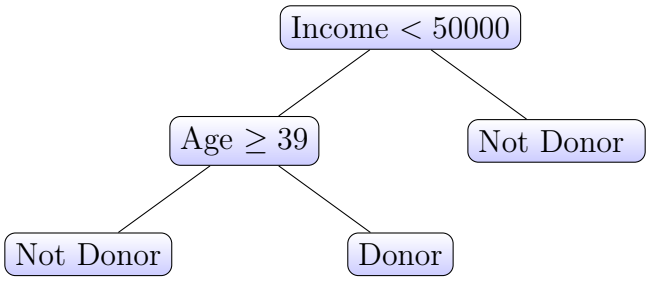
1. **Problem 1:** Develop manually (on paper) a decision-tree model that exhibits 100% accuracy on the Donor training data introduced in Homework 3.

Name	Age	Income	Donor
Nancy	21	37,000	N
Jim	27	41,000	N
Allen	43	61,000	Y
Jane	38	55,000	N
Steve	44	30,000	N
Peter	51	56,000	Y
Sayani	53	70,000	Y
Lata	56	74,000	Y
Mary	59	25,000	N
Victor	61	68,000	Y
Dale	63	51,000	Y

Right branch represent "yes" and left branch represents "no"



2. **Problem 2:** Show that the entropy of a node in a decision tree never increases after splitting it into smaller successor nodes.

Let $Y = \{y_1, y_2, ..., y_c\}$ where c represents the number of classes in attribute Y (the attribute we are hoping to predict). Also, let $X = \{x_1, x_2, \ldots, x_k\}$ where k is the number of attribute values a certain attribute X has. In the very beginning, when we are at the root of the tree, the entropy is given by the following equation:

$$E(Y) = -\Sigma_{j=1}^{c} P(y_j) log_2 P(y_j) = \Sigma_{j=1}^{c} \Sigma_{i=1}^{k} P(x_i, y_j) log_2 P(y_j),$$

which uses the following idea from the law of total probability: $P(y_j) = \Sigma_{i=1}^{k} P(x_i, y_j)$. After we split an attribute X, the entropy of each child node $X = x_i$ is

$$E(Y \mid x_i) = -\Sigma_{j=1}^{c} P(y_j \mid x_i) log_2 P(y_j \mid x_i),$$

where the proportion of the sample where $X = x_i$ belong to class y_j is represented by $P(y_j|x_i)$. Then, after we split X, the weighted entropy of the subsequent, child nodes

is as follows:

$$\begin{aligned} E(Y \mid X) &= \Sigma_{i=1}^{k} P(x_i) E(Y \mid x_i) \\ &= -\Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(x_i) P(y_j \mid x_i) log_2 P(y_j \mid x_i) \\ &= -\Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) log_2 P(y_j \mid x_i), \end{aligned}$$

from the law of total probability. To prove that the entropy never increases after splitting a node, we must show that $E(Y \mid X) \leq E(Y \mid X)$, which we can do by showing that $E(Y \mid X) - E(Y \mid X) \leq 0$.

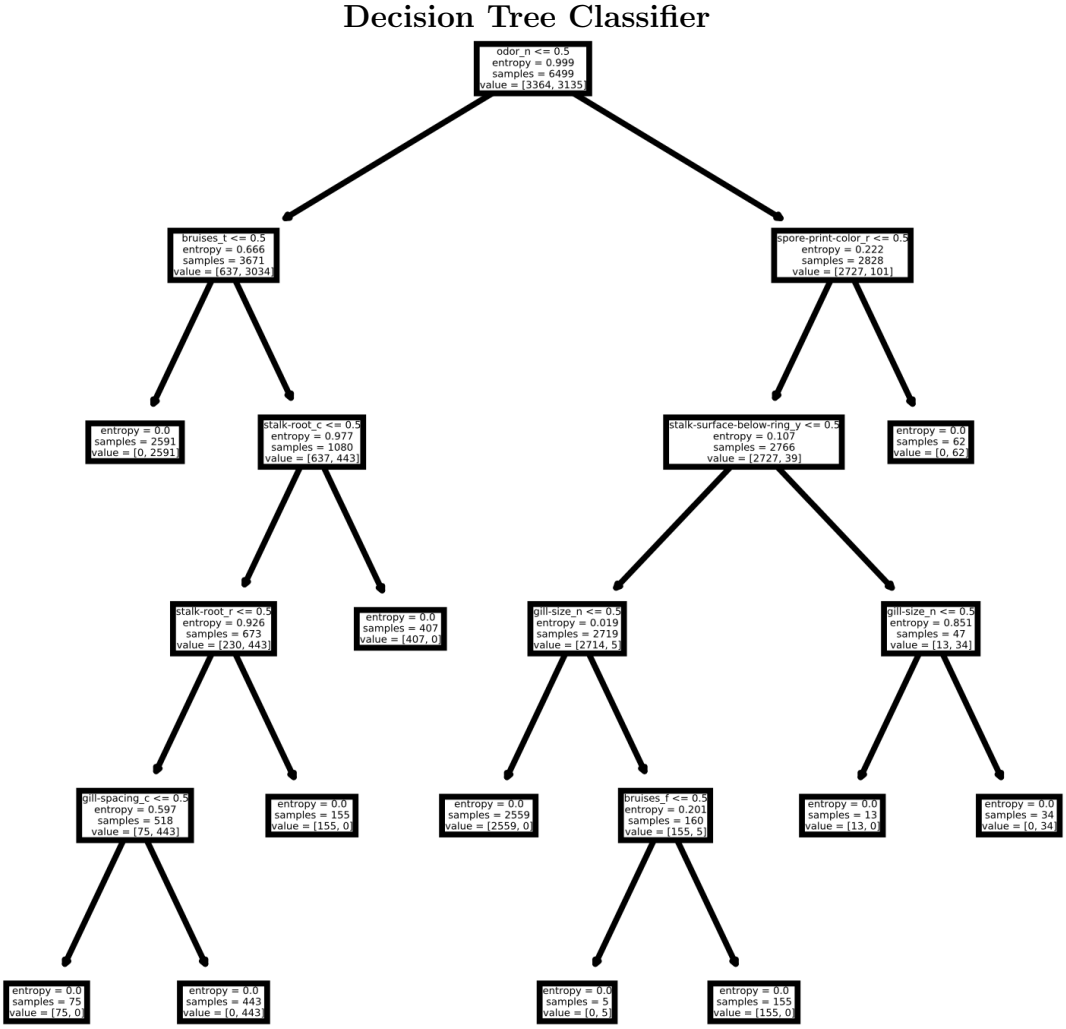
$$\begin{aligned} E(Y \mid X) - E(Y) &= -\Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) log_2 P(y_j \mid x_i) + \Sigma_{j=1}^{c} P(y_j, x_i) log_2 P(y_j \mid x_i) \\ &= \Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) log_2 P(y_j \mid x_i) - P(y_j, x_i) log_2 P(y_j) \\ &= \Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) log_2 \frac{P(y_j)}{P(y_j \mid x_i)} \\ &= \Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) log_2 \frac{P(y_j) P(x_i)}{P(y_j, x_i)} \\ &\leq log_2 (\Sigma_{i=1}^{k} \Sigma_{j=1}^{c} P(y_j, x_i) \frac{P(y_j) P(x_i)}{P(y_j, x_i)}) : by\ Jensen's\ Inequality \\ &= log_2 [\Sigma_{i=1}^{k} P(x_k) \Sigma_{j=1}^{c} P(y_j)] \\ &= log_2 (1) \\ &= 0 \end{aligned}$$

