1. Consider the following dataset:

	y_1	y_2	y_3	z
x_1	a	a	a	+
x_2	c	b	c	+
x_3	c	a	c	+
x_4	b	a	a	-
x_5	a	b	c	-
x_6	b	b	c	-

Plot the learned decision tree using information gain (Shannon entropy). Show your calculations.

Intuitively, the amount of information we can gather upon observing a random event is inversely proportional to the probability of it happening. Shannon's notion of **entropy** is a formalization of this idea. The entropy of a random variable is the average amount of information we can gather upon observing it, and is defined as follows:

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{p(x)} = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$

Here, $\mathcal{X}$ is the set of all possible values of X and $p(x)$ is the probability of X taking the value x.

The **information gain** concept is a measure of how much information we gain by observing a random variable X - the larger the information gain value, the more we can extract from a given feature for the given dataset. It is defined as the difference between the entropy of the random variable X and the *conditional entropy* of X given Y: a measure of how much information is needed to describe X given that we know Y.

$$\mathrm{IG}(X, Y) = H(X) - H(X|Y)$$

Having these definitions in mind, it should now be trivial to plot the decision tree for the dataset above. We start by computing the entropy of the target variable z:

$$H(z) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

Learning a decision tree will be intrinsically related to the features' information gain measure: for each level, we will select the feature that maximizes the information gain with respect to the target variable, making it that level's decision node. Let's compute the information gain for each feature:

$$H(z \mid y_1) = \sum_{y \in \mathcal{Y}_1} p(y_1 = y) H(z \mid y_1 = y) = \frac{2}{6}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) + \frac{2}{6}\left(-\frac{2}{2}\log_2\frac{2}{2}\right) + \frac{2}{6}\left(-\frac{2}{2}\log_2\frac{2}{2}\right) = \mathbf{\frac{1}{3}}$$

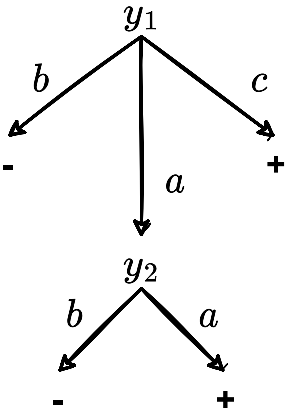
$$IG(z, y_1) = 1 - \frac{1}{3} = \mathbf{\frac{2}{3}}$$

Performing similar computations for the other features, we obtain:

$$\begin{aligned} H(z \mid y_2) &= 0.9183, \qquad & IG(z, y_2) &= 0.082 \\ H(z \mid y_3) &= 1, \qquad & IG(z, y_3) &= 0 \end{aligned}$$

As mentioned above, we will select the feature that maximizes the information gain for each level. In this case, y_1 is the feature that maximizes the information gain, hence it will be the root node of the decision tree. Note how there are already a couple of leaves on our tree: these indicate that, regarding the respective decision tree node, the target variable is already fully determined by the feature values (i.e for a given feature value, the target variable is always the same).

Continuing the process, and following the path for the feature y_1 that takes the value a, we can choose either y_2 or y_3 as the next decision node, since the entropies of y_2 and y_3 are the same, zero (considering data conditioned by $y_1 = a$). Choosing y_2 as the next decision node, we're now left with only leaf nodes, therefore no further decision nodes should be added to the tree.



2. Show if a decision tree can learn the AND, OR and XOR logical functions.

Note that the afore-mentioned logical functions can be represented as shown in the following ”dummy data sets”:

	y_1	y_2	z
x_1	0	0	0
x_2	0	1	0
x_3	1	0	0
x_4	1	1	1

Table 1: AND logical function

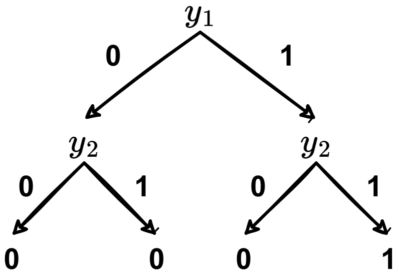
	y_1	y_2	z
x_1	0	0	0
x_2	0	1	1
x_3	1	0	1
x_4	1	1	1

