7. Accuracy Assessment: Quantifying Classification Quality

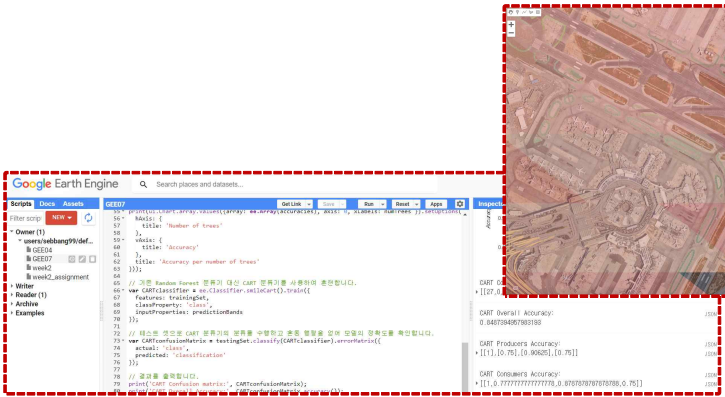
정확도 평가: 분류의 품질 (수치화, 정량화)측정

융합보안학과 윤세영

유투브 주소: https://youtu.be/wevFqcXl7wg

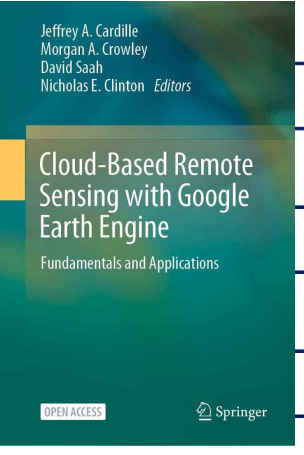
Google Earth Engine

- 지구 데이터를 대규모로 분석하고 시각화 할 수 있는 플랫폼



목차

- 이론
- 실습
- 예제 문제 풀이



개요 및 이론

- 이미지 분류(image classification)에 대한 정확도 평가(assess the accuracy)

7.1 Introduction to Theory

Any map or remotely sensed product is a generalization or model that will have inherent errors. Products derived from remotely sensed data used for scientific purposes and policymaking require a quantitative measure of accuracy to strengthen the confidence in the information generated (Foody 2002; Strahler et al. 2006; Olofsson et al. 2014). Accuracy assessment is a crucial part of any classification project, as it measures the degree to which the classification agrees with another data source that is considered to be accurate, ground-truth data (i.e., "reality").

The history of accuracy assessment reveals increasing detail and rigor in the analysis, moving from a basic visual appraisal of the derived map (Congalton 1994; Foody 2002) to the definition of best practices for sampling and response designs and the calculation of accuracy metrics (Foody 2002; Stehman 2013; Olofsson et al. 2014; Stehman and Foody 2019). The confusion matrix (also called the "error matrix") (Stehman 1997) summarizes key accuracy metrics used to assess products derived from remotely sensed data.

생성된 정보에 대한 신뢰를 위해
정확도의 정량적 측정이 필요

정확도 평가는 분류가 실제와 얼마나 일치하는지 측정

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

"Confusion Matrix"를 이용하여 정확도 평가

In Chap. 6, we asked whether the classification results were satisfactory. In remote sensing, the quantification of the answer to that question is called accuracy assessment. In the classification context, accuracy measurements are often derived from a confusion matrix.

In a thorough accuracy assessment, we think carefully about the sampling design, the response design, and the analysis (Olofsson et al. 2014). Fundamental protocols are taken into account to produce scientifically rigorous and transparent estimates of accuracy and area, which requires robust planning and time. In a standard setting, we would calculate the number of samples needed for measuring accuracy (sampling design). Here, we will focus mainly on the last step, analysis, by examining the confusion matrix and learning how to calculate the accuracy metrics. This will be done by partitioning the existing data into training and testing sets.

