Introduction to Data Mining

Lab 2: Evaluation

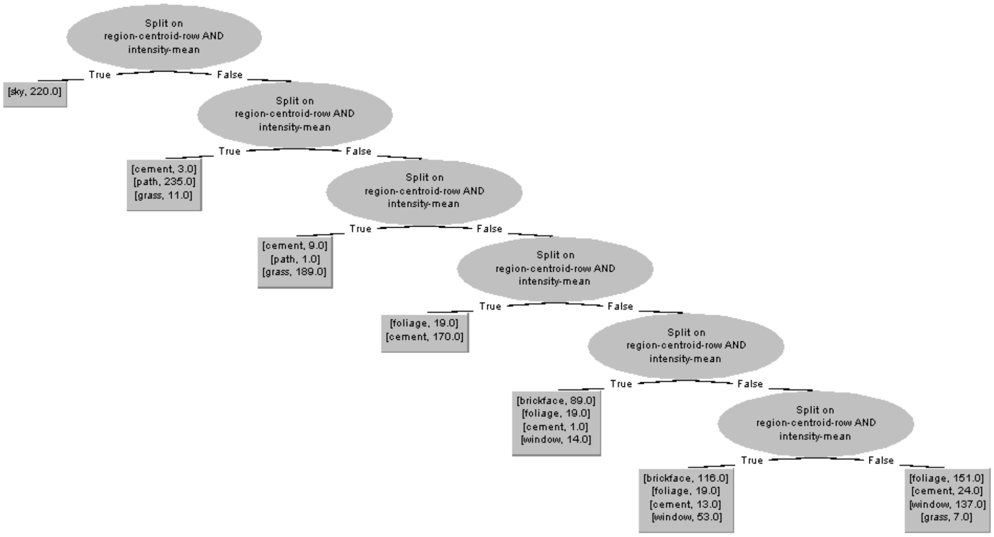
Name: Phạm Đức Đạt
ID Student: ITITIU20184

2.1. Be a classifier

In the second class, we are going to learn how to use datasets to evaluate data mining algorithms in Weka. (See the lecture of class 2 by Ian H. Witten, [1][1])

Interactive decision tree construction

- ➔ Follow the instruction in [1] to see how decision trees are created for different combinations of attributes in a dataset. Firstly, a dataset and a training set are selected. Secondly, we choose and start running UserClassifier to see a decision tree in the Tree Visualizer. In the Data Visualizer, thirdly, the attributes to use for X and Y are selected, we then select instances in a region in the graph and submit. At this point, the Tree Visualizer shows the tree.
- ➔ Examine segment-challenge dataset to draw a decision tree for the following pair of attributes by selecting and submitting classes one by one, then remark how many instances are predicted correctly.

Attributes	Split on *Region-centroid-row* and *intensity-mean*
Decision Tree	

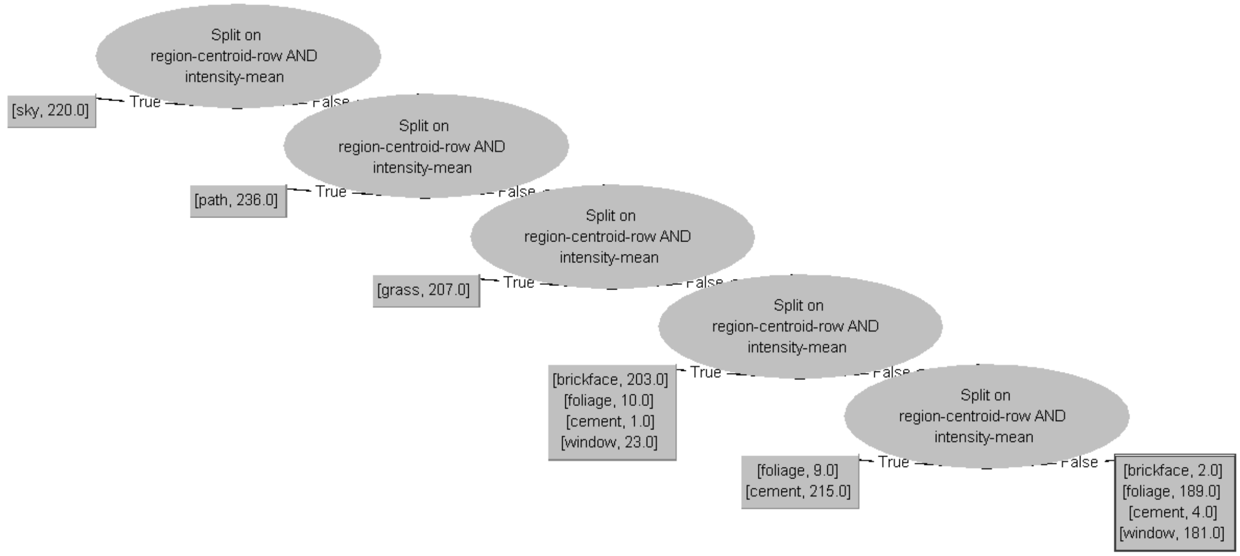
Split on region-centroid-row AND intensity-mean
True
False
[sky, 220.0]
Split on region-centroid-row AND intensity-mean
True
False
[cement, 3.0] [path, 235.0] [grass, 11.0]
Split on region-centroid-row AND intensity-mean
True
False
[cement, 9.0] [path, 1.0] [grass, 189.0]
Split on region-centroid-row AND intensity-mean
True
False
[foliage, 19.0] [cement, 170.0]
Split on region-centroid-row AND intensity-mean
True
False
[brickface, 89.0] [foliage, 19.0] [cement, 1.0] [window, 14.0]
Split on region-centroid-row AND intensity-mean
True
False
[brickface, 116.0] [foliage, 19.0] [cement, 13.0] [window, 53.0]
[foliage, 151.0] [cement, 24.0] [window, 137.0] [grass, 7.0]

[1] http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/

Remark	That tree has got 77.037% correctly.

