Band 3	Visible green	0.53 - 0.59	30	14
Band 4	Visible red	0.63 - 0.67	30	14
Band 5	Near-infrared	0.85 - 0.87	30	14
Band 6	Short infrared	1.56 - 1.65	30	14
Band 7	Short infrared	2.10 - 2.29	30	14
Band 8	Panchromatic	0.50 - 0.67	15	14
Band 9	Cirrus	1.36 - 1.38	30	14
Band 10	Thermal infrared	10.60 - 11.19	100	14
Band 11	Thermal infrared	11.50 - 12.51	100	14

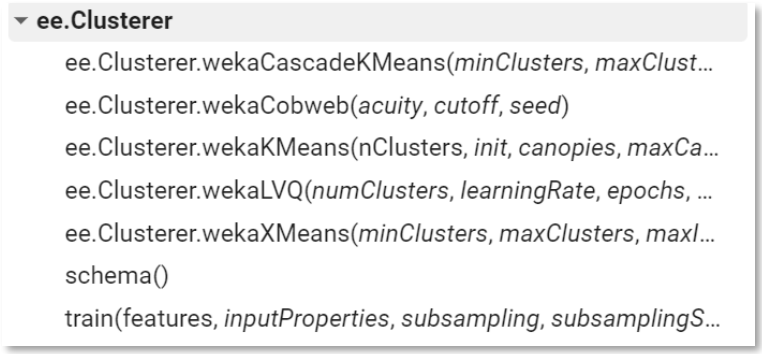
3 PRAKTICKÁ ČÁST

V praktické části cvičení projdeme téma klasifikací. Ukázkové skripty z dnešního cvičení jsou dostupné prostřednictvím sdíleného repositáře na adrese: https://code.earthengine.google.com/?accept_repo=users/michaelasvehlikova/cviceni2. Repositář si ideálně importujte do svého prostředí a kopírujte potřebné bloky kódu z jednotlivých skriptů.

3.1 Neřízená klasifikace

Neřízená klasifikace je z hlediska postupu méně náročná než řízená. V rámci neřízené klasifikace dochází k rozdělené pixelů do shluků na základě podobných spektrálních charakteristik.

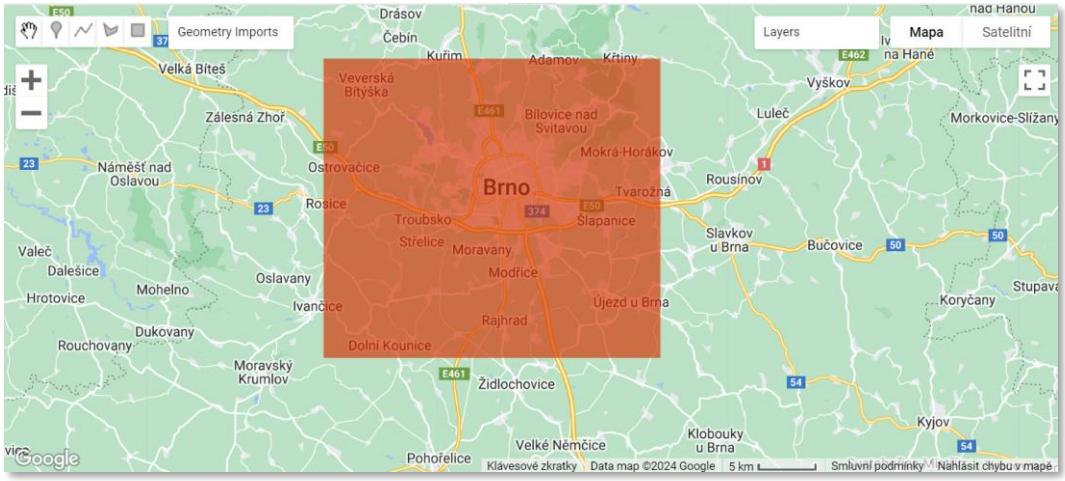
Google Earth Engine nabízí poměrně širokou škálu algoritmů pro neřízenou klasifikaci. Tyto algoritmy jsou součástí balíčků **ee.Clusterer** a lze si jejich výčet a bližší specifikaci zobrazit v záložce *Docs*, která obsahuje dokumentaci API. Jedním z nejznámějších algoritmů je jistě K-means, který si dnes i vyzkoušíme. K-means nebude však jediný, společně se podíváme i na algoritmus X-means.



1. Import potřebných dat

Na začátku si do našeho pracovního prostředí nahrajeme potřebná data. Pro účely dnešního cvičení využijeme datovou sadu Landsat-9, která je v datovém katalogu Google Earth Engine dostupná pod kódem LANDSAT/LC09/C02/T1_L2.

Následně si v mapovém okně pomocí geometrických nástrojů zakreslíme zájmovou oblast. Její volba je zcela libovolná.



Naimportovanou datovou sadu Landsat-9 a zakreslenou zájmovou oblast si můžeme přejmenovat dle svého uvážení. V rámci ukázkového skriptu jsou snímky Landsat-9 označeny jako **landsat** a zájmová oblast jako **oblast**.

```
Imports (2 entries)
  var oblast: Polygon, 4 vertices
  var landsat: ImageCollection LANDSAT/LC09/C02/T1_L2
```

2. Příprava dat

Naše data si ještě připravíme na další práci. Je nutné vyfiltrovat celou kolekci snímků Landsat-9 na rozsah zájmové oblasti ve vybraném časovém období pomocí metod **filterBounds()** a **filterDate()**. Zároveň z naší vyfiltrované kolekce vybereme snímek s nejmenší oblačností, a proto kolekci vzestupně seřadíme díky funkci **sort()** podle hodnot atributu **CLOUD_COVER**, který nese informaci o oblačnosti. Následně s využitím metody **first()** vybereme první snímek ze seřazené kolekce. Náš vybraný snímek ořežeme na rozsah zájmové oblasti metodou **clip()**.

```
1. // Filtrace snímku zájmové oblasti ve vybraném období s nejmenší oblačností
2. var snimekLandsat = landsat
3.     .filterBounds(oblast)
4.     .filterDate("2023-05-01","2023-9-30")
5.     .sort("CLOUD_COVER")
6.     .first()
7.     .clip(oblast);
```

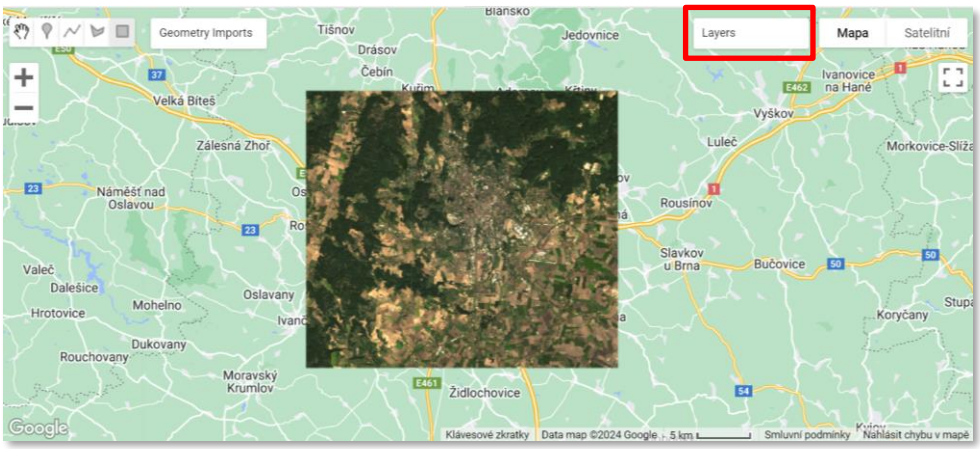
3. Vizualizace snímku

Snímek si ještě zobrazíme v pravých barvách v mapovém okně pomocí **Map.addLayer()**. Pro vizualizaci v pravých barvách využijeme pásma SR_B4, SR_B3 a SR_B2. Rozsah hodnot pro vizualizaci nastavíme na 7000-15000.