분류 결과 만족?
➔ 정확도 평가

정확도를 위해 기본 프로토콜이 고려

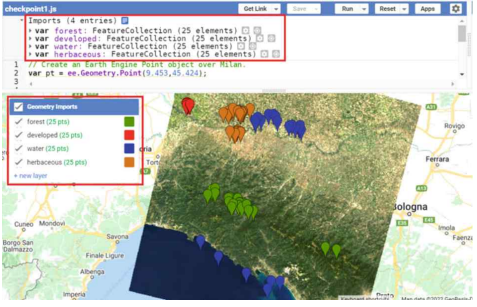
정확도를 측정하기 위해 필요한 샘플 수를 계산함

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

데이터 집합

```
var data = ee.FeatureCollection('projects/gee-book/assets/F2-2/milan_data'); //데이터셋 가져오기

var predictionBands = ['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7','ST_B10','ndvi', 'ndwi'];
//분류에 사용될 목록
```

Class	Class value
Forest	0
Developed	1
Water	2
Herbaceous	3



실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

FeatureCollection

```
var trainingTesting = data.randomColumn(); //무작위 숫자 열 추가-> 각 특징에 대해 무작위로 생성된 숫자를 포함하는 새 열 생성

var trainingSet = trainingTesting.filter(ee.Filter.lessThan('random', 0.8)); //Training : Test == 8 : 2
```

Random 열 값이 0.8보다 작은 특징을 필터링

```
var testingSet = trainingTesting.filter(ee.Filter.greaterThanOrEquals('random', 0.8));
```

Random 열 값이 0.8 이상인 특징을 필터링

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

50개의 RandomForest 생성

```
var RFclassifier = ee.Classifier.smileRandomForest(50).train({
    features: trainingSet, // training 데이터셋 사용
    classProperty: 'class',
    inputProperties: predictionBands
});
```

// 주어진 training데이터셋을 이용하여 50개의 의사결정 트리를 갖는 모델 생성
// training된 모델을 RFclassifier 변수에 할당

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

```
var confusionMatrix = testingSet.classify(RFclassifier).errorMatrix({
//test 데이터셋과 앞서 만든 Rfclassifier를 사용하여 분류
    actual: 'class', //실제
    predicted: 'classification' //예측
});
```

예측->실제 예측->실제

		Actual values	
		Positive	Negative
Predicted values	Positive	TP (true po 양성->양성	FP (false po 양성->음성
	Negative	FN (false ne 음성->양성	TN (true ne 음성->음성

confusionMatirx -> 예측된 값과 실제 값 간의 비교를 통해 분류의 정확도를 설명

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

Table 7.3 Confusion matrix for a binary classification where the classes are "positive" (forest) and "negative" (non-forest)

		Actual values	
		Positive	Negative
Predicted values	Positive	307	18
	Negative	14	661

307개: 숲으로 올바르게 식별
18개: 숲이 아닌데 숲으로 잘못 식별
14개: 숲인데 숲이 아니라고 잘못 식별
661개: 숲이 아니라고 올바르게 식별

$$\text{Overall Accuracy} = (\text{TP} + \text{TN})/\text{Sample size}$$

-> **전체 정확도**는 참조 데이터(전체에서) 중 **올바르게 분류**된 픽셀의 **비율**을 뜻함

-> (307 + 661) / 1000 = 96.8%

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

Table 7.3 Confusion matrix for a binary classification where the classes are "positive" (forest) and "negative" (non-forest)

		Actual values	
		Positive	Negative
Predicted values	Positive	307	18
	Negative	14	661

$$\text{Producer's accuracy of the Forest(Positive)class} = TP/(TP + FN)$$

$$\text{Producer's accuracy of the Non - Forest(Negative)class} = TN/(TN + FP)$$

특정 클래스의 Producer 정확도는 **"실제" 해당 클래스에** 속한 픽셀 중 **올바르게 분류된** 픽셀의 비율을 나타냄.

-> 307/(307 + 14) = 95.6%

-> 661/(661 + 18) = 97.3%

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

Table 7.3 Confusion matrix for a binary classification where the classes are "positive" (forest) and "negative" (non-forest)

		Actual values	
		Positive	Negative
Predicted values	Positive	307	18
	Negative	14	661

$$\text{User's accuracy of the Forest (Positive)class} = \text{TP}/(\text{TP} + \text{FP})$$

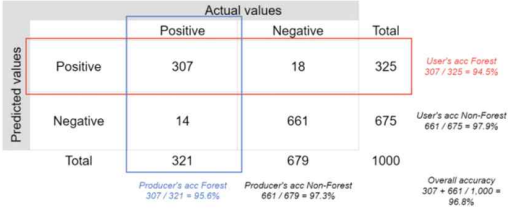
$$\text{User's accuracy of the Non - Forest(Negative)class} = \text{TN}/(\text{TN} + \text{FN})$$

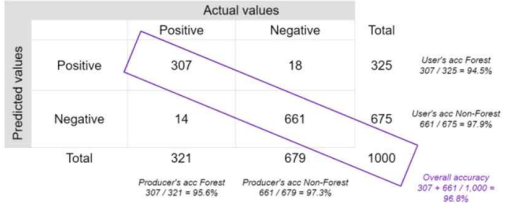
전체 예측 데이터에서 **올바른(실제)** 데이터는 얼마인가?

