School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies (Semester 1, 2019)
Workshop exercises: Week 11

Given the following dataset:

ID	*Outl*	*Temp*	*Humi*	*Wind*	PLAY
TRAINING INSTANCES					
A	s	h	h	F	N
B	s	h	h	T	N
C	o	h	h	F	Y
D	r	m	h	F	Y
E	r	c	n	F	Y
F	r	c	n	T	N
TEST INSTANCES					
G	o	c	n	T	?
H	s	m	h	F	?

1. Find the entropy of (the distribution of the attribute values) for each of the six attributes, given this probabilistic model.

   - Entropy is a measure of unpredictability — the information required to predict an event. Higher entropy means less predictability.
   - The entropy of a discrete random variable X with possible states $x_1, .., x_n$ is:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

where $0 \log_2 0 \stackrel{\text{def}}{=} 0$

   - Here, we are going to sum entropy over attribute values. For example: *Outl* with the values of s, o, and r:

$$\begin{aligned} H(Outl) &= -[p(\mathtt{s}) \log_2 p(\mathtt{s}) + p(\mathtt{o}) \log_2 p(\mathtt{o}) + p(\mathtt{r}) \log_2 p(\mathtt{r})] \\ &= -[\frac{2}{6} \log_2 \frac{2}{6} + \frac{1}{6} \log_2 \frac{1}{6} + \frac{3}{6} \log_2 \frac{3}{6}] \\ &\approx -[0.5(-1) + (0.1667)(-2.585) + (0.3333)(-1.585)] \approx 1.46 \text{ bits} \end{aligned}$$

2. Classify the test instances using a Decision Tree:

   (a) Using the **Information Gain** as a splitting criterion

   - For Information Gain, at each level of the decision tree, we choose the attribute which shows the highest difference between the entropy of the class distribution at the parent node, and the average entropy across its daughter nodes (weighted by the fraction of instances at each node);

Measures of Node Impurity.

- Gini Index
- Entropy
- Misclassification error

GINI Index: $GINI(t) = 1 - \sum_j [p(j|t)]^2$.

(1) Maximum value of GINI Index = $(1 - \frac{1}{n_c})$, equally distributed. most impure.

(2). Minimum is 0.0, all records belong to one class.

C_1	1
C_2	5

$P(C_1) = \frac{1}{6}$ $GINI = 1 - (\frac{1}{6})^2 - (\frac{5}{6})^2 = 0.278$.

$P(C_2) = \frac{5}{6}$

Splitting GINI: $GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$

If $GINI(j) - GINI\ split(j) >$ delta, then split the node j

- Binary Attributes
- Categorical Attributes
- Continuous Attributes: efficient worry about.

Entropy — Information Gain

$Entropy(e) = -\sum p(j|t) \log p(j|t)$.

$p(j|t)$: relative frequency of class j at node t.

1) Maximum $\log c$, equally distributed, high impurity

2) minimum (0.0), least impurity.

C_1	1
C_2	5

$P(C_1) = \frac{1}{6}$ $Entropy = -(\frac{1}{6}) \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = 0.65$.

$P(C_2) = \frac{5}{6}$ → Maximize

Information Gain: $GAIN_{split} = Entropy(p) - (\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i))$