Snímek přiblížíme díky **Map.centerObject()**.

```
1. // Vizualizace snímku v RGB
2. Map.addLayer(snimekLandsat, {bands: ["SR_B4", "SR_B3", "SR_B2"], min: 7000, max: 15000},
"Snímek RGB");
3.
4. // Přiblížení zájmové oblasti
5. Map.centerObject(oblast, 10);
```

V případě, že by se snímek zdál přesvětlený či moc tmavý, tak si vyzkoušejte upravit rozsah vizualizačních hodnot přes ozubené kolečko u našeho snímku v záložce *Layers* v pravém horním rohu mapového okna.



4. Analýza hlavních komponent

Před samotnou klasifikací provedeme ještě analýzu hlavních komponent, která se řadí mezi spektrální zvýraznění obrazu

Z přichystaného snímku vybereme pásma, která využijeme při PCA. Jedná se o SR_B2, SR_B3, SR_B4, SR_B5, SR_B6, SR_B7.

```
1. // Výběr pásem potřebných pro PCA
2. var snimekLandsat = snimekLandsat
3.                     .select(["SR_B2", "SR_B3", "SR_B4", "SR_B5", "SR_B6", "SR_B7"]);
```

Pro provedení analýzy hlavních komponent si definujeme novou funkci **vypocetHlavnichKomponent()**. Následující funkci si zkopírujte do svého vlastního pracovního prostředí a upravte dle své potřeby. Zkusíme si, ale tento postup alespoň trochu přiblížit.

Nejprve dojde k přípravě dat na samotnou PCA. To zahrnuje stanovení měřítka, seznamu pásem snímku a výpočet standardizovaných hodnot. Standardizované hodnoty jsou

koncentrovány kolem nuly a jsou získány na základě výpočtu průměrných hodnot ze snímku a jejich odečtení od původního snímku.

Dále je v rámci funkce na výpočet hlavních komponent definována také funkce na tvorbu nových pásem, která jako parametr bere předponu, kterou bude mít každé nově vytvořené pásmo snímku. K zadané předponě bude přidáno i pořadové číslo pásma.

Poté následuje výpočet kovariance mezi pásmy snímku a rozložení kovarianční matice na vlastní vektory a hodnoty. Jedná se o klíčový krok identifikace hlavních komponent.

Následně dojde k výpočtu hlavních komponent jako lineární kombinace původních proměnných/pásem a vlastních vektorů.

Výsledkem funkce na výpočet hlavních komponent bude snímek s hlavními komponentami, který je vícepásmový a je normalizovaný pomocí směrodatné odchylky vlastních hodnot.

```
var vypocetHlavnichKomponent = function(snimek, uzemi) {
  // Informace o pásmech a měřítku
  var scale = 30;
  var pasma = snimek.bandNames();
  var prumery = snimek.reduceRegion({
      reducer: ee.Reducer.mean(),
      geometry: uzemi,
      scale: scale,
      maxPixels: 1e9
  });
  var prumerneHodnoty = ee.Image.constant(prumery.values(pasma));
  var koncentrovaneHodnoty = snimek.subtract(prumerneHodnoty);

  // Funkce na tvorbu nových pásem
  var novePasma = function(predpona) {
    var seq = ee.List.sequence(1, pasma.length());
    return seq.map(function(b) {
      return ee.String(predpona).cat(ee.Number(b).int());
    });
  };

  // Sloučíme pásma obrazu do 1D pole na každý pixel
  var pole1D = koncentrovaneHodnoty.toArray();

  // Výpočet kovariance
  var kovariance = pole1D.reduceRegion({
    reducer: ee.Reducer.centeredCovariance(),
    geometry: oblast,
    scale: scale,
    maxPixels: 1e9
  });
  var kovariancePole = ee.Array(kovariance.get("array"));

  // Provedení Eigenovy analýzy (metoda používána v PCA) -> oddělení vektorů a hodnot
  var eigenAnalyza = kovariancePole.eigen();
  var eigenHodnoty = eigenAnalyza.slice(1, 0, 1);
  var eigenVektory = eigenAnalyza.slice(1, 1);

  // Převod snímku s 1D poli na 2D pole
```

```
40.    var pole2D = pole1D.toArray(1);
41.
42.    // Násobení snímku s 2D poli a matice Eigen vektorů
43.    var hlavniKomponenty = ee.Image(eigenVektory).matrixMultiply(pole2D);
44.
45.    // Transformace hodnot směrodatné odchylky z Eigen hodnot na obrazový snímek typu P-band
46.    var sdSnimek = ee.Image(eigenHodnoty.sqrt())
47.      .arrayProject([0]).arrayFlatten([novePasma("sd")]);
48.
49.    return hlavniKomponenty
50.      .arrayProject([0]) // Odstranění nepotřebných dimenzí, [[]] -> [].
51.      .arrayFlatten([novePasma("pc")])// Převod jednopásmového snímku na multi-band snímek
52.      .divide(sdSnimek);
53. };
```

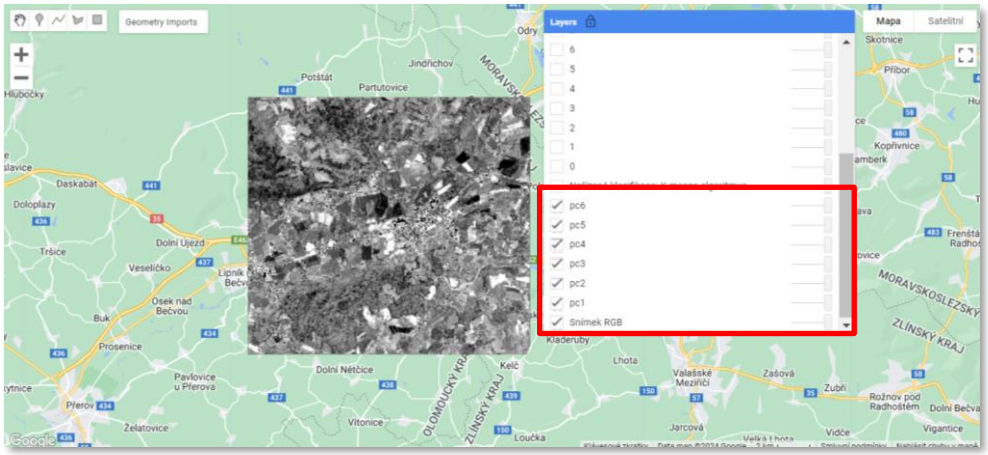
Následně funkci **vypocetHlavnichKomponent()** použijeme na snímek s vybranými pásmy.

