Google Earth Engine ile SAR Görüntüleri Kullanarak Otomatik Sel/Taşkın Haritalama: Düzce/Gölyaka İlçesi Örneği

Dünya üzerindeki en yaygın ve yıkıcı doğal afetlerden birisi olan sel/taşkın olayları büyük ölçüde can ve mal kaybına yol açmaktadır. Ayrıca iklim değişikliğinin de etkisiyle son yıllarda daha sık ve tahmin edilemez hâle gelmiştir. Can ve mal kaybını en aza indirmek için taşkın yayılımını hızlı bir şekilde tespit etmek ve gerekli müdahaleyi zamanında yapmak önemlidir. Taşkın yayılımı tespiti için çoğunlukla optik sensörlerden alınan görüntüler veya SAR (Sentetik Aralıklı Radar) görüntüleri kullanılır. Ancak SAR sistemi her türlü hava koşulunda gece-gündüz veri toplayabilmesi ile optik sensörlere üstünlük sağlamaktadır. Bu, taşkın olayları esnasındaki hava durumu düşünüldüğünde oldukça önemlidir. Ayrıca kullanılan görüntülerin yüksek boyutlara sahip olması sel/taşkın yayılım tespitini geciktirmekte ve zamanında müdahaleyi zorlaştırmaktadır. Bu çalışmada buna çözüm olarak Google Earth Engine ile Sentinel-1 SAR görüntülerini kullanarak 10/07/2023 tarihinde Düzce/Gölyaka'da meydana gelen selin yayılımını tespit edeceğiz.

Öncelikle https://code.earthengine.google.com/ sitesine gidip Google hesabımız ile yeni bir proje açıyoruz. Analizi hızlandırmak ve sadece işimize yarayan alanları incelemek için projeye sınır verisi ekleyeceğiz. Bu çalışma için sınır verisini Harita Genel Müdürlüğü'nün web sitesinden indirdim ve Gölyaka ilçe sınırlarını kapsayacak şekilde ArcGIS içerisinde kestim.

Sınır verisini projeye eklemek için *Assets -> NEW -> Shape files* tıklıyoruz.

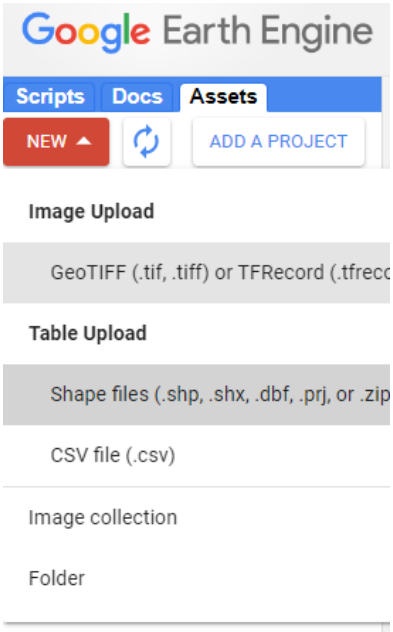

Fig. 1: Sınır verisinin projeye eklenmesi

SELECT tıklayıp sınır verisine ait dosyaları seçiyoruz.