$$IG(A|R) \quad = \quad H(R) - \sum_{i \in A} P(A = i) H(A = i)$$

- For the given dataset, we have 6 instances in total — 3 y and 3 n. The entropy at the top level of our tree is $H(R) = -\left[\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right] = 1$
- This is an even distribution. We're going to hope that by branching the tree according to an attribute, that will cause the daughters to have an uneven distribution — which means that we will be able to select a class with more confidence — which means that the entropy will go down.
- For example, for the attribute *Outl*, we have three attribute values: `s`, `o`, `r`.
  - When *Outl* = `s`, there are 2 instances, which are both N. The entropy of this distribution is $H(O = s) = -[0\log_2 0 + 1\log_2 1] = 0$. Obviously, at this branch, we will choose n with a high degree of confidence.
  - When *Outl* = `o`, there is a single instance, of class *Y*. The entropy here is going to be 0 as well.
  - When *Outl* = `r`, there are 2 Y instances and 1 N instance. The entropy here is $H(R) = -\left[\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right] \approx 0.9183$.
- To find the average entropy (the "mean information"), we sum the calculated entropy at each daughter multiplied by the fraction of instances at that daughter: $H(O) = \frac{2}{6}(0) + \frac{1}{6}(0) + \frac{3}{6}(0.9183) \approx 0.4592$.
- The overall information gain here is IG(O) = H(R) − MI(O) = 1 − 0.4592 = 0.5408.
- The table below lists the Mean Information and Information Gain, for each of the 5 attributes.

	R	*Outl*			*Temp*			*H*		*Wind*		*ID*					
		s	o	r	h	m	c	h	n	T	F	A	B	C	D	E	F
Y	3	0	1	2	1	1	1	2	1	0	3	0	0	1	1	1	0
N	3	2	0	1	2	0	1	2	1	2	1	1	1	0	0	0	1
Total	6	2	1	3	3	1	2	4	2	2	4	1	1	1	1	1	1
$P(Y)$	$\frac{1}{2}$	0	1	$\frac{2}{3}$	$\frac{1}{3}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{3}{4}$	0	0	1	1	1	0
$P(N)$	$\frac{1}{2}$	1	0	$\frac{1}{3}$	$\frac{2}{3}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	$\frac{1}{4}$	1	1	0	0	0	1
H	1	0	0	0.9183	0.9183	0	1	1	1	0	0.8112	0	0	0	0	0	0
MI			0.4592			0.7924			1		0.5408			0			
IG			0.5408			0.2076			0		0.4592			1			
SI			1.459			1.459			0.9183		0.9183			2.585			
GR			0.3707			0.1423			0		0.5001			0.3868			
GINI	0.5	0	0	0.4444	0.4444	0	0.5	0.5	0.5	0	0.375	0	0	0	0	0	0
GS			0.2778			0.1111			0		0.25			0.5			

- At this point, ID has the best information gain, so hypothetically we would use that to split the root node. At that point, we would be done, because each daughter is purely of a single class — however, we would be left with a completely useless classifier! (Because the IDs of the test instances won't have been observed in the training data.)
- Instead, let's take the second best attribute: *Outl*.

Splitting Based on INFO.

除分支

$$\text{Split INFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}.$$

Gain Ratio: $\text{GainRATIO}_{split} = \frac{\text{GAIN}_{split}}{\text{Split INFO}}$.

1) Adjust Information Gain by the entropy of the partitioning (Split INFO).

2) Large entropy partitioning (large number of small partitions) is penalized, which designed to overcome the disadvantage of Information Gain.

Splitting based on Classification Error

$$\text{Error}(t) = 1 - \max P(i|t)$$

1) Maximum $(1 - \frac{1}{nc})$

2) Minimum (0.0)

C_1	1
C_2	5

$P(C_1) = \frac{1}{6}$ $\quad$ Error $= 1 - \max(\frac{1}{6}, \frac{5}{6}) = \frac{1}{6}$.

$P(C_2) = \frac{5}{6}$

Decision Tree

pros: 1) inexpensive to construct

2) Extremely fast to classifying unknown records.

3) Easy to interpret for small-sized trees.