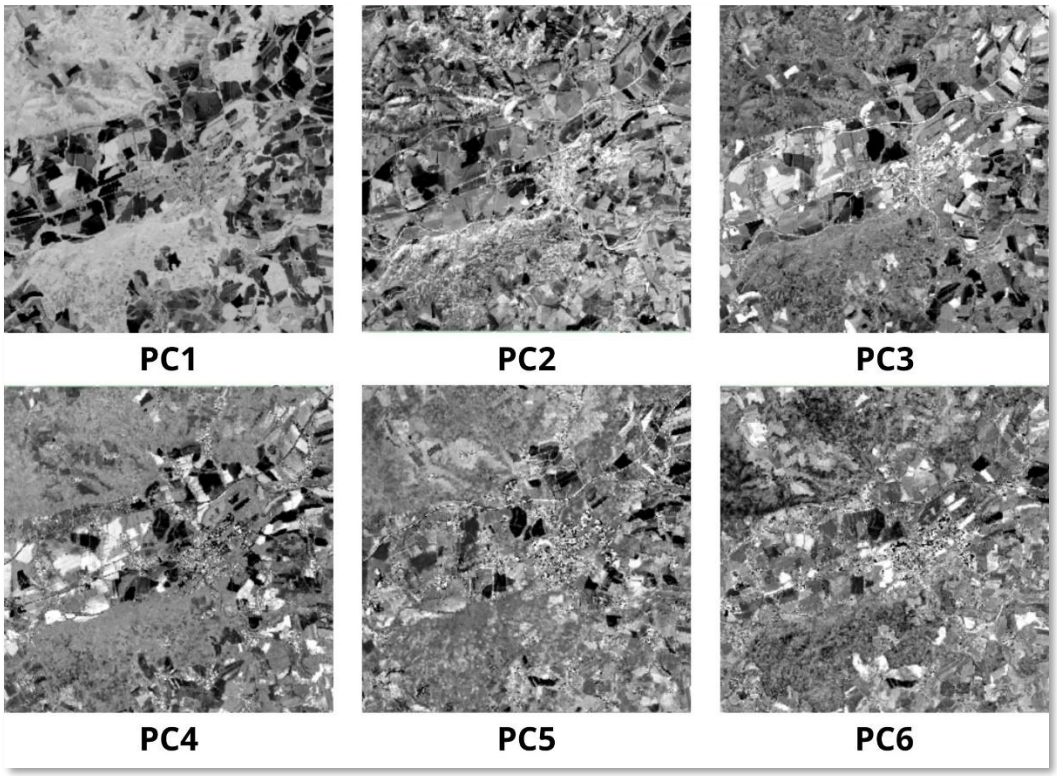
```
1. // Provedení PCA nad snímkem
2. var PCA = vypocetHlavnichKomponent(snimekLandsat, oblast);
```

Definujeme si cyklus, který provede vykreslení jednotlivých komponent do mapového okna. Využijeme cyklus **for**, kterému specifikujeme počet opakování. Počet opakování odpovídá počtu hlavních komponent.

```
1. // Vykreslení jednotlivých komponent
2. for (var i = 0; i < snimekLandsat.bandNames().length().getInfo(); i++) {
3.   var pasmo = PCA.bandNames().get(i).getInfo();
4.   Map.addLayer(PCA.select([pasmo]), {min: -2, max: 2}, pasmo);
5. }
```

Do mapového okna se postupně vykreslí jednotlivé hlavní komponenty, mezi kterými lze přecházet v záložce *Layers* v mapovém okně.





Vybere prvních pět hlavních komponent pomocí **select()** a uložíme je do nové proměnné. Poslední komponentu budeme ignorovat, protože obsahuje nejméně informací.

```
1. // Vybereme 5 hlavních komponent
2. var vybranePCA = PCA.select("pc1", "pc2", "pc3", "pc4", "pc5");
```

5. K-Means algoritmus

I přestože se jedná o neřízenou klasifikaci, tak implementace algoritmu K-means v Google Earth Engine vyžaduje definici trénovací množiny. Trénovací množina však může odpovídat celé zakreslené zájmové oblasti. Množinu definujeme nad snímkem s pěti hlavními komponentami s využitím **sample()**.

```
1. // Definice trénovací množiny
2. var trenovaciMnozina = vybranePCA.sample({
3.   region: oblast,
4.   scale: 30,
5.   numPixels: 5000
6. });
```

Dále vytvoříme instanci K-means pomocí **ee.Clusterer.wekaKMeans()** a natrénujeme ji s využitím **train()** na tréninkové množině. Algoritmu **ee.Clusterer.wekaKMeans()** zadáváme požadovaný počet spektrálních tříd. U ukázky se bude jednat o 15 spektrálních tříd.

```
1. // Vytvoření K-means a jeho trénink
```

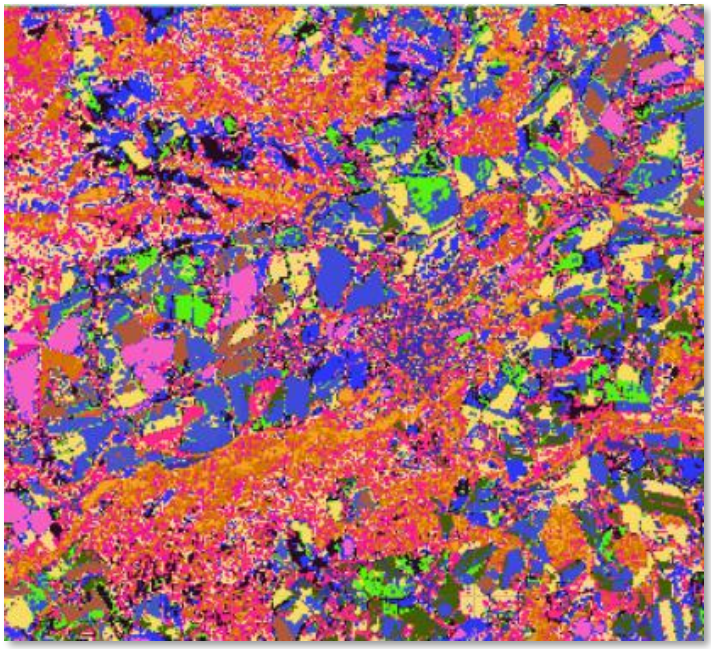
```
2. var kmeans = ee.Clusterer.wekaKMeans(15).train(trenovaciMnozina);
```

Teď už provedeme samotnou klasifikaci, což znamená, že naši instanci K-means použijeme na snímek s hlavními komponentami. K provedení neřízené klasifikace na snímku slouží funkce **cluster()**, které zadáme definovanou instanci K-means.

Samotný výsledek klasifikace zobrazíme s využitím **randomVisualizer()** v náhodných barvách v mapovém okně.

```
1. // Provedení neřízené klasifikace s K-means a vizualizace
2. var kmeansVysledek = vybranePCA.cluster(kmeans);
3.
4. // Vizualizace výsledku neřízené klasifikace v náhodných barvách
5.   Map.addLayer(kmeansVysledek.randomVisualizer(),   {},   "Neřízená   klasifikace:   K-means
algoritmus");
```

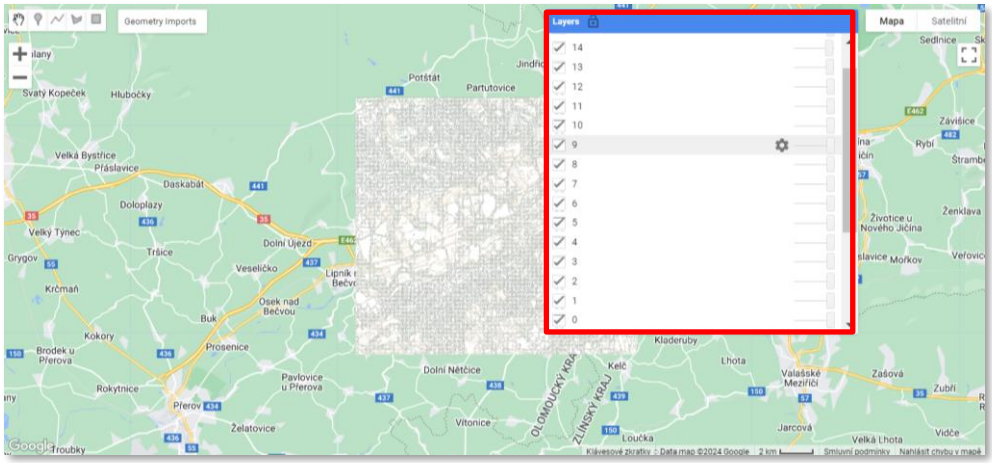
Náš klasifikovaný snímek má celkem 15 spektrálních tříd a je nutné tyto třídy převést na třídy informační.