Build a tree, what strategy do you use? → bottom-up covering strategy

Can you build a “perfect” tree?



This tree has got 79.5062% correctly.

2.2. Training and testing

The lecture of evaluation (see [1]-2.2)

Follow the instructions in [1]-2.3: use **J48** to analyze *segment* dataset, and write down how accuracy it can achieve with different seeds. (If a random number seed is provided, the dataset will be shuffled before the subset is extracted.)

Random number seeds	**Percent accuracy (x)**	**Random number seeds**	**Percent accuracy (x)**
1	96.6667%	6	96.6667%
2	94%	7	92%
3	94%	8	94%
4	96.6667%	9	93.3333%
5	95.3333%	10	94.6667%
Evaluation	*Sample Mean*	94.73334%	
	Standard deviation	1.62	

Note:

Sample mean $\bar{x} = \frac{\sum x_i}{n}$

Variance $\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$

Standard deviation σ

Remark? - The real performance of J48 on the segment-challenge dataset is approximately 95% accuracy, plus or minus approximately 2%.

So, we can say that performance of J48 on the segment-challenge dataset is between 93-97% accuracy.

2.3. Baseline accuracy

Follow the instructions in [1]-2.4 to run some classifiers for *diabetes* dataset:

Classifier	**Accuracy**
J48	76.2452%
NaiveBayes	77.0115%
IBk	72.7969%
PART	74.3295%
ZeroR	65.1042 %

What is Baseline accuracy? – It is approximately 65%.

For *supermarket* dataset

Classifier	**Accuracy**
ZeroR	63.713%
J48	62.6828%

NaiveBayes	62.6828%
IBk	38.2708%
PART	62.6828%

Why do the classifiers achive lower accuracy? – Because for supermarket dataset, the attributes are not really informative.

2.4. Cross-validation

The *holdout* procedure: a certain amount is held over for testing and the remainder used for training.

Stratification: each class is properly represented in both training and test sets.

The *repeated holdout* method of error rate estimation: In each iteration a certain proportion, say two-thirds, of the data is randomly selected for training (using different random-number seeds), possibly with *stratification*, and the remainder is used for testing. The error rates on the different iterations are averaged to yield an overall error rate.

The lecture of cross validation, 10-fold cross-validation, stratified cross-validation (see [1]-2.5).

In *cross-validation*, you decide on a fixed number of folds, or partitions, of the data. Suppose we use three. Then the data is split into three approximately equal partitions; each in turn is used for testing and the remainder is used for training. That is, use two-thirds of the data for training and one-third for testing, and repeat the procedure three times so that in the end, every instance has been used exactly once for testing. This is called *three-fold cross-validation*, and if stratification is adopted as well—which it often is—it is *stratified three-fold cross-validation*.

Weka does stratified cross-validation by default.

Follow the instructions in [1]-2.5, and examine **J48** on *Diabetes* dataset.

Holdout (10%)	**Percent accuracy (x)**	**10-fold cross-validation**	**Percent accuracy (x)**
Random seed: 1	75.3	Random seed: 1	73.8
-- 2	77.9	-- 2	75.0
-- 3	80.5	-- 3	75.5
-- 4	74.0	-- 4	75.5
-- 5	71.4	-- 5	74.4
-- 6	70.1	-- 6	75.6
-- 7	79.2	-- 7	73.6
-- 8	71.4	-- 8	74.0
-- 9	80.5	-- 9	74.5
-- 10	67.5	-- 10	73.0
Sample Mean	*74.8*	*Sample Mean*	**74.5**
Standard deviation	*4.6*	*Standard deviation*	**0.9**

Examine **PART** on *Diabetes* dataset:

Holdout (10%)	**Percent accuracy (x)**	**10-fold cross-validation**	**Percent accuracy (x)**
Random seed: 1	75.3	Random seed: 1	75.3
-- 2	75.3	-- 2	73.0
-- 3	71.4	-- 3	72.8
-- 4	72.7	-- 4	74.9
-- 5	77.9	-- 5	74.2
-- 6	71.4	-- 6	73.0
-- 7	74.0	-- 7	73.4
-- 8	68.8	-- 8	71.9
-- 9	75.3	-- 9	74.6
-- 10	66.2	-- 10	71.4
Sample Mean	*72.8*	*Sample Mean*	**67.0**
Standard deviation	*3.5*	*Standard deviation*	**7.0**

Examine **NaiveBayes** on *Diabetes* dataset:

Holdout (10%)	**Percent accuracy (x)**	**10-fold cross-validation**	**Percent accuracy (x)**
Random seed: 1	77.9	Random seed: 1	76.3
-- 2	75.3	-- 2	75.3
-- 3	72.7	-- 3	76.2
-- 4	68.8	-- 4	75.5
-- 5	80.5	-- 5	75.1
-- 6	76.6	-- 6	75.8
-- 7	76.6	-- 7	76.2
-- 8	74.0	-- 8	75.3
-- 9	76.6	-- 9	76.0
-- 10	71.4	-- 10	75.9
Sample Mean	*75.0*	*Sample Mean*	**75.8**
Standard deviation	*3.4*	*Standard deviation*	**0.4**