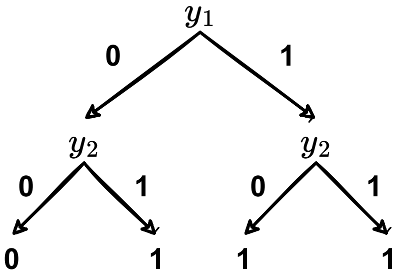
Table 2: OR logical function

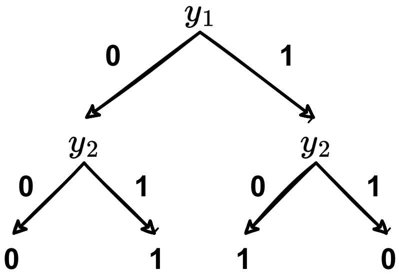
	y_1	y_2	z
x_1	0	0	0
x_2	0	1	1
x_3	1	0	1
x_4	1	1	0

Table 3: XOR logical function

Each one of these data sets can be learned by a decision tree, of course:







3. Consider the following testing targets, z, and the corresponding predictions, $\hat{z}$, by a decision tree:

$$z = [A, A, A, B, B, B, C, C, C, C]$$
$$\hat{z} = [B, B, A, C, B, A, C, A, B, C]$$

   (a) Draw the confusion matrix.

   (b) Compute the accuracy and recall (sensitivity) for each class.

   (c) Regarding class C, identify its precision and F-measure.

   (d) Identify the accuracy, sensitivity and precision of a random classifier.

As we know, a confusion matrix is a $K \times K$ matrix, where K is the number of classes in our target label, which aims to aid in the visualization of the performance of a (usually supervised) classifier. In this case, we have three classes, A, B and C, therefore our confusion matrix will be a 3×3 matrix. Considering the *real vs predicted* labels referenced above, we can fill the confusion matrix as follows:

		Real		
		A	B	C
Projected	A	1	1	1
	B	2	1	1
	C	0	1	2

The **accuracy** of a classifier is the proportion of correct predictions made by the classifier. In this case, we have 4 correct predictions (given by the matrix' trace), out of 10 total predictions. Therefore, the accuracy is, of course, $4/10 = 0.4$.

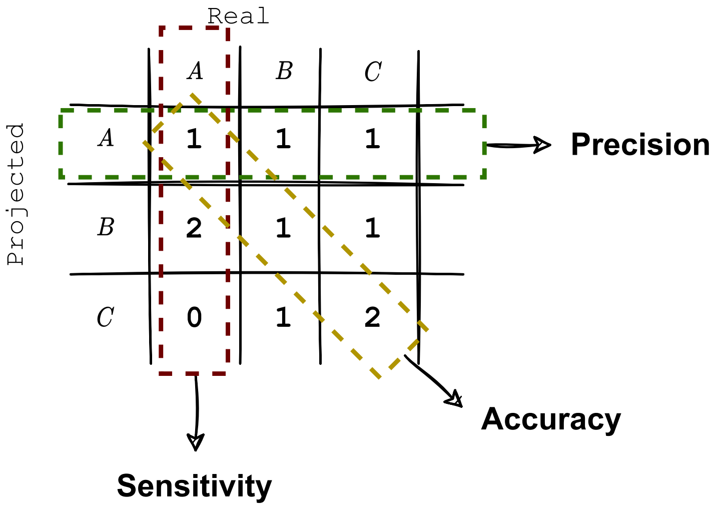
The **recall** (also known as **sensitivity**) measures the proportion of correctly X-predicted instances out of all instances of class X. Measuring the recall for each class, we have the following:

$$\text{recall}(A) = \frac{1}{1+2+0} = \frac{1}{3}, \quad \text{recall}(B) = \frac{1}{1+1+1} = \frac{1}{3}, \quad \text{recall}(C) = \frac{2}{1+1+2} = \frac{1}{2}$$

The **precision** is essentially a class-wise version of the accuracy measure. It measures the proportion of correctly X-predicted instances out of all instances predicted as X. Measuring the precision for each class, we have the following:

$$\text{precision}(A) = \frac{1}{1+1+1} = \frac{1}{3}, \quad \text{precision}(B) = \frac{1}{2+1+1} = \frac{1}{4}, \quad \text{precision}(C) = \frac{2}{0+1+2} = \frac{2}{3}$$

These measures can be more easily visualized as shown in the following figure:



	A	B	C
A	1	1	1
B	2	1	1
C	0	1	2

The **F-measure** is the harmonic mean of the precision and recall, and is defined as:

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

In this case, we have $\beta = 1$ (since the question's statement asks us for the F1-measure). As such, we'll have the following:

$$F_{1,a} = \frac{2 \cdot \text{precision}_a \cdot \text{recall}_a}{\text{precision}_a + \text{recall}_a} = \frac{2 \cdot \frac{1}{3} \cdot \frac{1}{3}}{\frac{1}{3} + \frac{1}{3}} = \frac{1}{3}, \quad F_{1,b} = \frac{2 \cdot \frac{1}{4} \cdot \frac{1}{3}}{\frac{1}{4} + \frac{1}{3}} = \frac{2}{7}, \quad F_{1,c} = \frac{2 \cdot \frac{2}{3} \cdot \frac{1}{2}}{\frac{2}{3} + \frac{1}{2}} = 0.5714$$

Finally, let's talk about the **random classifier**. A random classifier is a classifier that makes random predictions, and is usually used as a baseline for comparison with other classifiers. In this case, we have three classes, A, B and C, therefore the random classifier will predict each class with probability $\frac{1}{3}$. As such, the accuracy of the random classifier is also $\frac{1}{3}$.

Regarding both sensitivity and precision, we have the following (considering X as the class predicted by the random classifier, $\overline{X}$ as all the remaining classes, and $p = P(X)$):

$$\text{recall}(X) = \frac{TX}{TX + F\overline{X}} = \frac{p \cdot \#X}{p \cdot \#X + (1-p)\#X} = p, \quad \text{precision}(X) = \frac{TX}{TX + FX} = \frac{p \cdot \#X}{p \cdot \#X + p\#\overline{X}} = \frac{\#X}{\#X + \#\overline{X}}$$

Putting it into words, TX is the number of correct X predictions - considering a random classifier, and $\#X$ amount of X instances, there'll be $p \cdot \#X$ correct X predictions, of course.

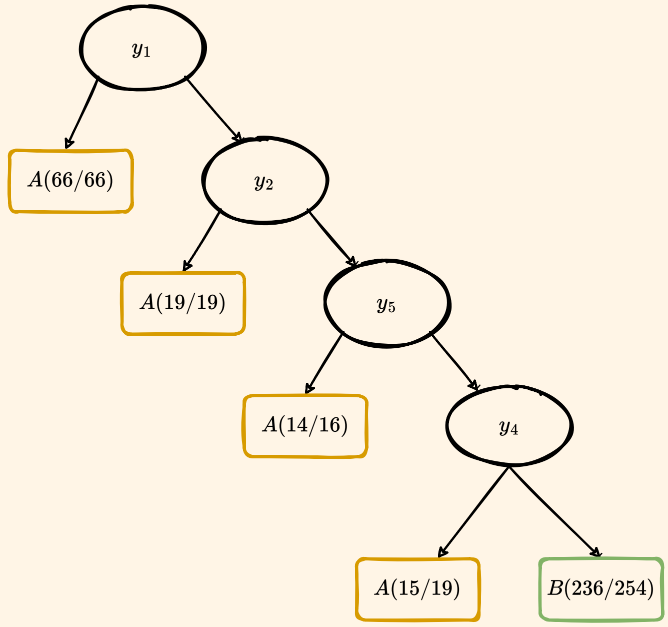
$F\overline{X}$ is the number of incorrect non-X guesses - that is, the amount of times we should have predicted X but didn't. It's intuitive that, in the case at hands for each time we hit, we'll end up missing 2 times. Therefore, $F\overline{X} = (1-p) \cdot \#X$.

Finally, FX is the number of incorrect X guesses - that is, the amount of times we should have predicted $\overline{X}$ but predicted X. We know that the probability of predicting $\overline{X}$ as X is p, and as such we'll have $p \cdot \#\overline{X}$ incorrect X guesses.