Necháme si vykreslit jednotlivé spektrální třídy, abychom jim mohli přiřadit příslušnou informační třídu na základě porovnání s mapovým podkladem. Pro vykreslení použijeme cyklus, který v mapovém okně vykreslí každou spektrální třídu jako samostatnou vrstvu.

```
1. // Vykreslení jednotlivých spektrálních tříd
2. for (var i = 0; i < 15; i = i + 1)
3.   {var subset = kmeansVysledek.select("cluster").eq(i).selfMask();
4.   Map.addLayer(subset,{},i.toString()); }
```

V mapovém okně se nám objeví 15 nových vrstev, mezi kterými můžeme přecházet v záložce *Layers*.



Naším cílem je spektrální třídy zařadit do 5 informačních tříd (0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba, 4 – les). Postupně tedy každé spektrální třídě přiřadíme novou hodnotu pomocí vizuální interpretace s využitím snímku Landsat-9 v pravých barvách či s využitím podkladových map Google Earth Engine.

Jako první si vytvoříme seznam s hodnotami jednotlivých spektrálních tříd, tedy seznam čísel od 0-14. Následně vytvoříme seznam, do kterého budeme postupně vkládat hodnoty informačních tříd pro každou spektrální třídu.

Vždy si ponecháme aktivní jen jednu vrstvu se spektrální třídou. Začneme s první spektrální třídou a do nového seznamu zapíšeme označení odpovídající informační třídy. Takto postupujeme u každé spektrální třídy.

```
1. // Informační třídy: 0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba,
4 – les
2. var spektralniTridy = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14];
3. var informacniTridy = [3, 0, 4, 0, 2, 2, 4, 0, 0, 2, 3, 2, 4, 2, 0];
```

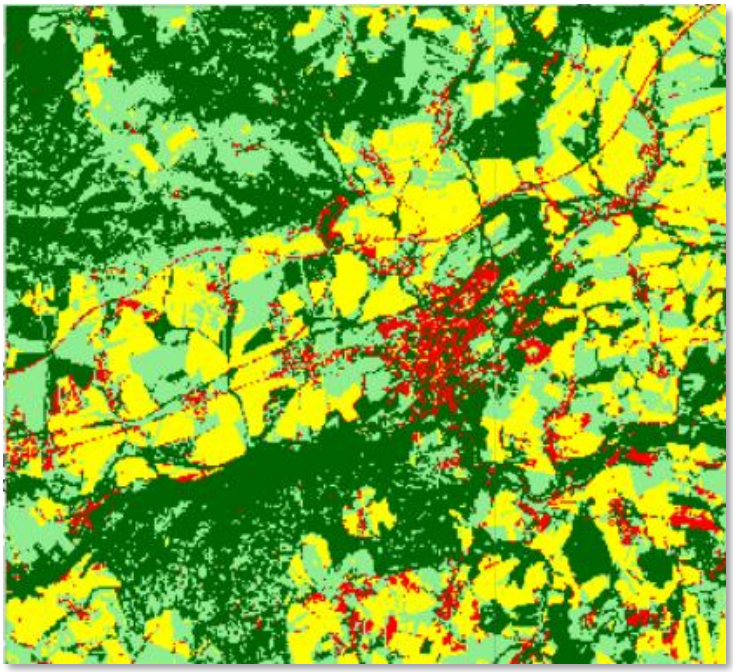
Po vytvoření seznamů přichází čas na reklasifikaci a funkci **remap()**. Funkci **remap()** zadáváme vstupní seznam, pro nás se jedná o seznam spektrálních tříd, a seznam s novými hodnotami, tedy seznam s odpovídajícími informačními třídami pro každou spektrální třídu.

```
1. // Reklasifikace spektrálních tříd na informační třídy
2. var vysledekReclass = kmeansVysledek.remap({ from: spektralniTridy, to: informacniTridy,
defaultValue: 0, bandName: "cluster" });
```

Výsledek reklasifikace si zobrazíme v mapovém okně. Funkci **Map.addLayer()** poskytneme seznam barev, které mají být použity pro vizualizaci jednotlivých informačních tříd. Vzhledem k tomu, že máme pět informačních tříd, tak specifikujeme pět barev

v odpovídajícím pořadí (0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba, 4 – les).

```
1. // Vizualizace reklasifikovaného snímku
2. Map.addLayer(vysledekReclass, { min: 0, max: 4, palette: ["yellow", "blue", "lightgreen",
"red", "darkgreen"] }, "Reklasifikovaný snímek - K-means klasifikace");
```



6. X-means algoritmus

X-means algoritmus se od K-means algoritmu liší tím, že mu místo požadovaného počtu spektrálních tříd, určujeme minimální a maximální počet spektrálních tříd, které může vytvořit. Jedná se o rozšíření K-means algoritmu, ale samotný X-means algoritmus rozhoduje o optimálním počtu spektrálních tříd pomocí hodnot Bayesova informačního kritéria.

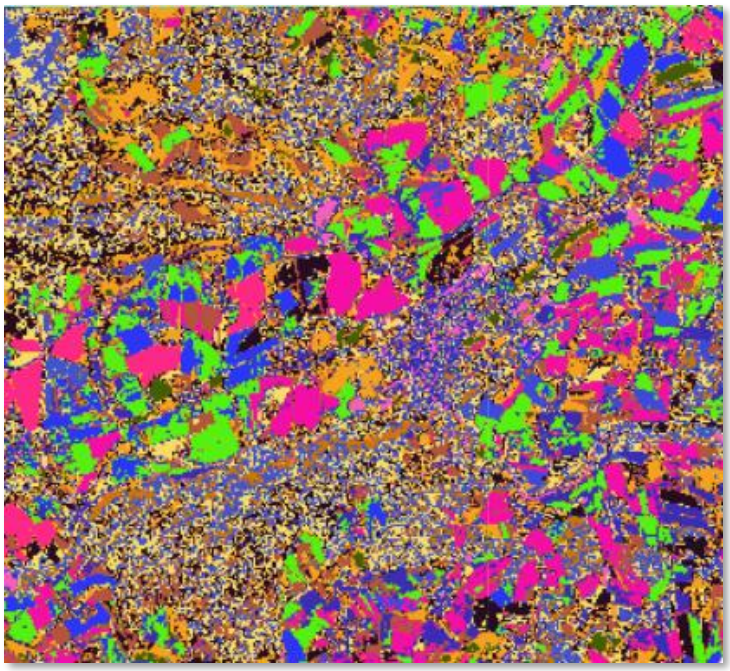
Stejně tak jako v případě K-means pracujeme s trénovací množinou, i přestože se jedná o neřízenou klasifikaci. Použijeme stejnou trénovací množinu jako v případě K-means algoritmu, tudíž není nutné ji znovu definovat.

