

- 1. Weka-ko Java liburutegia
- 2. Datuak kargatu
- 3. Instantziei buruzko informazioa
- 4. Ebaluazio eskemak
  - 4.1. Ez-zintoa
  - 4.2. Hold-out
    - 4.2.1. Simplea
    - 4.2.2. Stratified hold-out
    - 4.2.3. Repeated hold-out
  - 4.3. k-fold Cross Validation
- 5. Ebaluazio metrikak
- 6. Iragarpenak (predictions)
- 7. Exportazioa eta inportazioa
- 8. Beste batzuk

## 1. Weka-ko Java liburutegia

---

`weka.jar` fitxategia importatu behar da gure proiektuan. Horretarako, Weka instalatu den direktorioan aurkituko dugu.

[weka-wiki](#)

[JavaDoc](#)

## 2. Datuak kargatu

---

`.arff` fitxategi batetik:

```
DataSource source = new DataSource("path/to/arff");
Instances data = source.getDataSet();
```

Klase zein atributu indizean doan adierazi. Normalean, azken atributua izango da klasea:

```
data.setClassIndex(data.numAttributes()-1);
```

Defekutz, azken atributa hartzen da baina hobeto `Instances` motako instnatzia bat sortzean zein atributuan doan klase adierazten badugu.

## 3. Instantziei buruzko informazioa

---

Instantzia kopurua:

```
data.numInstances();
```

Instantzien atributu kopurua:

```
data.numAttributes();
```

Atributu batek har ditzakeen balio kopurua:

```
data.numDistinctValues(data.attribute(<index>));
```

Atributu batek dituen missing value kopurua:

```
data.attributeStats(<index>).missingCount;
```

Klase minoritarioa, adibidea:

```
int[] maiztasunak = data.attributeStats(data.numAttributes()-1).nominalCounts;
int i = 0;
int minMaiz = maiztasunak[i], minMaizPos = i;
while(i < maiztasunak.length){
    if (maiztasunak[i]<minMaiz){
        minMaiz = maiztasunak[i];
        minMaizPos = i;
    }
    i++;
}
System.out.println("Klase minoritarioa " + data.classAttribute().value(minMaizPos)
+ " da.\n");
```

Atributu batek har ditzakeen balioak (identifikatzailea) eta haien maiztasunak, klase atributua adibidez:

```
Enumeration<Object> balioIzenak =
data.attribute(data.numAttributes()-1).enumerateValues();
int[] maiztasunak = data.attributeStats(data.numAttributes()-1).nominalCounts;
int i = 0;
while (balioIzenak.hasMoreElements()){
    System.out.println("\t" + balioIzenak.nextElement() + " --> " +
maiztasunak[i++]);
}
```

Atributuen info gehiago kontsultatu: [weka.core.AttributeStats](https://weka.core.AttributeStats)

## 4. Ebaluazio eskemak

---

Ebaluazio eskemak ikusteko, [datuak kargatuta](#) ditugula [data](#) aldagaian.

### 4.1. Ez-zintoa

Ez dira [train\\_instances](#) eta [test\\_instances](#) bereizten.

```
NaiveBayes estimador = new NaiveBayes();
estimador.buildClassifier(data);

Evaluation evaluator = new Evaluation(data);
evaluator.evaluateModel(estimador, data);
```

### 4.2. Hold-out

Hold-out-ean, [data](#) multzotik [train\\_instance](#) eta [test\\_instances](#) bereizten dira.

Edozein moduan egiten badagu ere, instantzia sorta nahastu (randomize) egin behar dugu:

```
Randomize filter_random = new Randomize();
filter_random.setRandomSeed(42); // seed edo hazia
filter_random.setInputFormat(data);
Instances random_data = Filter.useFilter(data, filter_random);
```

**Kontuz!!** [setInputFormat](#) metodoa, beste set guztiak egin eta gero egin behar da. Edozein filtro erabiltzean berdina gertatuko da.

#### 4.2.1. Siplea

[random\\_data](#) multzotik, [train\\_instances](#) eta [test\\_instances](#) lortu, adibidez train %66:

```
RemovePercentage filter = new RemovePercentage();
filter.setPercentage(66);
filter.setInputFormat(random_data);

Instances test_instances = Filter.useFilter(random_data, filter);

filter.setInvertSelection(true); //eta ez setPercentage(36);
filter.setInputFormat(random_data);
Instances train_instances = Filter.useFilter(random_data, filter);
```

Evaluatu aurreko [train\\_instances](#) eta [test\\_instances](#) erabiliz, adibidez [NaiveBayes](#) sailkatzailea:

```
NaiveBayes estimador = new NaiveBayes();
estimador.buildClassifier(train_instances);

Evaluation evaluator = new Evaluation(train_instances);
evaluator.evaluateModel(estimador, test_instances);
```