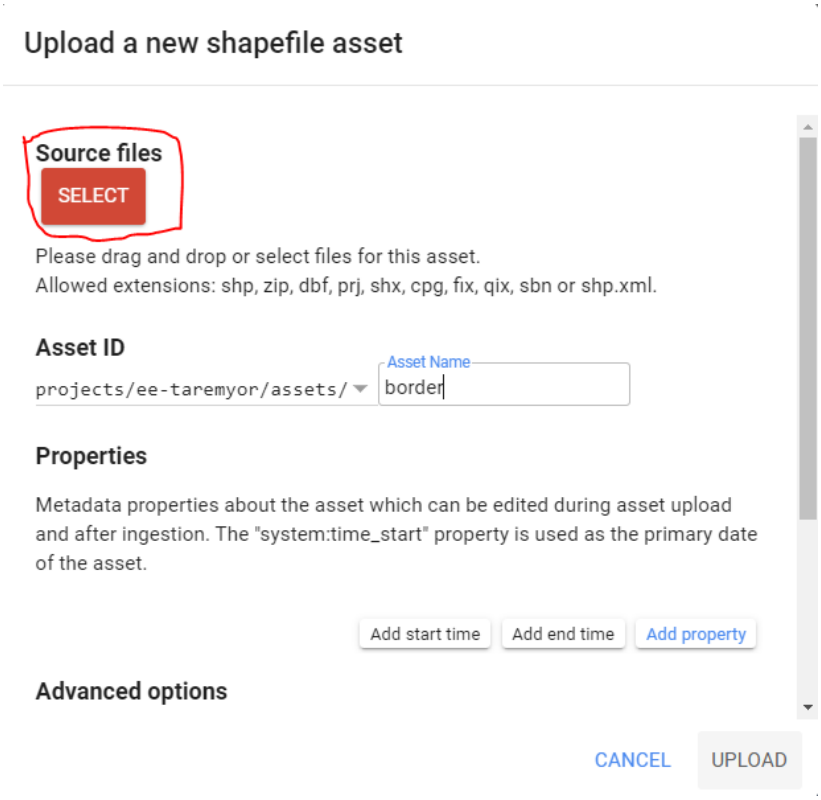

Fig. 2: Sınır verisini "border" olarak isimlendirdik

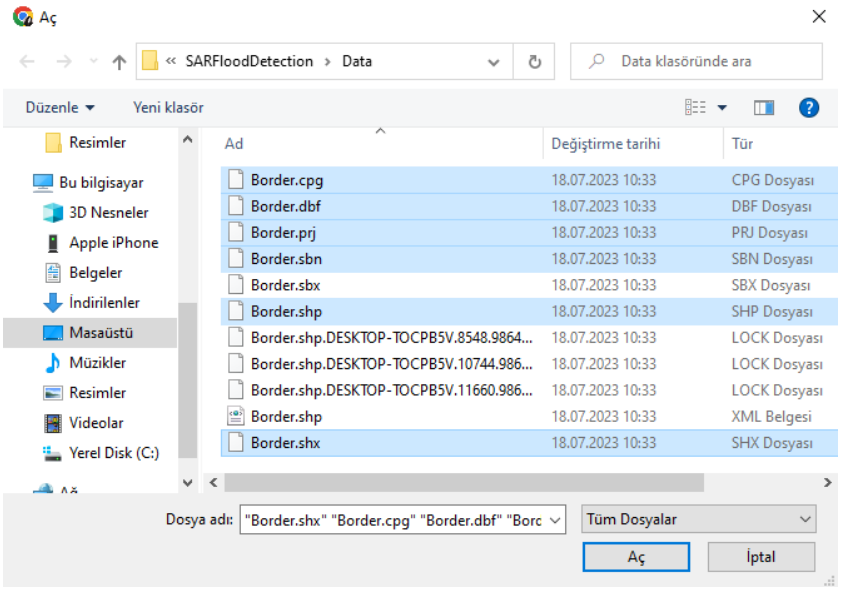

Fig. 3: Sınır verisine ait dosyaları seçiyoruz

UPLOAD tıkladığımızda *"Tasks"* penceresinde verinin yüklenmekte olduğunu göreceğiz.

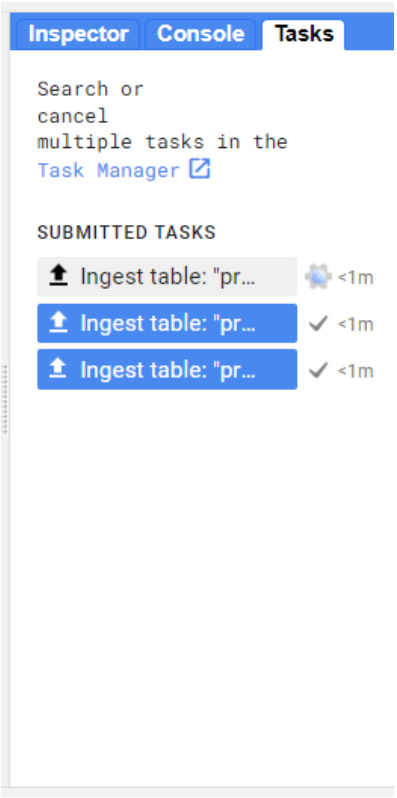

Fig. 4: Yükleme işleminin durumunu kontrol etmek için Tasks penceresini inceleyebiliriz

Yükleme işlemi tamamlandığında *"Assets"* penceresinde *'Yenile'* butonuna tıkladığımızda *"border"* isimli verinin yüklenmiş olduğunu göreceğiz.