Vytvoříme si instanci X-means pomocí **ee.Clusterer.wekaXMeans()**, které zadáme minimální a maximální počet tříd. Následně instanci shlukovače natrénujeme na trénovacích datech.

```
1. // Vytvoření X-means (jako parametry mu zadáme nejnižší a nejvyšší počet shluků) a jeho
trénink
2. var xmeans = ee.Clusterer.wekaXMeans(10, 15).train(trenovaciMnozina);
```

Pomocí **cluster()** provedeme klasifikaci snímku a zobrazíme jej v náhodných barvách v mapovém okně.

```
1. // Klasifikace snímku Landsat-9
2. var xmeansVysledek = vybranePCA.cluster(xmeans);
3.
4. // Vizualizace výsledku X-means algoritmu
5.   Map.addLayer(xmeansVysledek.randomVisualizer(),   {},   "Neřízená   klasifikace:   X-means
algoritmus");
```

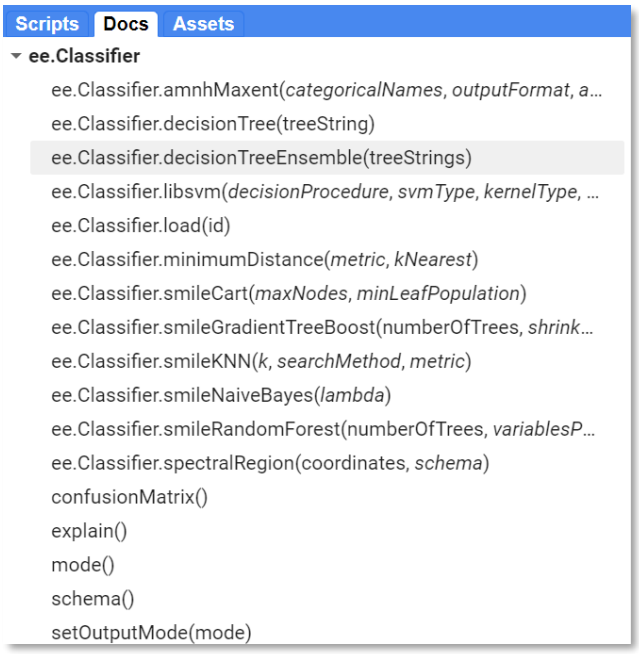


Snímek se spektrálními třídami v náhodných barvách o ničem nevypovídá. Pokud tedy chceme dodat spektrálním třídám význam a vytvořit informační třídy, tak postupujeme jako v případě K-means algoritmu. Necháme si vykreslit jednotlivé spektrální třídy, k seznamu spektrálních tříd vytvoříme seznam odpovídajících informačních tříd, a nakonec snímek reklasifikujeme a vykreslíme.

3.2 Řízená klasifikace

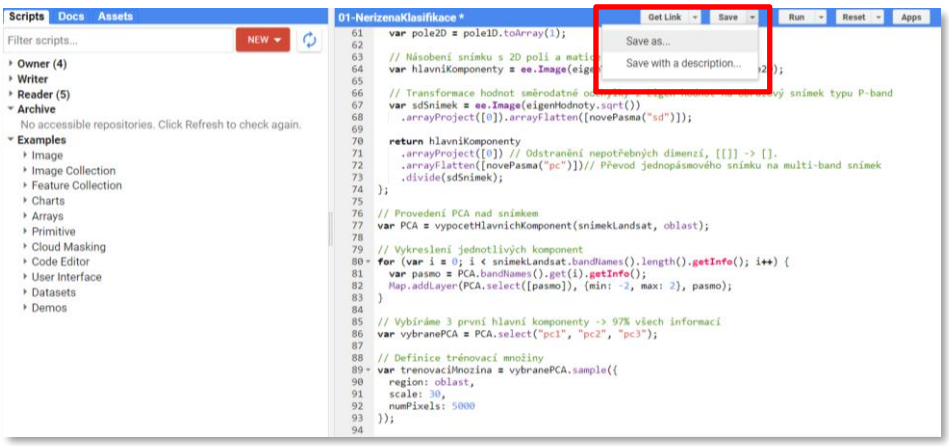
Řízená klasifikace vyžaduje trochu více práce než ta neřízená. Je nutné totiž zakreslit trénovací plochy, které budou reprezentovat jednotlivé třídy a poslouží k tréninku našeho klasifikátoru. Kromě samotné klasifikace si vyzkoušíme i odhadnout přesnost klasifikátoru na testovacích datech.

Algoritmy pro řízenou klasifikaci jsou součástí balíčku **ee.Classifier**. Google Earth Engine nabízí celou řadu algoritmů pro řízenou klasifikaci např. v porovnání s ArcGIS Pro či programem ESA Snap. Společně si vyzkoušíme klasifikátor CART (Classification and Regression Trees).



1. Příprava dat

Pro větší přehlednost si skript s postupem řízené klasifikace uložíme do nového souboru. Je zbytečné provádět znovu identický postup výběru dat a analýzy hlavních komponent. Pro usnadnění práce si tedy otevřeme skript s neřízenou klasifikací a klikneme na ikonu *Save as…*(uložit jako) v horní části editoru kódu.



Skript uložíme do odpovídajícího pracovního adresáře a přejmenujeme.

V editoru kódu si ponecháme pouze obsah *Imports*, kód pro výběr vhodného snímku a analýzu hlavních komponent. Náš kód by měl v této fázi vypadat následovně:

```
// Filtrace snímku zájmové oblasti ve vybraném období s nejmenší oblačností
var snimekLandsat = landsat
    .filterBounds(oblast)
    .filterDate("2023-05-01","2023-9-30")
    .sort("CLOUD_COVER")
    .first()
    .clip(oblast);

// Přiblížení zájmové oblasti
Map.centerObject(oblast, 10);

// Vizualizace snímku v RGB
Map.addLayer(snimekLandsat, {bands: ["SR_B4", "SR_B3", "SR_B2"], min: 7000, max: 15000}, "Snímek RGB");

////////////////////// ANALÝZA HLAVNÍCH KOMPONENT ///////////////////////

// Výběr pásem potřebných pro PCA
```