Thus, the entropy of a node in a decision tree never increases after splitting it into smaller successor nodes. //

3. **Problem 3:** Consider a data set containing four points located at the corners of the square. The two points on one diagonal belong to one class, and the two points on the other diagonal belong to the other class. Is this data set linearly separable? Provide a proof.

   Methods: I used the *DecisionTreeClassifier* function in Python's *scikitlearn* package to create the decision tree, with an 80-20, train-test data split. Because decision trees handle missing data well, I did not impute or filter out missing data. Based on the confusion matrix below, we see that the tree was able to predict test values with a near 100% accuracy.

	Poisonous	Safe
Poisonous Predicted	859	0
Safe Predicted	0	766



4. **Problem 4:** Consider two classifiers A and B. On one data set, a 10-fold cross validation shows that classifier A is better than B by 3% , with a standard deviation of 7% over 100 different folds. On the other data set, classifier B is better than classifier A by 1% , with a standard deviation of 0.1% over 100 different folds. Which classifier would you prefer on the basis of this evidence, and why?

   I would prefer classifier B over classifier A. Even though classifier A's improvement is 2% improvement, on average, greater than B's, classifier's A distribution is wider, suggesting less precision and less reliability each time. Additionally, since the accuracy of B is less than 1 SD away from A's accuracy, classifier B is not sacrificing much to reduce variance.

5. **Problem 5:** Discuss the advantages and disadvantages of a nearest neighbor classifier, over a decision tree.

Both nearest neighbor (KNN) and decision tree (DT) methods are non-parametric, which means the inference does not require an underlying distribution of the data to be assumed. KNN can produce boundaries of arbitrary shapes, whereas DTs can only produce rigid, right-angled boundaries. That being said, KNN are sensitive to missing value and noise and requires all data to be numeric and scaled to prevent one attribute from dominating the others, where as DTs are robust to noise, are inexpensive to construct, and allow for non-numeric data. That being said, DTs are flawed because space is exponentially large, greedy approaches often fail to find best tree, DTs don't handle interactions, and each boundary can only involve a single attribute at a time.

6. **Problem 6:** Explain how to implement the Bayes classifier for d-dimensional numeric training data, in which for all dimensions $i = 1$ to d, the probability density of data instance X in class i is proportional to $e^{-||X-\mu_i||_1}$, where $||\ ||_1$ is the Manhattan distance, and mean μ_i is known for each class.

   When we are using the Bayes Classifier, we are hoping to predict the class that some attribute C belongs to using conditional probabilities, that is

$$P(C = c|x_1 = a_1 \dots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c)$$

   Instead of assuming the Gaussian probability distribution for $P(x_j = a_j|C = c)$, as we do in the naïve bayes classifier, we can instead use the probability density function we are given.

$$\begin{aligned} P(C = c|x_1 = a_1 \dots x_d = a_d) &\propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c) \\ &= P(C = c) \prod_{j=1}^{d} e^{-||X-\mu_i||_1} \end{aligned}$$

7. **Problem 7:** How would you solve the Problem 6 if mean μ_i is unknown for a certain dimension i?

   If we do not know the true mean μ_i, we can estimate it by using the sample mean, $\bar{x}_i$. Then,

$$\begin{aligned} P(C = c|x_1 = a_1 \dots x_d = a_d) &\propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c) \\ &= P(C = c) \prod_{j=1}^{d} e^{-||X-\bar{x}_i||_1} \end{aligned}$$