-> 307/(307+18) = 94.5%

-> 661/(661 + 14) = 97.9%

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix





Omission: 누락, 올바른 걸 포함시키지 못했을 때

$$\text{Omission error} = 100\% - \text{Producer's accuracy}$$

$$\text{Commission error} = 100\% - \text{User's accuracy}$$

Commission: 아닌 걸 잘못 포함시켰을 때

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

Table 7.3 Confusion matrix for a binary classification where the classes are "positive" (forest) and "negative" (non-forest)

		Actual values	
		Positive	Negative
Predicted values	Positive	307	18
	Negative	14	661

각각 첫 번째 클래스와 두 번째 클래스의 행과 열의 합계
이 값들은 각 클래스의 실제 비율과 분류된 비율을 나타냄
곱함으로써 확률을 구할 수 있음

올바르게 예측(분류)한 비율
앞서 계산한 전체 정확도와 같음

0.968

[(0.321x0.325) + (0.679x0.675)]

$$\text{Kappa Coefficient} = \frac{\text{observed accuracy} - \text{chance agreement}}{1 - \text{chance agreement}}$$

[(0.321x0.325) + (0.679x0.675)]

Kappa Coefficient: 정확도 측정 지표, 분류를 무작위와 비교
-1부터 1까지의 범위를 가짐
음수: 분류한 것이 무작위로 할당하는 것보다 못함
0: 무작위로 할당한 것과 같다
양수: 무작위로 할당한 것보다 분류를 잘함

= 0.927 (분류 잘함)

실습 1. Quantifying Classification Accuracy Through a Confusion Matrix

```
print('Confusion matrix:', confusionMatrix);

print('Overall Accuracy:', confusionMatrix.accuracy());

print('Producers Accuracy:', confusionMatrix.producersAccuracy());

print('Consumers Accuracy:', confusionMatrix.consumersAccuracy());

print('Kappa:', confusionMatrix.kappa());
```

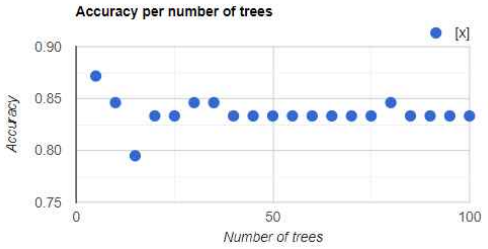
실습 2. Hyperparameter Tuning (파라미터 조정으로 성능 평가)

```
// Hyperparameter tuning.
var numTrees = ee.List.sequence(5, 100, 5);

var accuracies = numTrees.map(function(t) {
    var classifier = ee.Classifier.smileRandomForest(t)
        .train({
            features: trainingSet,
            classProperty: 'class',
            inputProperties: predictionBands
        });
    return testingSet
        .classify(classifier)
        .errorMatrix('class', 'classification')
        .accuracy();
});

print(ui.Chart.array.values({
    array: ee.Array(accuracies),
    axis: 0,
    xLabels: numTrees
}).setOptions({
    hAxis: {
        title: 'Number of trees'
    },
    vAxis: {
        title: 'Accuracy'
    },
    title: 'Accuracy per number of trees'
}));
```

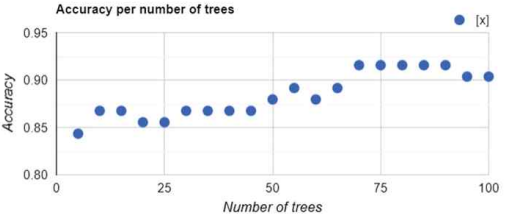
5에서 100까지 5씩 증가(5, 10, 15...)

RandomForest의 트리 수가 정확도에 얼마나 영향?

