School of Computing and Information Systems
The University of Melbourne
COMP30027 Machine Learning (Semester 1, 2019)

Tutorial sample solutions: Week 4

1. Consider the following 10 instances, given so-called "gold standard" labels (assuming a 3-class problem), and the output of four supervised machine learning models:

Instance	Gold	①	②	③	④
1	A	A	A or B	A	A
2	B	A	B or C	A	?
3	A	A	A	A	A
4	C	C	B or C	A	?
5	B	B	A or B or C	A	?
6	C	A	A or C	A	?
7	C	A	A or B or C	A	?
8	A	C	A or B	A	A
9	A	A	A	A	?
10	A	A	A or C	A	A

   (a) Where possible, calculate the **accuracy** and **error rate** of the four models.

   - For systems 1 and 3, we can just employ a casual definition: accuracy is the proportion of correct responses.
     - For system 1, there are 6 correct responses (instances 1, 3, 4, 5, 9, and 10 — where the prediction is the same as the "Gold" label), out of the 10 instances, so that the accuracy is 60%.
     - For system 3, there are 5 correct responses (all of the A instances), so that the accuracy is 50%.
   - Our formal definition of Accuracy is in terms of true and false positives and negatives as follows:

$$Acc = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

   - Interpretting this expression is a little complicated, as this definition is designed for a binary class system, like Y vs. N, or A vs. "classes other than A". But here, we have a multi–class problem.
   - Consequently, we're going to need to broaden our definitions to count across classes, as follows:
     - True positives are instances where we predicted some class and the instance was truly of that class
     - False positives are instances where we predicted some class, and the instance was truly of some other class
     - False negatives are instances where we didn't predict a class at all, and it was truly of some class. (Note that only System 4 above does this.)
     - True negatives are instances where we didn't predict a class at all, and the instance was truly not of any class. (Note that this doesn't happen in tis problem, and indeed many problem frameworks — but we occasionally have a problem where "none of the above" should be treated as a distinct circumstance that isn't a class.)
   - Using the above definitions, we can formally evaluate accuracy for all four systems:
     - For System 1, there are 6 true positives (equivalent to the "correct instances" in the casual definition above), 4 false positives (instances 2, 6, 7, and 8 — where we predicted a class that wasn't the "Gold" label), 0 false negatives (because the system

predicts a class for every instance), and 0 true negatives (because every instance truly has a Gold label). So, accuracy is 60%, the same as before.

  - For System 2, we have lots of false positives, because the system makes a lot of predictions: there are 10 true positives, because the "Gold" label appears in the list of predictions for every instances; there are additionally 10 false positives, where the system makes some prediction that isn't the "Gold" label; there aren't are false negatives or true negatives. So, accuracy is $\frac{10}{20} = 50\%$
  - For System 3, we have 5 true positives (the A predictions that are actually A) and 5 false positives (the ones that aren't); no false negatives or true negatives. So, accuracy is 50%, as before.
  - For System 4, we have 4 true positives (the A predictions, which were all actually A) and 6 false negatives (the ? predictions). So the accuracy is just 40%, as we might expect.

- You might notice that this ends up looking like "micro–averaging"; in fact, this leads to an equivalent definition to micro–averaged precision for systems 1, 2, and 3 (but not 4, because of the instances where no prediction has been made).
- We need to define false negatives a little differently for micro–averaged recall, but similarly, it ends up being equivalent for systems 1, 3, and 4 (but not 2, because of the instances where we make multiple predictions).
- Error rate has a comparable formal definition:

$$Acc = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

  But it ends up as being easier to calculate as just (1 - accuracy):

  - The error rate of system 1 is $1 - 0.6 = 0.4$
  - The error rate of system 2 is $1 - 0.5 = 0.5$
  - The error rate of system 3 is $1 - 0.5 = 0.5$
  - The error rate of system 4 is $1 - 0.4 = 0.6$

(b) Where possible, calculate the **precision** and **recall**, treating class A as the "positive" class. Do the same for the B and C classes, in turn, and then calculate the **macro-averaged precision and recall**.

- Precision, for a single class, is defined as:

$$Prec = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

  Since we're just looking at a single class, we have a more straight–forward interpretion, where: a true positive is an instance where we have predicted this class, and it was actually this class; a false positive is an instance where we have predicted this class, but it was actually some other class. (We don't care are instances where we predicted some other class.)

  - For system 1, we have predicted 7 instances as A, of which 4 were actually A, and 3 were actually some other class. So, the precision is $\frac{4}{7}$.
  - For system 2, we have predicted 8 instances as A, of which 5 were actually A, and 3 were actually some other class. So, the precision is $\frac{5}{8}$. (Note that we don't care about the B and C predictions.)
  - For system 3, we have predicted all 10 instances as A, of which 5 were actually A, and 5 were actually some other class. So, the precision is $\frac{5}{10}$.
  - For system 4, we have predicted 4 instances as A, all of which were actually A. So, the precision is $\frac{4}{4} = 1$.