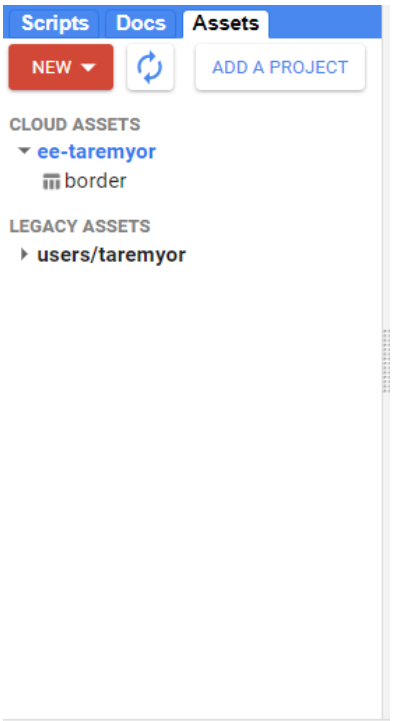

Fig. 5: Sınır verisi artık proje içerisinde kullanılmaya hazır

Veriyi projemizde kullanabilmek için “Import into Script” tıklıyoruz. Ardından da arama kutusuna “Sentinel 1” yazıyoruz ve aşağıdaki şekilde projemize dahil ediyoruz;

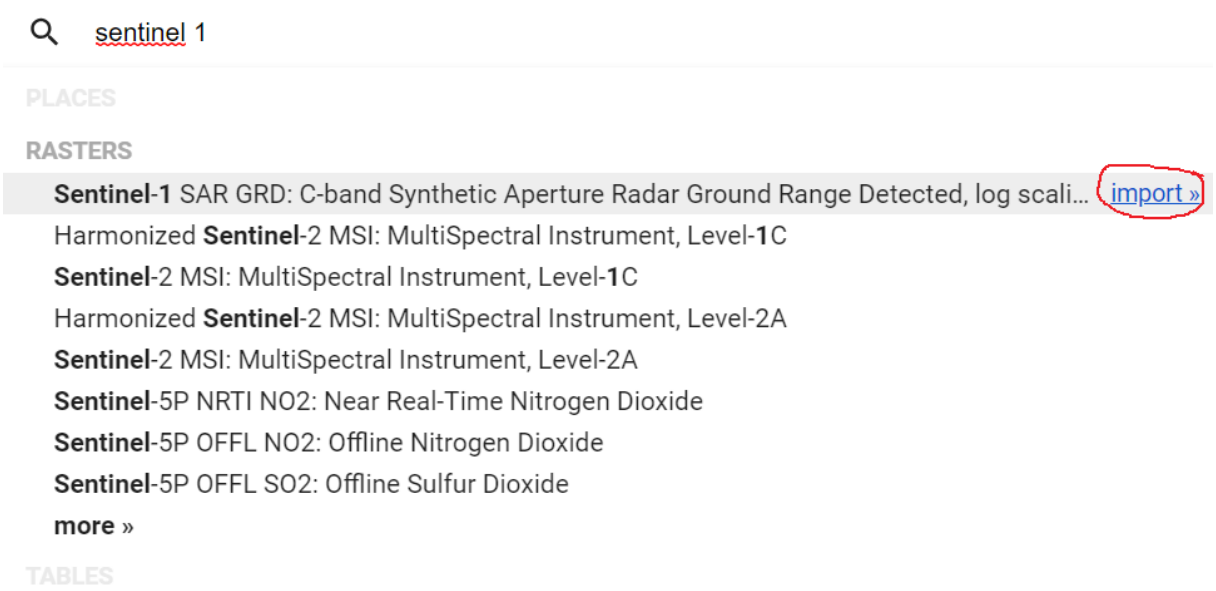

Fig. 6: SAR görüntülerini projede kullanabilmek için bu işlemi yapmamız gerekiyor

