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

Tutorial sample solutions: Week 6

ID	A (°C)	B (mm)	C (hPa)	CLASS
1	22.5	4.6	1021.2	AUT
2	16.7	21.6	1027.0	AUT
3	29.6	0.0	1012.5	SUM
4	33.0	0.0	1010.4	SUM
5	13.2	16.4	1019.5	SPR
6	14.9	8.6	1016.4	SPR
7	18.3	7.8	995.4	WIN
8	16.0	5.6	1012.8	WIN

1. What is **Discretisation**, and where might it be used?

   - We have a (continuous) numeric attribute, but we wish to have a nominal (or ordinal) attribute.
   - Some learners (like Decision Trees) generally work better with nominal attributes; some datasets inherently have groupings of values, where treating them as an equivalent might make it easier to discern underlying patterns.

   (a) Summarise some approaches to **supervised** discretisation.

   - Our general idea here is to sort the possible values, and create a nominal value for a region where most of the instances having the same label.

   (b) Discretise the above dataset according to the (unsupervised) methods of **equal width**, **equal frequency**, and **k-means** (breaking ties where necessary).

   - Equal width divides the range of possible values seen in the training set into equally–sized sub-divisions, regardless of the number of instances (sometimes 0!) in each division.
     - For attribute C above, the largest value is 1027.0 and the smallest value is 995.4; the difference is 31.6:
     - If we wanted two "buckets", each bucket would be 15.8 wide, so that instances between 995.4 and 1011.2 take one value (4 and 7), and instances between 1011.2 and 1027.0 take another (1, 2, 3, 5, 6, and 8).
     - If we wanted three "buckets", each bucket would be about 10.5 wide; so that instances between 995.4 and 1005.9 take one value (just 7), instances between 1005.9 and 1016.4 take another value (3, 4, 6, and 8), and instances between 1016.4 and 1027.0 take yet another value (1, 2, and 5).
   - Equal frequency divides the range of possible values seen in the training set, such that (roughly) the same number of instances appear in each bucket.
     - For attribute C above, if we sort the instances in ascending order, we find 7, 4, 3, 8, 6, 5, 1, 2.
     - If we wanted two "buckets", each bucket would have four instances; so that instances 7, 4, 3, and 8 would take one value, and the rest would take the other value.
     - Sometimes, we also need to explicitly define the ranges, in case we obtain new instances later that we need to transform. There is some question about the intermediate values (between 1012.8 and 1016.4, in this case); typically, we place the dividing point at the median.

- k-means is actually a "clustering" approach, but it can work well in this context. If we want k buckets, we randomly choose k points to act as seeds. We then have an iterative approach where we: assign each instance to the bucket of the closest seed; update the "centroid" of the bucket with the mean of the values.
  - For attribute C above, let's say we begin with two seeds: instance 3 (1012.5, bucket A) and instance 4 (1010.4, bucket B).
  - Instance 1 (1021.2) is closer to A than B; 2 (1027.0) is closer to A; 3 is closer to A; 4 is closer to B; 5 (1019.5) is closer to A; 6 (1016.4) is closer to A; 7 (995.4) is closer to B; 8 (1012.8) is closer to A.
  - We take the average of the values of instances 1, 2, 3, 5, 6, and 8 (1018.2) to be representative of cluster A, and the average of instances 4 and 7 (1002.9) to be representative of cluster B.
  - Now, we iterate: 1 is still closer to A; 2 is still closer to A; 3 is still closer to A; 4 is still closer to B; 5 is still closer to A; 6 is still closer to A; 7 is still closer to B; 8 is still closer to A.
  - Since this is the same assignment of values to clusters, we stop: instances 4 and 7 will have one value, and the other instances will have another value.

2. Find the (sample) **mean** and (sample) **standard deviation**[1] for the attibutes in the above dataset:

   (a) In its entirety, and;

   - We have actually already been calculating sample means, in the centroid calculations for k-means.
   - For example, the mean μ of the values of attribute C is determined by summing the values and dividing by the number of values:

$$\begin{aligned}\mu_c &= \frac{1}{N}\sum c_i \\ &= \frac{1}{8}(1021.2 + 1027.0 + 1012.5 + 1010.4 + 1019.5 + 1016.4 + 995.4 + 1012.8) \\ &= \frac{1}{8}(8115.2) = 1014.4\end{aligned}$$

   - The sample standard deviation is calculated by squaring the difference from the sample mean, dividing by 1 less than the number of values (this is a little surprising), and then taking the (positive) square root:

$$\begin{aligned}\sigma_c &= \sqrt{\frac{\sum(c_i - \mu_c)^2}{(N-1)}} \\ &= \sqrt{\frac{46.24 + 158.76 + 3.61 + 16.0 + 26.01 + 4.0 + 361 + 2.56}{7}} \\ &\approx \sqrt{88.31} \approx 9.40\end{aligned}$$

   (b) For each individual class[2].

   (c) How could we use this information when building a classifier over this data?

   - We could construct a **Gaussian** probability density function, which would allow us to estimate the probability of observing any given value, using pointwise estimation, or by counting the number of standard deviations it is from the mean (its **z-score**).

[1] n.b. You might need a calculator.
[2] We would ideally do this with more instances!

Given the following dataset:

ID	*Outl*	*Temp*	*Humi*	*Wind*	PLAY
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

3. If we wished to perform **feature selection** (or **feature weighting**) on this dataset, where the class is PLAY:

   (a) Which of *Humi* and *Wind* has the greatest **Pointwise Mutual Information** for the class Y? What about N?

