COSC2110/COSC2111 Data Mining
Tutorial Solutions Week 11

---

1. Using J48 as the base classifier show the operation of bagging on the following data. Assume there will be 3 classifiers in the ensemble.

Run	Supervisor	Operator	Machine	Overtime	output
1	Patrick	Joe	a	no	high
2	Patrick	Sam	b	yes	low
3	Thomas	Jim	b	no	low
4	Patrick	Jim	b	no	high
5	Sally	Joe	c	no	high
6	Thomas	Sam	c	no	low
7	Thomas	Joe	c	no	low
8	Patrick	Jim	a	yes	low
9	Patrick	Sam	c	no	???

Step 1.

Build three classifiers with the following randomly sampled training sets,

	Train Set
Decision Tree 1	1,2,3,4,5,6,8,8
Decision Tree 2	1,1,2,2,2,6,7,8
Decision Tree 3	1,2,3,4,6,7,8,3

Note that some instances have not been used while some have been repeated. This is what happens with bootstrap sampling. The unused instances go to the test set.

Step 2.

Create the first decision tree (Decision Tree 1) by calculating the ***Information Ratio*** of each attribute and choose the one with highest Information Ratio as the root node.

Information gain is the method of splitting that the J48 algorithm uses. It is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree using J48 is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).

Note from Lecture 3: *disorder = entropy = information*

Calculation of the Information Ratio for the *Supervisor attribute (based on the training set of instances 1,2,3,4,5,6,8,8):*

		Output	
Attribute Value	**# of instances**	**Low**	**High**
Patrick	5	3	2
Thomas	2	2	0
Sally	1	0	1
Total	**8**	**5**	**3**

$$Info(Output) = Info\left(\frac{5}{8}, \frac{3}{8}\right)$$

$$= -\left(\frac{5}{8} log_2 \frac{5}{8}\right) - \left(\frac{3}{8} log_2 \frac{3}{8}\right)$$

$$= 0.424 + 0.531$$

$$= 0.955$$

$$Info(Supervisor) = \frac{5}{8} Info(Patrick) + \frac{2}{8} Info(Thomas) + \frac{1}{8} Info(Sally)$$

$$= \frac{5}{8}\left(-\left(\frac{3}{5} log_2 \frac{3}{5}\right) - \left(\frac{2}{5} log_2 \frac{2}{5}\right)\right) + \frac{2}{8}\left(-\left(\frac{2}{2} log_2 \frac{2}{2}\right) - \left(\frac{0}{2} log_2 \frac{0}{2}\right)\right) + \frac{1}{8}\left(-\left(\frac{0}{1} log_2 \frac{0}{1}\right) - \left(\frac{1}{1} log_2 \frac{1}{1}\right)\right)$$

$$= \frac{5}{8}\big(-(-0.442) - (-0.529)\big) + \frac{2}{8}\big((0) - (0)\big) + \frac{1}{8}\big((0) - (0)\big)$$

$$= \frac{5}{8}(0.442 + 0.529) + 0 + 0$$

$$= 0.607$$

$$Information\ Gain(Supervisor) = Info(Output) - Info(Supervisor)$$

$$= 0.955 - 0.607$$

$$= 0.348$$

$$Information\ Ratio(Supervisor) = \frac{Information\ Gain(Supervisor)}{Info(Supervisor)}$$

$$= \frac{0.348}{0.607}$$

$$= 0.573$$

The Information-ratios for the all attributes are:

Supervisor	0.573
Operator	0.144
Machine	0.017
Overtime	0.358

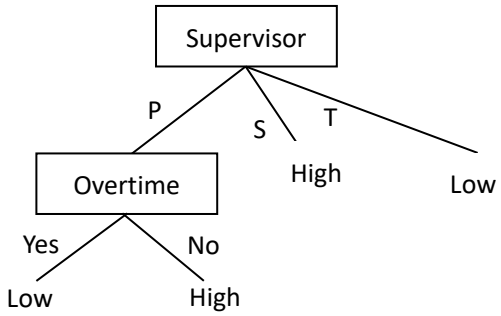
We choose the attribute with highest Info-Ratio as the root node. With the train set {1,2,3,4,5,6,8,8}, the best attribute is *Supervisor* with 0.573. To complete the tree, do this recursively until every leaf node holds only instances from one class.

Step 3.

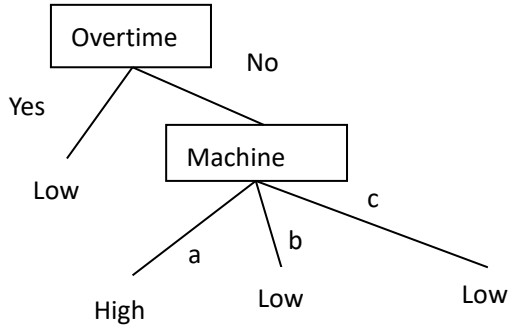
Following the same process, we can induce the trees for the other two training sets (Decision Tree 2: {1,1,2,2,2,6,7,8}, and Decision Tree 3: {1,2,3,4,6,7,8,3}.