Note that, considering the formulas above, the recall for all classes is given by $p = 1/3$, while the precision differs between classes:

$$\text{precision}(A) = \text{precision}(B) = \frac{3}{3+(3+4)} = \frac{3}{10}, \quad \text{precision}(C) = \frac{4}{4+(3+3)} = \frac{4}{10}$$

4. Consider a dataset composed by 374 records, described by 6 variables, classified according to the following decision tree:



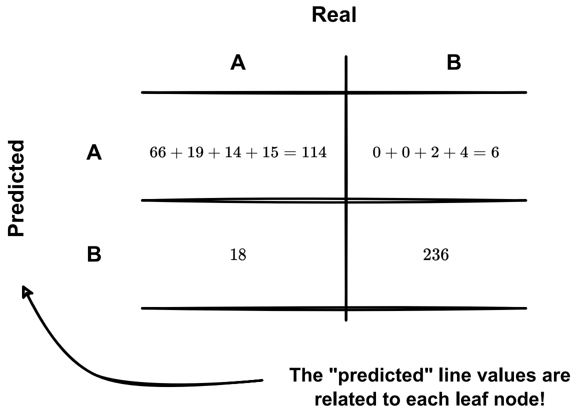
Each leaf in the tree shows the label, number of classified records with the label, and total number of observations in the leaf. The positive class is the minority class.

(a) Compute the confusion matrix.

(b) Compare the accuracy of the given tree versus a pruned tree, with only two nodes. Is there any evidence towards overfitting?

(c) Are decision trees learned from high-dimensional data susceptible to underfitting? Why does an ensemble of DTs minimize this problem?

By looking at the question statement's figure, we can easily build its respective confusion matrix:



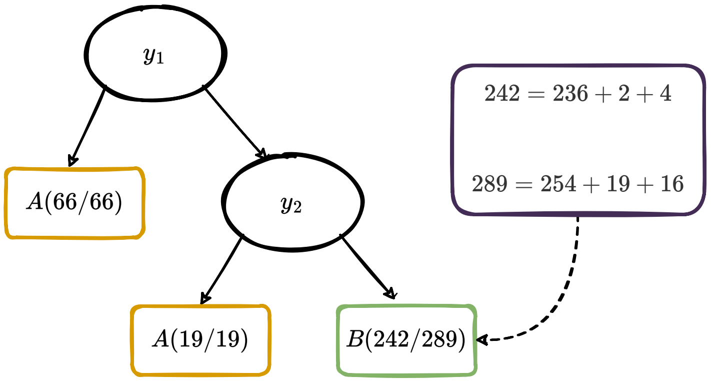
		Real	
		A	B
Predicted	A	$66 + 19 + 14 + 15 = 114$	$0 + 0 + 2 + 4 = 6$
	B	18	236

The "predicted" line values are related to each leaf node!

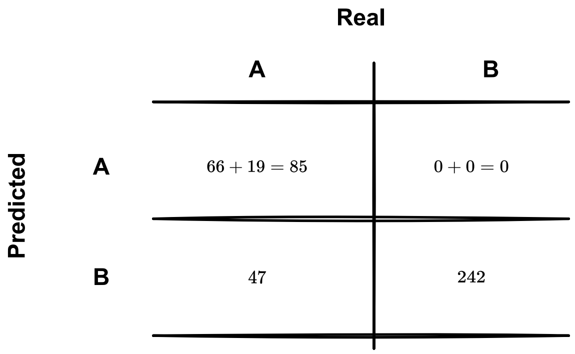
By consulting the confusion matrix above, we can see that the accuracy of the given tree is given by (note that it's the equivalent of dividing the matrix's trace by the sum of all its elements):

$$\text{accuracy} = \frac{TA + TB}{TA + TB + FA + FB} = \frac{114 + 236}{114 + 236 + 6 + 18} = 0.936$$

By pruning the tree to only two decision nodes, considering that the second decision node already has an A-related leaf, every leaf in the now-pruned path will belong to a newly created B-related leaf. As such, the decision tree now looks like this:



The new confusion matrix and accuracy score are as follows:



		Real	
		A	B
Predicted	A	$66 + 19 = 85$	$0 + 0 = 0$
	B	47	242

$$\text{accuracy} = \frac{TA + TB}{TA + TB + FA + FB} = \frac{85 + 242}{85 + 242 + 0 + 47} = 0.874$$

The accuracy score of the pruned tree is lower than the one of the original tree - here, less complexity in the DT ends up being disadvantageous for the model. We can't exactly say that there's evidence towards overfitting, though, since we have no validation/testing sets to compare the two models' performance on unseen data.

This article gives a good explanation on why tree ensembles (and random forests specifically) are less susceptible to overfitting than single decision trees.