- Let's look briefly at the precision of B:

  - For system 1, we have predicted just a single instances as B, and it was actually B. So, the precision is $\frac{1}{1} = 1$.

  - For system 2, we have predicted 6 instances as B, of which only 2 were actually B. So, the precision is $\frac{2}{6}$.
  - For system 3, we didn't predict any instances as being B. In this case, the precision ends up not being well–defined ($\frac{0}{0}$), which will be a problem when we try to calculate a macro–average later.
  - For system 4, we also haven't predicted any B instances.
- C is similar; we will leave it as an exercise.
- Recall, for a single class, is defined as:

$$Rec = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  Here, true positives are defined as above, and false negatives are instances which are truly of this class, but we predicted some other class. Note that true positives and false negatives can't both be zero, unless there aren't any instances of this class, so it is typically always well–defined.
  - For system 1, of the 5 A instances, we have predicted 4 of them as A, and 1 as some other class. So, the recall is $\frac{4}{5}$.
  - For system 2, of the 5 A instances, we have predicted all of them as A; however, we have *also* predicted some other class 3 times. So, the recall is $\frac{5}{8}$. (You might see some different interpretations elsewhere, like $\frac{5}{5}$; it turns out that people don't agree how we should handle systems with multiple predictions like this one.)
  - For system 3, of the 4 A instances, we have predicted all of them as A. So, the recall is $\frac{5}{5} = 1$.
  - For system 4, of the 5 A instances, we have predicted 4 of them as A. So, the recall is $\frac{4}{5} = 1$.
- Macro–averaging precision or recall simply refers to taking the (arithmetic) mean across the values for each class.
  - For system 1, the precision of A is $\frac{4}{7}$, the precision of B is $\frac{1}{1}$, and the precision of C is $\frac{1}{2}$. Consequently, the macro–averaged precision is $\frac{1}{3}(\frac{4}{7} + \frac{1}{1} + \frac{1}{2}) = \frac{29}{42} \approx 0.69$
  - For system 1, the recall of A is $\frac{4}{5}$, the recall of B is $\frac{1}{2}$, and the recall of C is $\frac{1}{3}$. Consequently, the macro–averaged recall is $\frac{1}{3}(\frac{4}{5} + \frac{1}{2} + \frac{1}{3}) = \frac{49}{90} \approx 0.54$
- We will leave the other calculations as an exercise, but note that the macro–averaged precisions of systems 3 and 4 are impossible to calculate.

2. What is the difference between evaluating using a **holdout** strategy and evaluating using a **cross–validation strategy**?

   - In a holdout evaluation strategy, we partition the data into a training set and a test set: we build the model on the former, and evaluate on the latter.
   - In a cross-validation evaluation strategy, we do the same as above, but a number of times, where each iteration uses one partition of the data as a test set and the rest as a training set (and the partition is different each time).

   (a) What are some reasons we would prefer one strategy over the other?

   - Holdout is subject to some random variation, depending on which instances are assigned to the training data, and which are assigned to the test data. Any instance that forms part of the model is excluded from testing, and *vice versa*. This could mean that our estimate of the performance of the model is way off, or changes a lot from data set to data set.
   - Cross–validation mostly solves this problem: we're averaging over a bunch of values, so that one weird partition of the data won't throw our estimate of performance completely off; also, each instance is used for testing, but also appears in the training data for the models built on the other partitions. It usually takes much longer to cross-validate, however, because we need to train a model for every test partition.

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	?

3. For the following dataset:

   (a) Classify the test instances using the method of 0-R.

   - 0-R is the quintessentially **baseline** classifier: we throw away all of the attributes, other than the class labels, and just predict each test instance according to whichever label is most common in the training data. (Hence, it is also common called the "majority class classifier".)
   - In this case, the two labels `Y` and `N` are equally common in the training data — so we are required to apply a tie–breaker. Remember that we're simply choosing one label to be representative of the entire collection, and both labels seem equally good for that here: so, let's say `N`.
   - Consequently, both test instances are classified as `N`.

   (b) Classify the test instances using the method of 1-R.