Now we get the following trees (The trees show below are just examples for the purpose of demonstrating bagging only and not based on actual calculations),

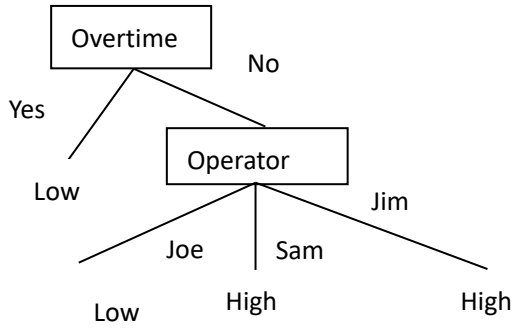
Decision Tree 1



Decision Tree 2



Decision Tree 3



Step 4.

Now if given a test instance,

Run	Supervisor	Operator	Machine	Overtime	output
9	Patrick	Sam	c	no	???

Decision Tree 1 will classify it as: **Low**

Decision Tree 2 will classify it as: **Low**

Decision Tree 3 will classify it as: **High**.

With bagging, the majority wins: We classify the instance as **Low**.

2. ***Show the operation of Adaboost for k = 2 on the below data.***

Run	Supervisor	Operator	Machine	Overtime	output
1	Patrick	Joe	a	no	high
2	Patrick	Sam	b	yes	low
3	Thomas	Jim	b	no	low
4	Patrick	Jim	b	no	high
5	Sally	Joe	c	no	high
6	Thomas	Sam	c	no	low
7	Thomas	Joe	c	no	low
8	Patrick	Jim	a	yes	low
9	Patrick	Sam	c	no	???

Step 1.

Assign weights to the data instances.

Since we have 8 instances, the initial weights will be:

Instance	1	2	3	4	5	6	7	8
Initial Weight	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
	=0.125	=0.125	=0.125	=0.125	=0.125	=0.125	=0.125	=0.125

Step 2.

Choose a "weak learner" classifier that will be "boosted" to become a "strong learner" classifier. A weak learner is an algorithm (eg. A classifier such as ZeroR or OneR) that performs relatively poorly, with an accuracy that is only just better than random chance.

For this example we will use **OneR**.

Step 3.

Create a randomly sampled training set with replacement. Just like with Bagging, we create a sample training dataset D_1 with replacement, for example: $D_1 = \{1,2,3,4,5,6,8,8\}$

Step 4.

Train a model with your weak learner and calculate the error of the classifier on the training set. Using **OneR** with the dataset D_1 we train a model:

a) Create frequency tables for each attribute and calculate the potential classification error if the attribute was used to classify the dataset $\boldsymbol{D_1}$.

Supervisor	**Output**	
	Low	**High**
Patrick	3	2
Thomas	2	0

Sally	0	1

Low has higher frequency, therefore:

$$error(Supervisor) = \frac{(2+0+1)}{8} = \frac{3}{8} = 37.5\%$$

Operator	**Output**	
	Low	**High**
Joe	0	2
Sam	2	0
Jim	3	1

Low has higher frequency, therefore:

$$error(Operator) = \frac{(2+0+1)}{8} = \frac{3}{8} = 37.5\%$$

Machine	**Output**	
	Low	**High**
a	2	1
b	2	1
c	1	1

Low has higher frequency, therefore:

$$error(Machine) = \frac{(1+1+1)}{8} = \frac{3}{8} = 37.5\%$$

Overtime	**Output**	
	Low	**High**
Yes	3	0
No	2	3

Low has higher frequency, therefore:

$$error(Overtime) = \frac{(0+1)}{8} = \frac{3}{8} = 37.5\%$$

Using OneR and a model trained with dataset D_1, each attribute has the same error (37.5%), therefore we use the attribute with the least number of attribute types. For this data that attribute is "Overtime". We then generate a OneR classifier ($\boldsymbol{C_1}$)that uses "Overtime" as the predictive attribute. **The error of our first weak learner $\boldsymbol{C_1}$, is therefore 37.5%.**

Step 5.

Adjust instance weights and normalise.

With C_1, when *overtime=yes then output is low, when overtime=no then output is also low*. This **correctly classifies instances** $\{\mathbf{2}, \mathbf{3}, \mathbf{6}, \mathbf{8}\}$ but incorrectly classifies $\{1,4,5,7\}$.

To adjust our initial instance weights, we multiply the **correctly classified instance** weights by the ratio:

$\frac{\mathbf{0.375}}{\mathbf{1-0.375}} = \frac{\mathbf{3}}{\mathbf{5}}$ where 0.375 is our classification error (ie. 37.5%).