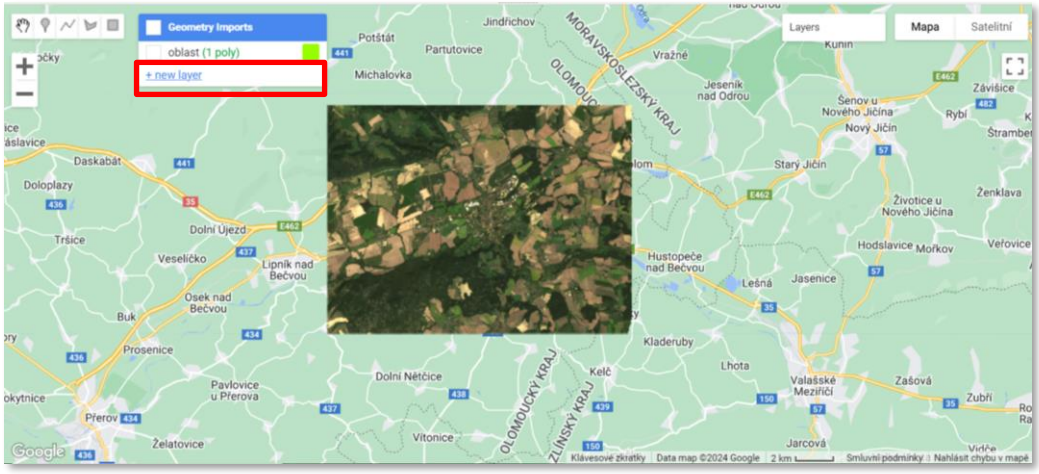
```
var snimekLandsat = snimekLandsat
                    .select(["SR_B2", "SR_B3", "SR_B4", "SR_B5", "SR_B6", "SR_B7"]);


var vypocetHlavnichKomponent = function(snimek, uzemi) {
  // Informace o pásmech a měřítku
  var scale = 30;
  var pasma = snimek.bandNames();
  var prumery = snimek.reduceRegion({
      reducer: ee.Reducer.mean(),
      geometry: uzemi,
      scale: scale,
      maxPixels: 1e9
  });
  var prumerneHodnoty = ee.Image.constant(prumery.values(pasma));
  var koncentrovaneHodnoty = snimek.subtract(prumerneHodnoty);

  // Funkce na tvorbu nových pásem
  var novePasma = function(predpona) {
    var seq = ee.List.sequence(1, pasma.length());
    return seq.map(function(b) {
      return ee.String(predpona).cat(ee.Number(b).int());
    });
  };

  // Sloučíme pásma obrazu do 1D pole na každý pixel
  var pole1D = koncentrovaneHodnoty.toArray();

  // Výpočet kovariance
  var kovariance = pole1D.reduceRegion({
    reducer: ee.Reducer.centeredCovariance(),
    geometry: oblast,
    scale: scale,
    maxPixels: 1e9
  });
  var kovariancePole = ee.Array(kovariance.get("array"));

  // Provedení Eigenovy analýzy (metoda používána v PCA) -> oddělení vektorů a hodnot
  var eigenAnalyza = kovariancePole.eigen();
  var eigenHodnoty = eigenAnalyza.slice(1, 0, 1);
  var eigenVektory = eigenAnalyza.slice(1, 1);

  // Převod snímku s 1D poli na 2D pole
  var pole2D = pole1D.toArray(1);

  // Násobení snímku s 2D poli a matice Eigen vektorů
  var hlavniKomponenty = ee.Image(eigenVektory).matrixMultiply(pole2D);

  // Transformace hodnot směrodatné odchylky z Eigen hodnot na obrazový snímek typu P-band
  var sdSnimek = ee.Image(eigenHodnoty.sqrt())
    .arrayProject([0]).arrayFlatten([novePasma("sd")]);

  return hlavniKomponenty
    .arrayProject([0]) // Odstranění nepotřebných dimenzí, [[]] -> [].
    .arrayFlatten([novePasma("pc")])// Převod jednopásmového snímku na multi-band snímek
    .divide(sdSnimek);
};

// Provedení PCA nad snímkem
var PCA = vypocetHlavnichKomponent(snimekLandsat, oblast);

// Vykreslení jednotlivých komponent
for (var i = 0; i < snimekLandsat.bandNames().length().getInfo(); i++) {
  var pasmo = PCA.bandNames().get(i).getInfo();
  Map.addLayer(PCA.select([pasmo]), {min: -2, max: 2}, pasmo);
}
// Vybíráme 5 hlavních komponent
```

```
85. var vybranePCA = PCA.select("pc1", "pc2", "pc3", "pc4", "pc5",);
```

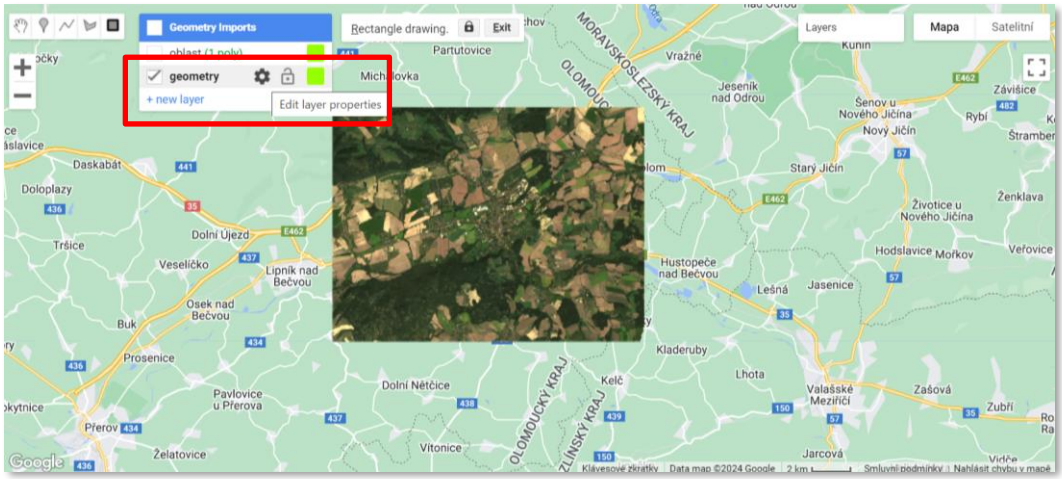
2. Tvorba trénovací množiny

V rámci řízené klasifikace je našim prvním úkolem zakreslení jednotlivých trénovacích ploch. Potřebujeme zakreslit plochy, které budou reprezentovat 5 informačních tříd: 0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba, 4 – les.

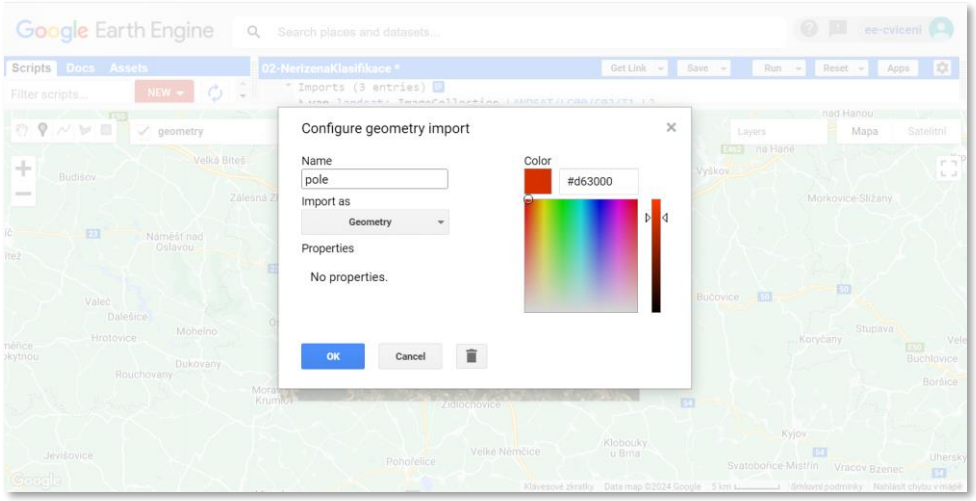
V mapovém okně přejdeme do záložky *Geometry Imports* a vybereme možnost *+new layer*.



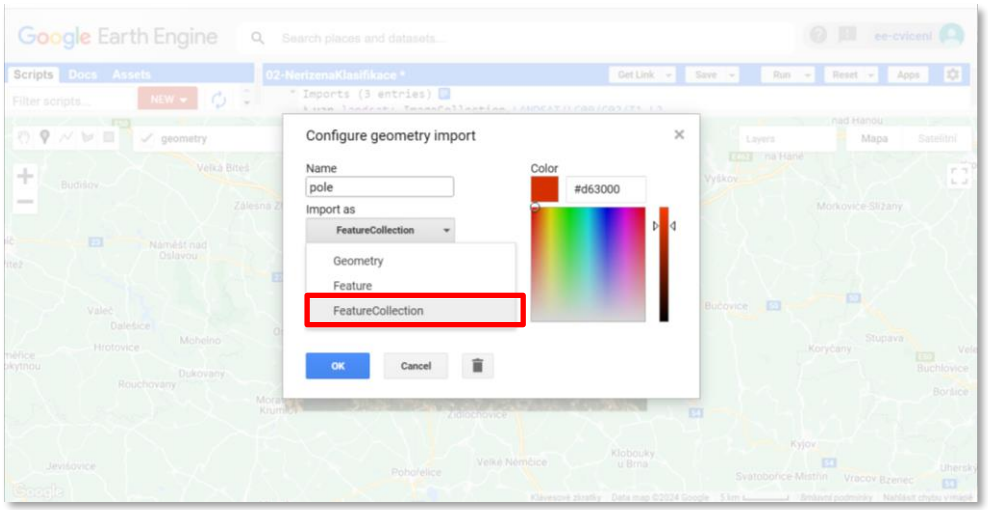
Vznikne nám nová vrstva **geometry**, u které si rozklikneme její vlastnosti pomocí ozubeného kolečka vedle názvu vrstvy.