4) Accuracy is comparable to other classification technique for many simple data sets

5) No need to scale attribute values

- There are now three branches from our root node: for `s`, for `o`, and for `r`. The first two are pure, so we can't improve them anymore. Let's examine the third branch (*Outl*=`r`):
  - Three instances (D, E, and F) have the attribute value `r`; we've already calculated the entropy here to be 0.9183.
  - If we split now according to *Temp*, we observe that there is a single instance for the value `m` (of class N, the entropy is clearly 0); there are two instances for the value `c`, one of class Y, and one of class N (so the entropy here is 1). The mean information is $\frac{1}{3}(0) + \frac{2}{3}(1) \approx 0.6671$, and the information gain at this point is $0.9183 - 0.6667 \approx 0.2516$.
  - For *Humi*, we again have a single instance (with value `h`, class Y, H = 0), and two instances (of `n`) split between the two classes (H = 1). The mean information here will also be 0.6667, and the information gain 0.2516.
  - For *Wind*, there are two `F` instances, both of class Y (H = 0), and one `T` instance of class N (H = 0). Here, the mean information is 0 and the information gain is 0.9183.
  - *ID* would still look like a good attribute to choose, but we'll continue to ignore it.
  - All in all, we will choose to branch based on *Wind* for this daughter.
- All of the daughters of **r** are pure now, so our decision tree is complete:
  - *Outl*=`o` ∪(Outl=`r` ∩ *Wind* = `F`) → Y (so we classify G as Y)
  - *Outl*=`s` ∪ (*Outl* = `r` ∩ *Wind* = `T`) → N (so we classify H as N)

(b) Using the **Gain Ratio** as a splitting criterion

- Gain ratio is similar, except that we're going to weight down (or up!) by the "split information" — the entropy of the distribution of instances across the daughters of a given attribute.
- For example, we found that, for the root node, *Outl* has an information gain of 0.5408. There are 2 (out of 6) instances at the `s` daughter, 1 at the `o` daughter, and 3 at the `r` daughter.
- The split information for *Outl* is $SI(O) = -\left[\frac{2}{6}\log_2\frac{2}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{3}{6}\log_2\frac{3}{6}\right] \approx 1.459$. The Gain ratio is consequently $GR(O) = \frac{IG(O)}{SI(O)} \approx \frac{0.5408}{1.459} \approx 0.3707$.
- The values for Split Information and Gain Ratio for each attribute at the root node are shown in the table above. The best attribute (with the greatest gain ratio) at the top level this time is *Wind*.
- *Wind* has two branches: `T` is pure, so we focus on improving `F` (which has 3 Y instances (C, D, E), and 1 N instance (A)). The entropy of this daughter is 0.8112.
  - For *Outl*, we have a single instance at `s` (class N, H = 0), a single instance at `o` (class Y, H = 0), and 2 Y instances at `r` (H = 0). The mean information here is clearly 0; the information gain is 0.8112. The split information is $SI(O \mid (W = F)) = -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{2}\log_2\frac{1}{2}\right] = 1.5$, so the gain ratio is GR(O | (W = F)) $= \frac{0.8112}{1.5} \approx 0.5408$.

Cons: 1) too restrictive in terms of number rules learned.
2) The rules may be too specific and can be very complex.

Decision Tree cannot capture full information available in the data, especially for very large number of dimensions.

Solution **Bagging**: build several Decision Tree by sampling data.
1) $\frac{2}{3}$ of N original records, build decision Tree
2) build 20-50 such trees.
3) collect all decisions and apply majority rule to make the final decision.

Random Forest: similar to bagging, construction of each tree is different
1) randomly select m features, choose best attribute to split.
2) no need to prune (to reduce the size)

Depends on 1). accuracy of the trees (Large m)
2). independence (small m)

⇒ find optimal value

Underfitting: Overfitting: One way to do is stopping criterion that make sure if GINI index is smaller than some value.

pruning tree: improvement in accuracy is significant.