   - 1-R is a slightly better baseline, which requires us to choose a single attribute to represent the entire decision–making process. For example, if *Outl* is our preferred attribute, then we base the classification of each test instance solely based on its value of *Outl*. (This is sometimes called a "Decision Stump".)
   - Given our preferred attribute, we will label a test instance according to whichever label in the training data was most common, for training instances with the corresponding attribute value.
   - How do we decide which attribute to choose? Well, the most common method is simply by counting the errors made on the training instances.
   - Let's say we chose *Outl*: first, we need to observe the predictions for the 3 values (`s`, `o`, and `r`), and then we will count up the errors that those predictions will make on the training data (it turns out that we can do this simultaneously):
     - When *Outl* is `s`, there are two such instances in the training data. Both of these instances are labelled as `N`: our label for this attribute value will be `N`. We will therefore predict the label of both of these training instances correctly (we will predict `N`, and both of these instances are actually `N`).
     - When *Outl* is `o`, there is just a single instance in the training data, labelled as `Y`: our label for this attribute value will be `Y`. We will therefore predict the label of this training instance correctly.
     - When *Outl* is `r`, there are three such instances in the training data. Two of these instances are labelled as `Y`, the other is `N`: our label for this attribute value will be `Y` (as there are more `Y` instances than `N` instances — you can see that we're applying the method of 0-R here). We will therefore make one error: the instance which was actually `n`.
   - In total, that is 1 error for *Outl*. Hopefully, you can see that this is a very wordy explanation of a very simple idea. (We will go through the other attributes more quickly.)
   - For *Temp*:

  - When *Temp* is `h`, there are two `N` instances and one `Y`: we will make one mistake.
  - When *Temp* is `m`, there is one `Y` instance: we won't make any mistakes.
  - When *Temp* is `c`, there is one `N` instance and one `Y` instance: we will make one mistake.
- This is 2 errors in total for *Temp*, which means that it's less good that *Outl*.
- We'll leave the other attributes as an exercise, and assume that *Outl* was the best attribute (it's actually tied with *Wind*): how do the test instances get classified?
  - For test instance G, *Outl* is `o` — the 0-R classifier over the `o` instances from the training data gives `Y`, so we predict `Y` for this instance.
  - For test instance H, *Outl* is `s` — the 0-R classifier over the `s` instances from the training data gives `N`, so we predict `N` for this instance.

(c) Classify the test instances using the ID3 **Decision Tree** method:

i. Using the **Information Gain** as a splitting criterion

- For Information Gain, at each level of the decision tree, we're going to choose the attribute that has the largest 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);

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

- In this dataset, we have 6 instances total — 3 `Y` and 3 `N`. The entropy at the top level of our tree is $H(R) = -[\frac{3}{6}\log_2 \frac{3}{6} + \frac{3}{6}\log_2 \frac{3}{6}] = 1$.
- This is a very 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 $H(O = \texttt{s}) = -[0 \log 0 + 1 \log 1] = 0$. 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 $H(O = \texttt{r}) = -[\frac{1}{3}\log_2 \frac{1}{3} + \frac{2}{3}\log_2 \frac{2}{3}] \approx 0.9183$.
- 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: $MI(O) = \frac{2}{6}(0) + \frac{1}{6}(0) + \frac{3}{6}(0.9183) \approx 0.4592$.
- 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.
- 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*.
- 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 any more. 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.

	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					

- 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 $\frac{1}{3}(0) + \frac{2}{3}(1) \approx 0.6667$, 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 $\cup$(*Outl*=r $\cap$ *Wind*=F) $\rightarrow$ Y (so we classify G as Y)
  - *Outl*=s $\cup$(*Outl*=r $\cap$ *Wind*=T) $\rightarrow$ N (so we classify H as N)

ii. 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 $SI(O) = -[\frac{2}{6}\log_2\frac{2}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{3}{6}\log_2\frac{3}{6}] \approx 1.459$. 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 ($SI(O \mid (W = F)) = -[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{2}\log_2\frac{1}{2}] = 1.5$, so the gain ratio is $GR(O \mid (W = F)) = \frac{0.8112}{1.5} \approx 0.5408$.
  - 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 $\frac{2}{4}(1) + \frac{1}{4}(0) + \frac{1}{4}(0) = 0.5$, 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 $GR(T \mid (W = F)) = \frac{0.3112}{1.5} \approx 0.2075$.

  - 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 $\frac{3}{4}(0.9183) + \frac{1}{4}(0) = 0.6887$ and the information gain is $0.8112 - 0.6887 = 0.1225$. The split information is $SI(H \mid (W = F)) = -[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}] \approx 0.8112$, so the gain ratio is $GR(H \mid (W = F)) = \frac{0.1225}{0.8112} \approx 0.1387$.
  - For *ID*, the mean information is obviously still 0, so the information gain is 0.8112. The split information at this point is $\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} = 2$, 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` $\cap$(*Outl*=`o` $\cup$ *Outl*=`r`) $\rightarrow$ `Y`
  - *Wind*=`T` $\cup$(*Wind*=`F` $\cap$ *Outl*=`s`) $\rightarrow$ `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!