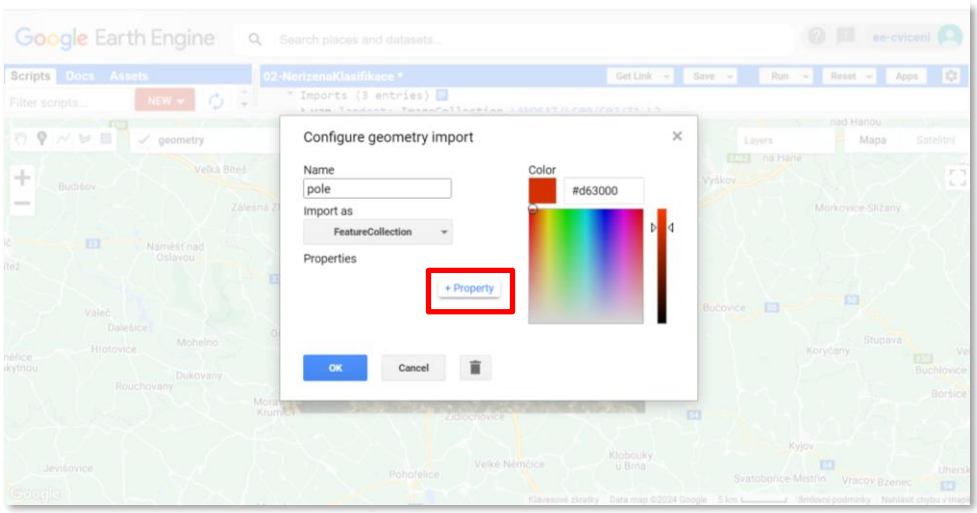
Vrstvu přejmenujeme podle toho, kterou informační třídu bude reprezentovat. První vytvoříme vrstvu pro třídu pole.



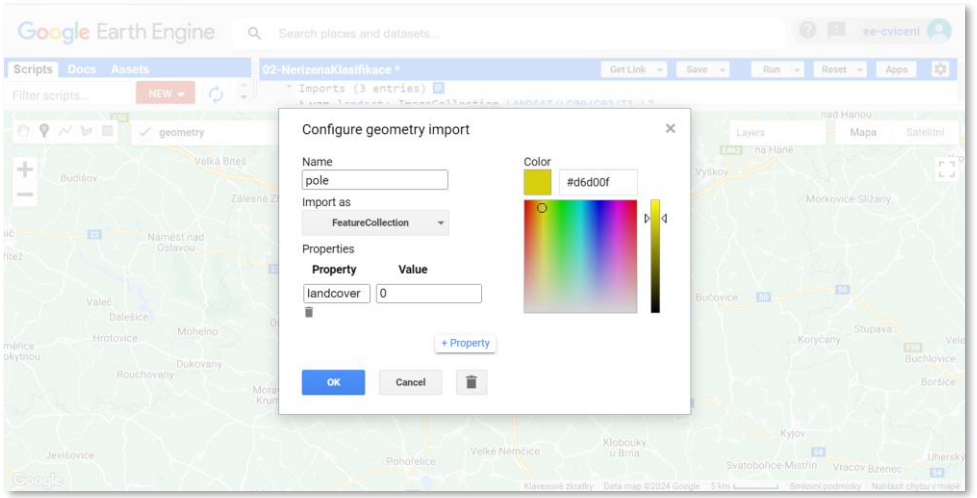
Naše nová vrstva bude reprezentovat kolekci prvků, a proto je nutné z rozbalovací menu u *Import as* vybrat možnost *FeatureCollection*.



Nově vzniklé vrstvě přiřadíme novou vlastnost s názvem **landcover**, která bude obsahovat ID třídy. ID třídy není nic jiného než číselné označení našich tříd: 0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba, 4 – les. Tuto novou vlastnost vytvoříme pomocí ikony *+Property*.



Nově vzniklou vlastnost pojmenujeme **landcover** a její hodnota bude odpovídat ID třídy. V případě pole se tedy bude jednat o číslo 0. Nakonec už jen v sekci *Color* můžeme změnit barvu zobrazení třídy a konečné nastavení uložíme pomocí ikony *OK*.



V levém horním rohu mapového pole zvolíme nástroj pro nakreslení tvaru  a postupně zakreslíme plochy pole. Snažíme se trénovací plochy rozmístit po celém snímku. Tvoříme homogenní plochy, které odpovídají zakreslované třídě.

Obdobným způsobem vytvoříme vrstvy s plochami stejného typu pro zbylé třídy.

Veškeré tréninkové plochy spojíme do jedné souhrnné kolekce funkcí **ee.FeatureCollection()**.

```
1. // Kolekce snímků s reprezentanty našich tříd
2. var treninkPrvky = ee.FeatureCollection([pole, vodstvo, louky, zastavba, lesy]).flatten();
```

Před vytvořením trénovací množiny specifikujeme pásma, která budou využita při klasifikaci. Budou to tedy vybrané hlavní komponenty – pc1, pc2, pc3, pc4 a pc5.

```
1. // Seznam pásem, které budou využity při klasifikaci
2. var klasifikacniPasma = ["pc1", "pc2", "pc3", "pc4", "pc5"];
```

Teď už vytvoříme samotnou trénovací množinu. Ze snímku s hlavními komponentami vybereme pásma pomocí **select()**, pak pomocí **sampleRegions()** vybereme ze snímku pouze pixely, které odpovídají našim zakresleným trénovacím plochám. Funkci **sampleRegions()** zadáváme kolekci s trénovacími plochami, název atributu s hodnotou ID třídy a měřítko.

```
1. // Vytvoření tréninkové množiny
```

```
var treninkovaMnozina = vybranePCA.select(klasifikacniPasma)
    .sampleRegions({
        collection: treninkPrvky,
        properties: ["landcover"],
        scale: 30
    });
```

Protože v rámci řízené klasifikace si vyzkoušíme i vyhodnotit přesnost použitého klasifikátoru, je nezbytné si trénovací množinu rozdělit na plochy pro trénink klasifikátoru a plochy pro testování přesnosti klasifikátoru. Provedeme rozdělení množiny trénovacích ploch na 70 % trénovacích dat a 30 % testovacích dat pomocí následujícího kódu:

```
// Rozdělení naší tréninkové množiny na dvě - jedna bude využita pro trénink, druhá pro ověření přesnosti klasifikátoru
var mnozinaRandom = treninkovaMnozina.randomColumn("random");
var rozdeleni = 0.7;
var treninkovaMnozina2 = mnozinaRandom.filter(ee.Filter.lt("random", rozdeleni));
var testovaciMnozina = mnozinaRandom.filter(ee.Filter.gte("random", rozdeleni));
```

3. Trénink klasifikátoru

Jak již bylo zmíněno, tak řízenou klasifikaci provedeme s využitím algoritmu CART. CART při klasifikačních úlohách rekurzivně dělí data na stále menší a menší podmnožiny. Cílem je vytvořit takové podmnožiny, které jsou co nejčistší, a tedy dochází k nejmenšímu promíchání dat v podmnožině. CART vytváří stromovou strukturu, kdy uzly představují body rozhodování a hrany představují možné výsledky. Algoritmus v každém uzlu vybírá tu možnost, která nejlépe snižuje nečistotu výsledných podmnožin.