   - To determine Pointwise Mutual Information (PMI), we compare the joint probability to the product of the prior probabilities as follows:

$$PMI(A;C) = \log_2 \frac{P(A \cap C)}{P(A)P(C)}$$

     Note that this formulation only really makes sense for binary attributes and binary classes (which we have here.)
   - Suitably interpretting $P(X)$ as $P(X = \texttt{Y})$ (or T or h here), we find:

$$\begin{aligned} PMI(\textit{Humi}; \text{PLAY}) &= \log_2 \frac{P(\textit{Humi} \cap \text{PLAY})}{P(\textit{Humi})P(\text{PLAY})} \\ &= \log_2 \frac{\frac{2}{6}}{\frac{4}{6}\frac{3}{6}} \\ &= \log_2(1) = 0 \\ PMI(\textit{Wind}; \text{PLAY})) &= \log_2 \frac{P(\textit{Wind} \cap \text{PLAY})}{P(\textit{Wind})P(\text{PLAY})} \\ &= \log_2 \frac{\frac{0}{6}}{\frac{2}{6}\frac{3}{6}} \\ &= \log_2(0) = -\infty \end{aligned}$$

   - So, we find that *Wind* is (perfectly) negatively correlated with PLAY; whereas *Humi* is (perfectly) uncorrelated.
   - You should compare these results with the negative class PLAY=N, where *Wind* is positively correlated, but *Humi* is still uncorrelated.

   (b) Which of the attributes has the greatest **Mutual Information** for the class, as a whole? (Note that we need to extend the lecture definition to handle non–binary attributes.)

   - A general form of the Mutual Information (MI) is as follows:

$$MI(X;C) = \sum_{x \in X} \sum_{c \in \{Y,N\}} P(x,c) PMI(x;c)$$

     Effectively, we're going to consider the PMI of every possible attribute value–class combination, weighted by the proportion of instances that actually had that combination.
   - To handle cases like $PMI(\textit{Wind})$ above, we are going to equate $0 \log 0 \equiv 0$ (which is true in the limit anyway).

- For *Outl*, this is going to look like:

$$\begin{aligned} MI(\textit{Outl}) &= P(\mathtt{s},\mathtt{Y})PMI(\mathtt{s};\mathtt{Y}) + P(\mathtt{o},\mathtt{Y})PMI(\mathtt{o};\mathtt{Y}) + P(\mathtt{r},\mathtt{Y})PMI(\mathtt{r};\mathtt{Y}) + \\ &\quad P(\mathtt{s},\mathtt{N})PMI(\mathtt{s};\mathtt{N}) + P(\mathtt{o},\mathtt{N})PMI(\mathtt{o};\mathtt{N}) + P(\mathtt{r},\mathtt{N})PMI(\mathtt{r};\mathtt{N}) \\ &= \frac{0}{6}\log_2\frac{\frac{0}{6}}{\frac{2}{6}\frac{3}{6}} + \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{1}{6}\frac{3}{6}} + \frac{2}{6}\log_2\frac{\frac{2}{6}}{\frac{3}{6}\frac{3}{6}} + \\ &\quad \frac{2}{6}\log_2\frac{\frac{2}{6}}{\frac{2}{6}\frac{3}{6}} + \frac{0}{6}\log_2\frac{\frac{0}{6}}{\frac{1}{6}\frac{3}{6}} + \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{3}{6}\frac{3}{6}} \\ &\approx 0\log_2 0 + (0.1667)(1) + (0.3333)(0.4150) + (0.3333)(1) + 0\log_2 0 + (0.1667)(-0.5850) \\ &\approx 0.541 \end{aligned}$$

- It's worth noting that while some individual terms can be negative, the sum must be at least zero.
- For *Temp*, this is going to look like:

$$\begin{aligned} MI(\textit{Temp}) &= P(\mathtt{h},\mathtt{Y})PMI(\mathtt{h};\mathtt{Y}) + P(\mathtt{m},\mathtt{Y})PMI(\mathtt{m};\mathtt{Y}) + P(\mathtt{c},\mathtt{Y})PMI(\mathtt{c};\mathtt{Y}) + \\ &\quad P(\mathtt{h},\mathtt{N})PMI(\mathtt{h};\mathtt{N}) + P(\mathtt{m},\mathtt{N})PMI(\mathtt{m};\mathtt{N}) + P(\mathtt{c},\mathtt{N})PMI(\mathtt{c};\mathtt{N}) \\ &= \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{3}{6}\frac{3}{6}} + \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{1}{6}\frac{3}{6}} + \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{2}{6}\frac{3}{6}} + \\ &\quad \frac{2}{6}\log_2\frac{\frac{2}{6}}{\frac{3}{6}\frac{3}{6}} + \frac{0}{6}\log_2\frac{\frac{0}{6}}{\frac{1}{6}\frac{3}{6}} + \frac{1}{6}\log_2\frac{\frac{1}{6}}{\frac{2}{6}\frac{3}{6}} \\ &\approx (0.1667)(-0.5850) + (0.1667)(1) + (0.1667)(0) + (0.3333)(0.4150) + 0 + (0.1667)(-0.5850 \\ &\approx 0.110 \end{aligned}$$

- We will leave the workings as an exercise, but the Mutual Information for *Humi* is 0, and for *Wind* is 0.459.
- Consequently, *Outl* appears to be the best attribute (perhaps as we might expect), and *Wind* also seems quite good; whereas *Temp* is not very good, and *Humi* is completely unhelpful.