The adjusted weights are now:

Instance	1	2	3	4	5	6	7	8
Adjusted Weight	$\frac{1}{8}$	$\frac{1}{8}\times\frac{3}{5}$	$\frac{1}{8}\times\frac{3}{5}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}\times\frac{3}{5}$	$\frac{1}{8}$	$\frac{1}{8}\times\frac{3}{5}$
	=0.125	=0.075	=0.075	=0.125	=0.125	=0.075	=0.125	=0.075

Normalise the weights using the formula: $weight = weight \times \frac{\sum weight(old)}{\sum weight(new)}$

The new normalised instance weights are now:

Instance	1	2	3	4	5	6	7	8
New Weight	$0.125\times\frac{1}{0.8}$	$0.075\times\frac{1}{0.8}$	$0.075\times\frac{1}{0.8}$	$0.125\times\frac{1}{0.8}$	$0.125\times\frac{1}{0.8}$	$0.075\times\frac{1}{0.8}$	$0.125\times\frac{1}{0.8}$	$0.075\times\frac{1}{0.8}$
	0.156	0.094	0.094	0.156	0.156	0.094	0.156	0.094

Step 6.

Resample the data using the new weights from Step 5.

Use a method such as "weighted random sampling" or "rejection sampling" where the weight represents a probability of being selected for a new training dataset D_2. Resampling the data with the new adjusted and normalised weights will result in a higher "confidence" to leave the correctly classified instances alone in the next iteration.

Since the incorrectly classified instances now have a higher weight, they are less likely to get selected in the next iteration when sampling the new dataset. Let's assume that we get $D_2 = \{1,1,4,4,5,5,7,8\}$. The sampling process, albeit taking weights into consideration, involves some randomness. The low-weight instances still might appear.

Step 7.

Using **OneR** with the dataset $D_2 = \{1,1,4,4,5,5,7,8\}$, we train a new model ($\boldsymbol{C_2}$):

Create frequency tables for each attribute and calculate the potential classification error if the attribute was used to classify the dataset $\boldsymbol{D_2}$.

Note that we bring the weight of the instance into consideration in this step. For example, for the attribute 'Supervisor', instance 8 has only $^3/_5$ of the weight of other instances and is taken into account when we calculate the error for this attribute.

Supervisor	**Output**	
	Low	**High**
Patrick	1 x $^3/_5$	4
Thomas	1	0
Sally	0	2

High has the greater frequency, therefore:

$$error(Supervisor) = \frac{(0.6 + 1 + 0)}{8} = \frac{1.6}{8} = 20\%$$

Operator	Output	
	Low	High
Joe	1	4
Sam	0	0
Jim	1 x $^3/_5$	2

High has higher frequency, therefore:

$$error(Operator) = \frac{(1 + 0 + 0.6)}{8} = \frac{1.6}{8} = 20\%$$

Machine	Output	
	Low	High
a	1 x $^3/_5$	2
b	0	2
c	1	2

High has higher frequency, therefore:

$$error(Machine) = \frac{(0.6 + 0 \_ 1)}{8} = \frac{1.6}{8} = 20\%$$

Overtime	Output	
	Low	High
Yes	1 x $^3/_5$	0
No	1	6

High has higher frequency, therefore:

$$error(Overtime) = \frac{(0.6 + 1)}{8} = \frac{1.6}{8} = 20\%$$

With C_2, when *overtime=no then output is high, when overtime=yes then output is also high*. This **correctly classifies instances** $\{\mathbf{1}, \mathbf{4}, \mathbf{5}\}$ but incorrectly classifies $\{7, 8\}$, and **error of 20%** (remember D_2, has 2 instances of '1', '4', and '5').

We stop creating new models as the question specified "k = 2" and we have already got 2 classification models. If we were required to complete further iterations we would then adjust and normalise the weights from this iteration using the methods described in **Step 5**, then select a new sample dataset as described in **Step 6** and repeat this step, **Step 7**.

Step 8.

We now need to calculate the weight of the classification models used before we can use them for classification on unseen data.

$\boldsymbol{C_1}$ has an error rate of 37.5% so its weight is $-log\left(\frac{0.375}{1-0.375}\right) \approx 0.737$

C_2 has an error rate of 20% so its weight is $-log\left(\frac{0.2}{1-0.2}\right) = 2$

Step 9.

Now classify the given a test instance,

Run	Supervisor	Operator	Machine	Overtime	output
9	Patrick	Sam	c	no	???

Now let's do the classification:

C_1 would classify the new instance as **low**, while C_2 would classify it as **high**.

We now sum the weights of all classifiers which classified our unseen example as low and sum the weights of those that classified it as high.

As we only have two classifiers the sum of weights for the **low classifiers = 0.737** and the sum for the **high classifiers = 2**. As the high classifiers have the greater sum of weights, we classify the unseen instance as having an output of **high**.

---

3. ***A random forest classifier uses decision trees in which the split attribute is chosen at random. Show the how a random forest of 3 trees could be generated and used on the above data.***

The only difference between this question and question 1 is that in question 1 to build decision trees, we need to calculate entropy (disorder) or info gain to choose the split attribute. While in random forest, split attributes are selected randomly.

---

4. ***Boosting requires a classifier in which the instances can be weighted. How can this be achieved without modifying the base classifier to use weighted instances?***

Make multiple copies of instances to reflect their intended weights.