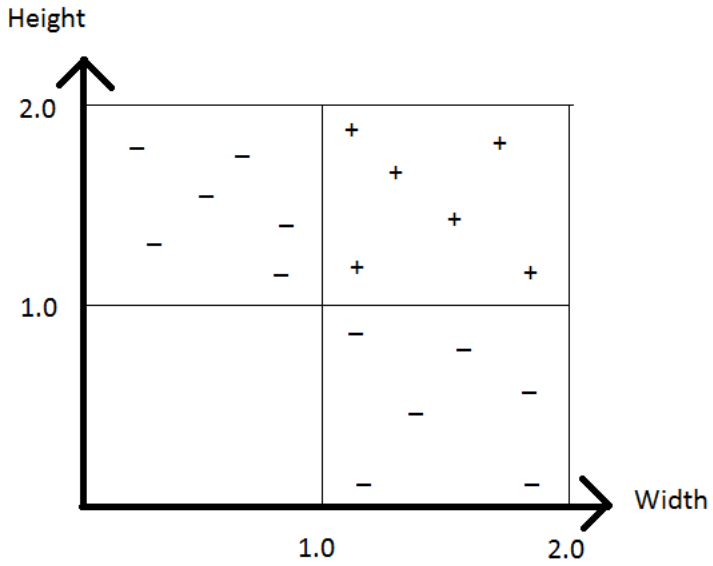
CS 495 – Assignment 7

Go to http://euclid.nmu.edu/~mkowalcz/cs495/notes/datasets/rectangles and download the datasets goodRectangles.arff and badRectangles.arff.

The structure of goodRectangles.arff is depicted here. There are 100 Yes (+) instances and 200 No (-) instances. Note that for the "No" instances, this dataset strongly violates the independence assumption of Naive Bayes. There are 100 instances in each "square" below.



Run goodRectangles.arff using NaiveBayes in Weka (using 10-fold cross validation on the entire dataset). What is the accuracy?

Supervised discretization takes into account the target feature when deciding where to create the discrete bins for continuous features. Select supervised discretization and run it again. What is the accuracy now? Is this result surprising to you?

Now run Trees →J48 (this is a refined version of the ID3 decision tree algorithm). How does the accuracy compare to Naive Bayes?

Assume that Weka made two bins for each descriptive feature: one for the range [0,1) and one for the range [1,2]. Write out the Naive Bayes calculations for the following instances, given goodRectangles.arff as the training data:

Instance: (1.5, 1.5, ?)...

```
Pr[big = Yes] * Pr[1 ≤ width ≤ 2 | big = "Yes"] * Pr[1 ≤ height ≤ 2 | big = "Yes"] =

Pr[big = "No"] * Pr[1 ≤ width ≤ 2 | big = "No"] * Pr[1 ≤ height ≤ 2 | big = "No"] =
```

So it would classify the instance (1.5, 1,5, ?) as:

Now predict instance: (0.5, 1.5, ?)…

```
Pr[big = "Yes"] * Pr[0 ≤ width < 1 | big = "Yes"] * Pr[1 ≤ height ≤ 2 | big = "Yes"] =

Pr[big = "No"] * Pr[0 ≤ width ≤ 1 | big = "No"] * Pr[1 ≤ height ≤ 2 | big = "No"] =
```

So it would classify the instance (0.5, 1.5, ?) as:

Now try badRectangles.arff. This is the exact same dataset except there are only 40 "Yes" (+) instances instead of 100. What is the accuracy for naive bayes and J48 this time? Which is better?

Write out the Naive Bayes calculations for the following instances, given badRectangles.arff as the training data.

Instance: (1.5, 1.5, ?)…

```
Pr[big = "Yes"] * Pr[1 ≤ width ≤ 2 | big = "Yes"] * Pr[1 ≤ height ≤ 2 | big = "Yes"] =

Pr[big = "No"] * Pr[1 ≤ width ≤ 2 | big = "No"] * Pr[1 ≤ height ≤ 2 | big = "No"] =
```

So it would classify the instance (1.5, 1.5, ?) as:

Instance: (0.5, 1.5, ?)…

```
Pr[big = "Yes"] * Pr[0 ≤ width ≤ 1 | big = "Yes"] * Pr[1 ≤ height ≤ 2 | big = "Yes"] =

Pr[big = "No"] * Pr[0 ≤ width ≤ 1 | big = "No"] * Pr[1 ≤ height ≤ 2 | big = "No"] =
```

So it would classify the instance (1.5, 1.5, ?) as:

At this point Naïve Bayes is making the same predictions as ZeroR, which just predicts the median of the whole dataset all the time (all "no" predictions). Why did the lower number of "Yes" instances cause Naïve Bayes to perform so poorly on this dataset?

Go to http://euclid.nmu.edu/~mkowalcz/cs495/notes/datasets/kungPeople and download the dataset population.arff.

Run NaiveBayes on population.arff with supervised discretization still in effect. What is the accuracy?

Remove features r1 and r2 (these features are just random numbers added into the dataset and have no connection with any other feature). What is the accuracy now?

Why do the random features have no significant effect on the accuracy? Explain in terms of how Naive Bayes does its probability calculations.