Projemizdeki kodlar aşağıdaki gibi görünmeli;

```
var Sentinel_Sar: ImageCollection COPERNICUS/S1_GRD
var border: Table projects/ee-nickname/assets/border

Map.addLayer(border, {}, "Study Area");
```

Kodu çalıştırdığımızda alttaki haritada sınır verimizi görebiliriz;

Fig. 7: Sınır verisinin harita üzerinde görüntülenmesi

Bölgeye ait görüntülerin seçiminde kullanılacak filtreleri belirlemek için literatürden faydalandım. Bu filtreleri uygulamak için aşağıdaki kodu kullanıyoruz;

```
var collection = ee.ImageCollection(Sentinel_Sar)
                            .filterBounds(border)

.filter(ee.Filter.listContains('transmitterReceiverPolarisation','VV'))
                            .filter(ee.Filter.eq('instrumentMode','IW'))

.filter(ee.Filter.eq('orbitProperties_pass','DESCENDING'));
```

Çalışma kapsamında taşkın öncesi ve sonrasına ait birer görüntü kullanmak istiyoruz. Bunun için zaman filtresi kullanmamız gerekiyor;

```
var beforeStart = '2023-06-27';
var beforeEnd = '2023-07-07';

var afterStart = '2023-07-09';
var afterEnd = '2023-07-15';


var beforeCollection = collection.filterDate(beforeStart, beforeEnd);
var afterCollection = collection.filterDate(afterStart, afterEnd);
```

Kodu çalıştırdığımızda doğrudan çalışma alanını görmek için;

```
Map.centerObject(border);
```

Taşkın öncesi-sonrası görüntüleri haritada görüntüleyebilmek için;

```
Map.addLayer(before,{min:-30,max:0},'Before');
Map.addLayer(after,{min:-30,max:0},'After');
```

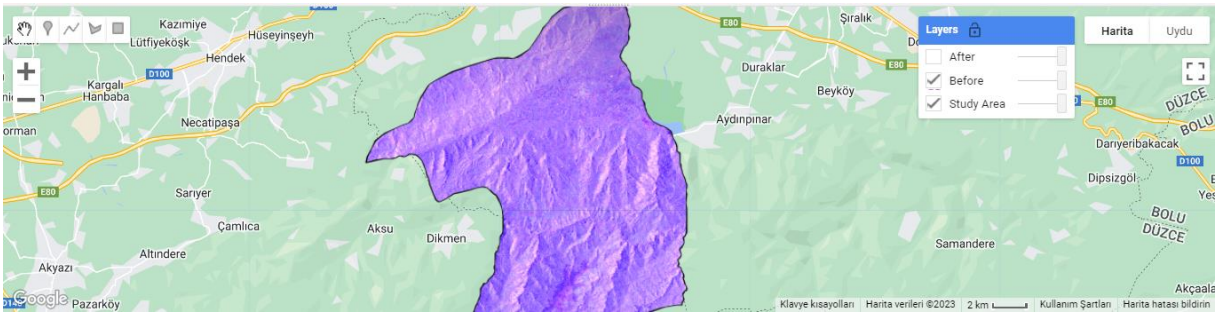

Fig. 8: Taşkın öncesine ait görüntü

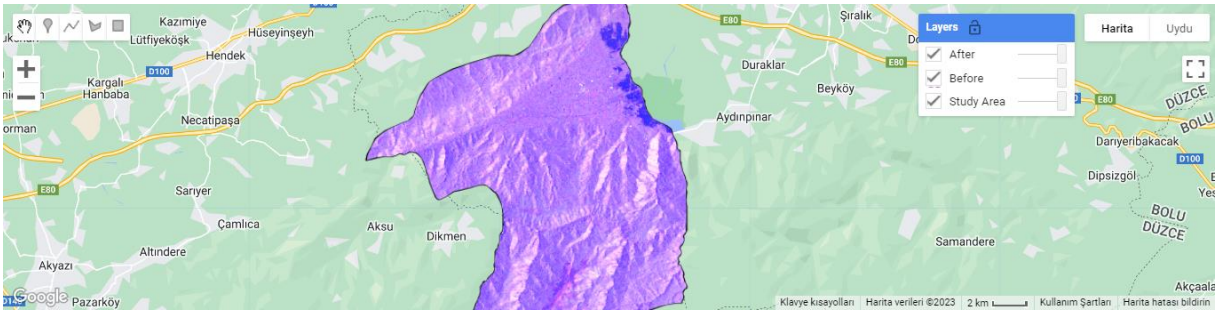

Fig. 9: Taşkın sonrasına ait görüntü

Sentinel-1 görüntülerinde mevcut olan tuz-biber etkisi, analizin performansının düşmesine yol açabilir. Bunu gidermek için filtre uygulayacağız. Buna göre 'NULL' değerine sahip her bir