Před klasifikací celého snímku natrénujeme náš klasifikátor na trénovacích datech. CART klasifikátor najdeme v Google Earth Engine pod metodou **ee.Classifier.smileCart().** Metodě **train()** pak zadáme trénovací množinu, atribut, který nese ID třídy, a pásma, která mají být využita pro trénink klasifikátoru.

```
// Trénink CART klasifikátoru
var klasifikator = ee.Classifier.smileCart().train({
    features: treninkovaMnozina2,
    classProperty: "landcover",
    inputProperties: klasifikacniPasma
});
```

4. Klasifikace snímku

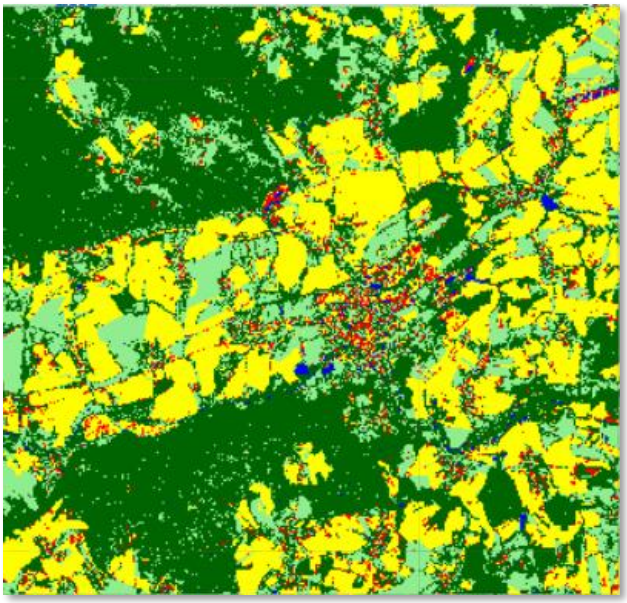
Po natrénování klasifikátoru už jej použijeme na klasifikaci snímku. Specifikujeme pásma potřebná pro klasifikaci pomocí **select()** a následně použijeme náš klasifikátor. U neřízené klasifikace jsme používali metodu **cluster()**, při řízené klasifikaci snímku se však používá funkce **classify()**. Funkci **classify()** zadáváme název natrénovaného klasifikátoru.

```
// Klasifikace našeho snímku
```

```
2. var klasifikovanySnimek = vybranePCA.select(klasifikacniPasma).classify(klasifikator);
```

Klasifikovaný snímek si samozřejmě zobrazte v mapovém okně. Metodě **Map.addLayer()** poskytneme paletu barev pro zobrazení výsledných tříd. Pořadí barev odpovídá pořadí jednotlivých tříd dle hodnoty atributu **landcover**. Pokud jste postupovali stejně jako v ukázce pak 0 – pole bez vegetace, 1 – voda, 2 – louka/pole s vegetací, 3 – zástavba, 4 – les.

```
1. // Zobrazení klasifikovaného snímku
2. Map.addLayer(klasifikovanySnimek, {min: 0, max: 4, palette: ["yellow", "blue", "lightgreen",
"red", "darkgreen"]}, "Řízená klasifikace - CART");
```



5. Odhad přesnosti klasifikace

Na závěr provedeme odhad přesnosti klasifikátoru, k čemuž využijeme testovací data, která jsme si již nachystali při rozdělení trénovací množiny na trénovací data a testovací data.

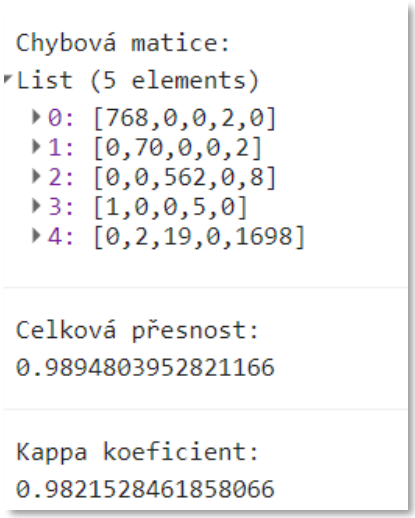
Provedeme klasifikaci testovací množiny pomocí natrénovaného klasifikátoru.

```
1. // Klasifikace testovací množiny
2. var testovaciKlasifikace = testovaciMnozina.classify(klasifikator);
```

Vytvoříme si chybovou matici za pomoci **errorMatrix()**. Vzniklá 2D chybová matice porovnává skutečnou klasifikaci území s tou predikovanou (vzniklou na základě řízené klasifikace). Chybovou matici si necháme vytisknout do konzole stejně, tak jako celkovou přesnost a Kappa index.

```
1. // Tvorba chybové matice, vytisknutí informací o celkové přesnosti + kappa indexu
2. var chybovaMatice = testovaciKlasifikace.errorMatrix("landcover", "classification");
```

```
3. print("Chybová matice:", chybovaMatice);
4. print("Celková přesnost:", chybovaMatice.accuracy());
5. print("Kappa index:", chybovaMatice.kappa());
```



3.3 Porovnání výsledků

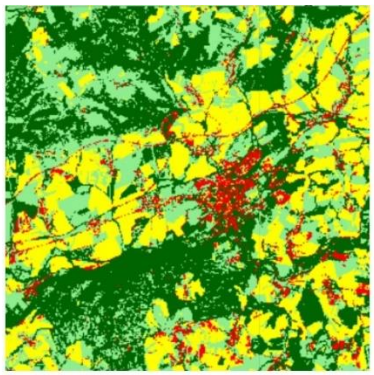
Vzhledem k tomu, že jsme v rámci neřízené klasifikace neprovedli odhad přesnosti klasifikace, tak lze porovnat výsledky neřízené a řízené klasifikace pouze na základě výsledného obrazu.

U ukázkového skriptu se při neřízené klasifikaci nepovedlo detekovat vodní plochy, avšak v rámci řízené klasifikace byly úspěšně nalezeny i vodní plochy s menší rozlohou. Naopak výsledek neřízené klasifikace lépe zobrazuje detailní strukturu lesů.

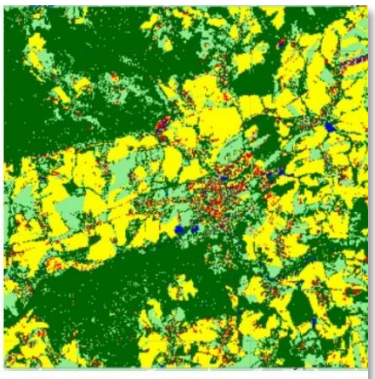
Jistě vhodnější formou porovnání výsledků klasifikace by byl odhad přesnosti jak u neřízené klasifikace, tak u klasifikace řízené. Z výsledků odhadu přesnosti u řízené klasifikace u ukázkového skriptu však víme, že i přes drobné nedostatky byl CART klasifikátor velmi přesný.

U obou výsledků klasifikací lze pozorovat místy strukturu pepř a sůl, a proto by bylo vhodné provést post-klasifikační úpravy. V příštím cvičení si ukážeme ohniskové operace, které bychom mohli využít pro vyhlazení obrazu a eliminovat strukturu pepř a sůl.

Snímek RGB



Neřízená klasifikace



Řízená klasifikace