- For *Temp*, we have two h instances (one Y and one N, so H = 1), a single m instance (Y, H = 0), and a single c instance (Y, H = 0). The mean information is $\frac{2}{4}(1) + \frac{1}{4}(0) + \frac{1}{4}(0) \approx 0.5$, so the information gain is 0.8112 − 0.5 = 0.3112. The distribution of instances here is the same as *Outl*, so the split information is also 1.5, and the gain ratio is GR(T | (W = F)) = $\frac{0.3112}{1.5} \approx 0.2075$.
- For *Humi*, we have 3 h instances (2 Y and 1 N, H = 0.9183), and 1 n instance (Y, H = 0): the mean information is $\frac{3}{4}(0.9183) + \frac{1}{4}(0) \approx 0.6887$ and the information gain is 0.8112 − 0.6887 = 0.1225. The split information is $SI(H \mid (W = F)) = -\left[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right] \approx 0.8112$, so the gain ratio is GR(H | (W = F)) = $\frac{0.1225}{0.8112} \approx 0.1387$.
- For *ID*, the mean information is obviously still 0, so the information gain is 0.8112. The split information at this point is $-\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4}\right] = 2$, so the gain ratio is approximately 0.4056.

- Of our four choices at this point, *Outl* has the best gain ratio. The resulting daughters are all pure, so the decision tree is finished:
  - *Wind* = F ∩ (*Outl*=o ∪ *Outl*=r) → Y
  - *Wind* = T ∪ (*Wind*=F ∩ *Outl*=s) → N (so we classify G and H as n)
- Note that this decision tree is superficially similar to the tree above, but gives different classifications because of the order in which the attributes are considered.
- Note also that we didn't need to explicitly ignore the *ID* attribute for Gain Ratio (as we needed to do for Information Gain) — the split information pushed down its "goodness" to the point where we didn't want to use it anyway!

(c) Using the **Gini Index** as a splitting criterion

- For the Gini Index, at each level of the decision tree, we're going to choose the attribute that has the largest difference between the *Gini Index* of the class distribution at the parent node, and the averaged GINIs across its daughter nodes (weighted by the fraction of instances at each node); this is sometimes called *GINI-split*:

$$GS(A|R) = GINI(R) - \sum_{i \in A} P(A = i)GINI(A = i)$$

- Observe that this is the same formula as Information Gain above.
- How do we calculate GINI for this dataset?

$$GINI(X) = 1 - [p(Y)^2 + p(N)^2]$$

- You might like to compare this with the formula of entropy to see why these values are closely correlated.
- Anyway, since the steps of the method are so similar to Information Gain, we've simply recorded the GINI values and GINI-split values in the table above. You can double-check that the same tree is produced as for Information Gain.

3. What is **bagging**, in the context of **Decision Trees**?

- Bagging and Random Forests are both variants of Decision Trees.
- In Bagging, we build a number of Decision Trees by re-sampling the data:
  - For each tree, we randomly select (with repetition) N instances out of the possible N instances, so that we have the same sized data as the deterministic decision tree, but each one is based around a different data set
  - We then build the tree as usual.
  - We classify the test instance by voting | each tree gets a vote (the class it would predict for the test instance), and the class with the plurality wins.

(a) What is a **Random Forest**?

- In Random Forests, we follow the same strategy as Bagging, but:
  - When we build a tree, for each node in the tree, we randomly select some subset of the possible attributes. (Typically, roughly log k for k attributes in total.)
  - This is different to building a deterministic, where we always consider every possible attribute available (unless we already used it further up the tree).

(b) What **advantages** does a Random Forest have, with comparison to a (deterministic) Decision Tree model, or a bag of Decision Trees?

- This seemingly small change gives Random Forests a number of very important benefits:
  - As in Bagging, by using many trees, we can overcome a sampling bias in the original dataset, which might produce an undesirable (deterministic) Decision Tree (because some class is over-represented, or there is a spurious correlation between some class and some attribute)
  - By using many trees, we can overcome the problem of irrelevant attributes if some attribute has a spurious correlation with the class, it will appear near the top of the (deterministic) Decision Tree, but a given attribute will only be available occasionally ($\frac{\log k}{k}$), and many of the trees will find (hopefully) better attributes near the top of the tree.
  - By using a small proportion of the attribute set, we can build many trees in a reasonable amount of time; Bagging, on the other hand, often takes too long to generate enough trees to overcome sampling bias.