piksel için arka planda 30 metre yarıçapında bir çember çizilecek ve içinde kalan piksellerin değerlerinin ortalaması söz konusu piksel için uygulanacak.

Filtreleme işlemi için aşağıdaki kodu kullanıyoruz;

```
var before_s = before.focal_median(30, "circle", "meters");
var after_s = after.focal_median(30, "circle", "meters");
```

Sel/taşkın tespiti için iki görüntü arasındaki farkı tespit etmek istiyoruz;

```
var difference = after_s.subtract(before_s);
```

Fark görüntüsü içerisindeki pikselleri taşkın-taşkın olmayan şeklinde iki sınıfa ayırmak istiyoruz. Bunun için bazı çalışmalarda lokal eşik değerleri kullanılmış. Ancak yazdığımız kodu başka çalışma alanlarında da kullanabilmek için bu çalışma kapsamında gerekli eşik değerini Otsu algoritması ile belirleyeceğiz. Otsu algoritmasının uygulanması için aşağıdaki kodu kullanıyoruz;

```
var S1 = difference.clip(border);
var histogram =S1.select('VV').reduceRegion({
  reducer: ee.Reducer.histogram(255, 2),
  geometry: border,
  scale: 20,
  bestEffort: true
});
print(histogram);

print(Chart.image.histogram(S1.select('VV'), border, 20));

var otsu = function(histogram) {
  var counts = ee.Array(ee.Dictionary(histogram).get('histogram'));
  var means = ee.Array(ee.Dictionary(histogram).get('bucketMeans'));
  var size = means.length().get([0]);
  var total = counts.reduce(ee.Reducer.sum(), [0]).get([0]);
  var sum = means.multiply(counts).reduce(ee.Reducer.sum(), [0]).get([0]);
  var mean = sum.divide(total);

  var indices = ee.List.sequence(1, size);


  var bss = indices.map(function(i) {
    var aCounts = counts.slice(0, 0, i);
    var aCount = aCounts.reduce(ee.Reducer.sum(), [0]).get([0]);
    var aMeans = means.slice(0, 0, i);
    var aMean = aMeans.multiply(aCounts)
        .reduce(ee.Reducer.sum(), [0]).get([0])
        .divide(aCount);
    var bCount = total.subtract(aCount);
    var bMean = sum.subtract(aCount.multiply(aMean)).divide(bCount);
    return aCount.multiply(aMean.subtract(mean).pow(2)).add(
           bCount.multiply(bMean.subtract(mean).pow(2)));
  });

  print(ui.Chart.array.values(ee.Array(bss), 0, means));


  return means.sort(bss).get([-1]);
};
```

```
var threshold = otsu(histogram.get('VV'));
print('threshold', threshold);
```

Kodu çalıştırdığımızda elimizdeki fark görüntüsü için üretilen histogram aşağıdaki gibidir;

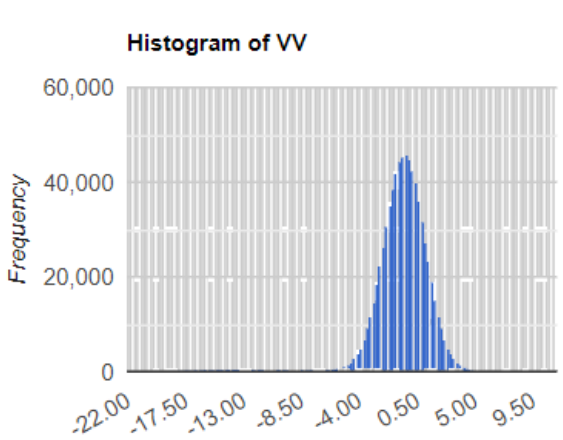

Fig. 10: Fark görüntüsüne ait histogram

Ayrıca üretilen eşik değerini de görebiliriz;

```
threshold
-6.879507657843002
```

JSON

Fig. 11: Fark görüntüsüne ait eşik değeri

Kodumuz içerisindeki ‘threshold’ değişkeni, görüntümüz için otomatik olarak belirlenen eşik değerini ifade ediyor.