실습 2. Hyperparameter Tuning



정확도



(의사 결정) 트리 수

Spatial Autocorrelation (공간 자기상관)

7.2.3 Spatial Autocorrelation

We might also want to ensure that the samples from the training set are uncorrelated with the samples from the testing set. This might result from the spatial autocorrelation of the phenomenon being predicted. One way to exclude samples that might be correlated in this manner is to remove samples that are within some distance to any other sample. In Earth Engine, this can be accomplished with a spatial join. The following Code Checkpoint replicates Sect. 7.2.1 but with a spatial join that excludes training points that are less than 1000 m distant from testing points.

지리적으로 가까운 데이터들의 연관성을 측정하는 개념
공간 자기상관이 높다는 것은 공간적으로 인접한 지점들이 비슷한 값을 갖는다는 것을 의미함
따라서 Training 데이터셋과 test 데이터셋이 공간적으로 독립적이도록 샘플링 해야 함.
교재에서는 1000m 이내에 있는 training 데이터셋 포인트들을 test 데이터셋에서 제외함

예제 문제 풀이 (1)

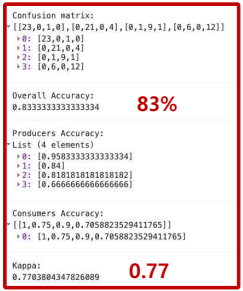
- Based on Sect. 7.2.1, test other classifiers (e.g., a Classification and Regression Tree or Support Vector Machine classifier) and compare the accuracy results with the Random Forest results. Which model performs better?

```
// 기존 Random Forest 분류기 대신 CART(Classification and Regression Tree) 분류기를 사용하여 훈련
var CARTclassifier = ee.Classifier.smileCart().train({
  features: trainingSet,
  classProperty: 'class',
  inputProperties: predictionBands
});

// 테스트 셋으로 CART 분류기의 분류를 수행하고 혼동 행렬을 얻어 모델의 정확도를 확인
var CARTconfusionMatrix = testingSet.classify(CARTclassifier).errorMatrix({
  actual: 'class',
  predicted: 'classification'
});

// 결과 출력
print('CART Confusion matrix:', CARTconfusionMatrix);
print('CART Overall Accuracy:', CARTconfusionMatrix.accuracy());
print('CART Producers Accuracy:', CARTconfusionMatrix.producersAccuracy());
print('CART Consumers Accuracy:', CARTconfusionMatrix.consumersAccuracy());
print('CART Kappa:', CARTconfusionMatrix.kappa());
```

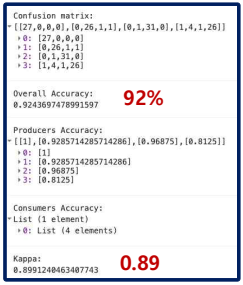
```
CART Confusion matrix:                                          JSON
▾ [[23,0,1,0],[0,19,0,6],[0,1,9,1],[0,6,0,12]]                  JSON
  ▸ 0: [23,0,1,0]
  ▸ 1: [0,19,0,6]
  ▸ 2: [0,1,9,1]
  ▸ 3: [0,6,0,12]

CART Overall Accuracy:                                          JSON
0.8076923076923077        80%

CART Producers Accuracy:                                        JSON
▾ List (4 elements)                                             JSON
  ▸ 0: [0.9583333333333334]
  ▸ 1: [0.76]
  ▸ 2: [0.8181818181818182]
  ▸ 3: [0.6666666666666666]

CART Consumers Accuracy:                                        JSON
▾ [[1,0.7307692307692307,0.9,0.631578947368421]]                JSON
  ▸ 0: [1,0.7307692307692307,0.9,0.631578947368421]

CART Kappa:                                                     JSON
0.7358916478555305        0.73
```

예제 문제 풀이 (2)

- Try setting a different seed in the randomColumn method and see how that affects the accuracy results. You can also change the split between the training and testing sets (e.g., 70/30 or 60/40).




```
// 시드값 변경
var trainingTesting = data.randomColumn('random', 123);

// 데이터셋을 비율 변경 (70/30)
var trainingSet = trainingTesting.filter(ee.Filter.lessThan('random', 0.7));
var testingSet = trainingTesting.filter(ee.Filter.greaterThanOrEquals('random', 0.7));
```



감사합니다