#### 4.2.2. Stratified hold-out

`random_data` multzotik, `train_instances` eta `test_instances` aterako dugu baina bi azpimultzo haietan, klasearen balioen proportzioa mantenduko da. Adibidez, train %80 eta test %20:

```
Resample filter_resample = new Resample();
filter_resample.setSampleSizePercent(80);
filter_resample.setNoReplacement(true);
filter_resample.setInputFormat(random_data);
Instances train_data = Filter.useFilter(random_data, filter_resample);

filter_resample.setInvertSelection(true);
filter_resample.setNoReplacement(true);
filter_resample.setInputFormat(random_data);
Instances test_data = Filter.useFilter(random_data, filter_resample);
```

Evaluatu aurreko `train_instances` eta `test_instances` erabiliz, adibidez `NaiveBayes` sailkatzailea:

```
NaiveBayes estimador = new NaiveBayes();
estimador.buildClassifier(train_instances);

Evaluation evaluator = new Evaluation(train_instances);
evaluator.evaluateModel(estimador, test_instances);
```

#### 4.2.3. Repeated hold-out

Hold-out hainbat aldiz errepikatuko da `train_instances` eta `test_instances` zatiak desberdinak izanik. Adibidea, non aurretik kalkulaturako klase minoritarioaren recall kalkulatu den:

```
double[] recall_values = new double[50];
for (int j = 0; j < 50; j++) {
    // 1. RANDOMIZE
    Randomize filter_random = new Randomize();
    filter_random.setSeed(j); // partiketak desberdinak izateko, i-ren balioa
    erabiliko dugu
    filter_random.setInputFormat(data);
    // random zenbaki bat hautatzeko
    Instances random_data = Filter.useFilter(data, filter_random);
```

```
// 2. REMOVE PERCENTAGE
RemovePercentage filter = new RemovePercentage();
filter.setPercentage(66);
filter.setInputFormat(random_data);

Instances test_instances = Filter.useFilter(random_data, filter);

filter.setInvertSelection(true); //eta ez setPercentage(36);
filter.setInputFormat(random_data);
Instances train_instances = Filter.useFilter(random_data, filter);

// 3. EVALUATU
NaiveBayes estimador = new NaiveBayes(); //sailkatzailea (classifier)
estimador.buildClassifier(train_instances); //hau erabili behar da, ez delako
k-fCV

Evaluation evaluator = new Evaluation(train_instances);
evaluator.evaluateModel(estimador, test_instances);

// 4. RECALL GORDE
recall_values[j] = evaluator.recall(minMaizPos);
}
```

### 4.3. k-fold Cross Validation

```
NaiveBayes estimador = new NaiveBayes(); //sailkatzaile
Evaluation evaluator = new Evaluation(data); // "data" formatua jakiteko
evaluator.crossValidateModel(estimador, data, <k>, new Random(1));
```

## 5. Ebaluazio metrikak

Kontuan izanda `evaluator` aldagaia erabili dela ebaluazioa atalean [Ikusi 4. atala](#)

Ebaluatzailearen zenbakiak ikusteko:

```
evaluator.toSummaryString() // laburpena

double accuracy = evaluator.pctCorrect();
```

Nahasmen matrizea (bi aukera, `String` edo `double[]`):

```
String confMat = evaluator.toMatrixString();
double[] confMat = evaluator.confusionMatrix();
```

Klase balioei buruzko metriakak:

```
double precision = evaluator.precision(<class_value_index>);
double recall = evaluator.recall(<class_value_index>);
double fMeasure = evaluator.fMeasure(<class_value_index>);
```

## 6. Iragarpenak (predictions)

---

Instantzia bakoitzeko klasearen balioa iragarri. `eredua` entrenatutako `NaiveBayes` instantzia bat izanik:

```
for (int i = 0; i < data.numInstances(); i++){
    double pred = eredua.classifyInstance(data.instance(i));
    System.out.println(data.classAttribute().value((int) pred));
}
```

## 7. Exportazioa eta inportazioa

---

Instances klase bat exportatu. Adibidez, `train_data` multzoa:

```
ArffSaver saver = new ArffSaver();
saver.setInstances(train_data);
saver.setFile(new File("path"));
saver.writeBatch();
```

`.model` fitxategia exportatu, adibidez `sailkatzailea` NaiveBayes-ren instantzia bat izanik:

```
SerializationHelper.write("path", sailkatzailea);
```

**Kontuz!!** Gogoratu `.model` bat gordetzen denean, datu guztiekin entrenatuta gorde behar dela, ez bakarrik estimazioak egin den `.model`-a.

`.model` fitxategia inportartzeko, adibidez NaiveBayes sailkatzaile bat:

```
NaiveBayes eredua = (NaiveBayes) SerializationHelper.read(args[0]);
```

## 8. Beste batzuk

---

`jar` bat egikaritu:

```
java -jar <pathToJar> <arg0> <arg1> ... <argN>
```

`InnaccessibleObjectException` ez emateko `jar`-a exekutatzekoan erabili behar den aukera:

```
java -jar --add-opens java.base/java.lang=ALL-UNNAMED <pathToJar> <arg0> <arg1>  
... <argN>
```