Son olarak aşağıdaki kod ile sel/taşkın alanını temsil eden pikselleri haritamıza ekliyoruz;

```
var water = S1.select('VV').lt(threshold);

Map.addLayer(water.selfMask(), {palette: 'Blue'}, 'Flood');
```

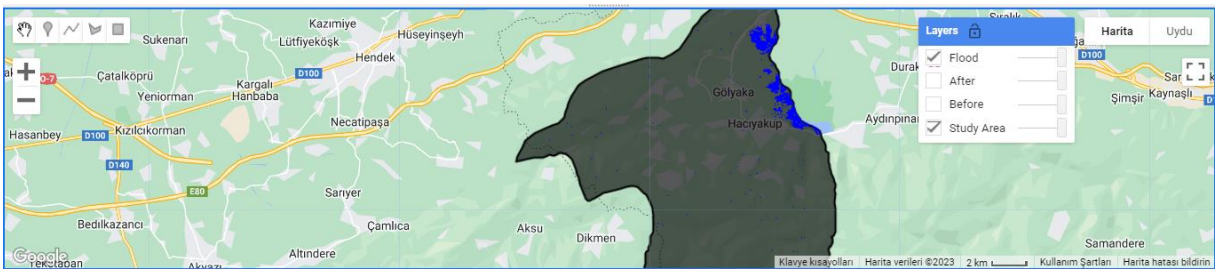

Fig. 12: Sel/taşkın alanlarının harita üzerinde gösterimi

Görüntüleri karşılaştırdığımızda elde ettiğimiz sonuç hiç de fena görünmüyor ancak yakından incelediğimizde ufak hatalar fark ediyoruz;

Fig. 13: Bu küçük hataları gidermek için literatürdeki ilave kriterler kullanılabilir

Bu çalışmada Google Earth Engine ile SAR görüntüleri ve Otsu metodunu kullanarak sel/taşkın alanlarını tespit etmeye çalıştık. Sentinel-1 uydusunun söz konusu çalışma alanı üzerinde 6 günde bir alım yapması çalışmanın en büyük limitasyonu olarak sayılabilir ve buna çözüm olarak ücretli radar görüntülerinin kullanımı düşünülebilir. Ayrıca sonuç haritasında meydana gelen ufak hataların giderilmesi için ilave olarak Sayısal Yükseklik Modeli ve Global Su Yüzeyi Veri Seti kullanılabilir. Sayısal Yükseklik Modeli ile eğim değeri 5 dereceden yüksek pikseller bu alanlarda su birikemeyeceği için maskelenebilir. Ayrıca hatalı olarak su yüzeyine denk gelen taşkın pikselleri, su yüzeyi verisi kullanarak maskelenebilir. Elde edilen sonuçlar sel/taşkın anında gerekli çözümlerin hızlı bir şekilde uygulanabilmesi için faydalı olacaktır.

Yararlandığım Kaynaklar

An exploratory study of Sentinel-1 SAR for rapid urban flood mapping on Google Earth Engine

Automatic flood detection using sentinel-1 images on the google earth engine

Flood inundation mapping- Kerala 2018; Harnessing the power of SAR, automatic threshold detection method and Google Earth Engine

Using Sentinel-1 and Google Earth Engine cloud computing for detecting historical flood hazards in tropical urban regions: a case of Dar es Salaam

GEE4FLOOD: rapid mapping of flood areas using temporal Sentinel-1 SAR images with Google Earth Engine cloud platform

Multi-Temporal Sentinel-1 SAR and Sentinel-2 MSI Data for Flood Mapping and Damage Assessment in Mozambique

Multi-temporal synthetic aperture radar flood mapping using change detection

Flood Monitoring in Rural Areas of the Pearl River Basin (China) Using Sentinel-1 SAR