Decision Tree Quiz

1. Calculate Entropy and Gini Impurity

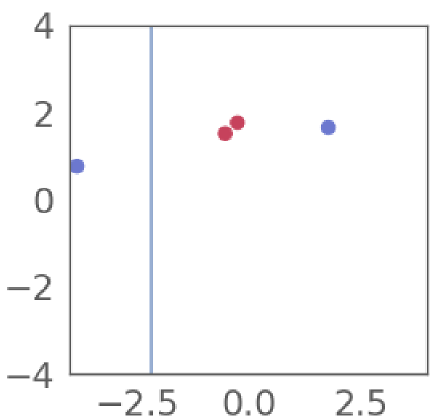
Suppose you have a biased 6-sided die, where there is a $\frac{1}{4}$ chance to roll a 1 and all other numbers have an equal chance to be rolled. What is the entropy and the Gini impurity of this die? No need to simplify the logs.

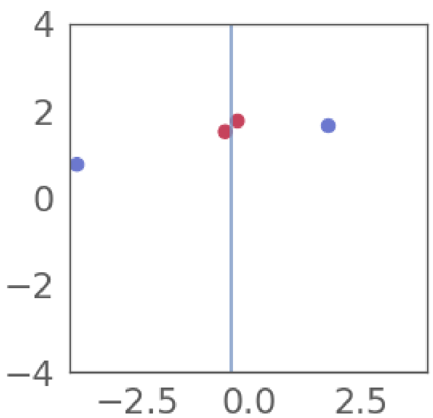
2. Given enough splits, can you classify any dataset using a decision tree with 100% accuracy?

a. Yes, because decision trees are deterministic
b. Yes, because given enough splits we can always uniquely identify a data point
c. No, because decision trees are probabilistic
d. No, because points with the identical features may belong to different classes

```

``` #### 3. Calculate the information gain for each choice data split (leave your answer in terms of natural logs). Which choice of threshold produces a greater information gain?

a. 

b. 

4. Entropy upper bound

Let X be a random variable with discrete outcomes $\{x1, x2, ..., xk\}$. We denote the probability mass function as $p(X)$. That is, for a specific outcome xj, the probability that $X = xj$ is $p(X = xj)$. Recall entropy is defined as,

$$H(X) = -\sum_{j=1}^{k} p(X = xj) \ln p(X = xj)$$

1. Show that $H(X) = \mathbb{E}[-\ln p(X)]$. Use the fact that $\mathbb{E}[g(X)] = \sum_{j=1}^{k} p(X = xj) g(xj)$ for discrete outcomes

2. Given that $g(x) = \ln(x)$ is a concave function, and the Jensen inequality which states for a concave function $g(X)$,

$$\mathbb{E}[g(X)] \leq g(\mathbb{E}[X])$$

Find an upper bound for $H(X)$, simplify as much as possible.

3. For what distribution of X is $H(X)$ equal to its upper bound?

5. Select which type of decision tree is MOST LIKELY to overfit:

a. Small tree
b. Large tree
c. Both are equally likeley

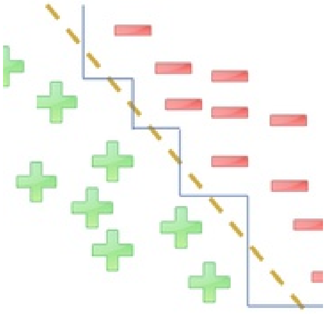
6. Using the truth table below, construct a decision tree using the minimum number of a layers.

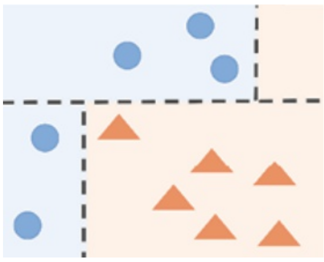
Hint: given A is true does the label depend on C? Given A is false, does the label depend on B?

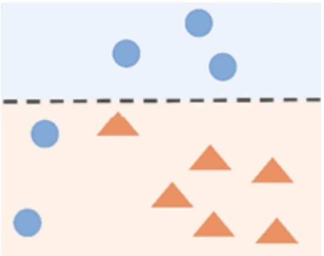
A	B	C	Label
T	T	T	T
F	T	T	T
T	T	F	T
F	T	F	F
T	F	T	F
F	F	T	T
T	F	F	F
F	F	F	F

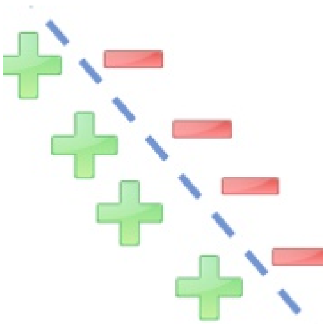
7. Which classification boundary(s) could NOT be from a decision tree?

IGNORE dashed yellow line in (a)

a.  Source: 11/16 lec Prof. J. Listgarten, CS189

b.  Source: 11/18 lec Prof. J. Listgarten, CS189

c.  Source: 11/18 lec Prof. J. Listgarten, CS189

d.  Source: 11/16 lec Prof. J. Listgarten, CS189

8. Choose all FALSE statements about information gain?

a. Knowing more information cannot decrease your current knowledge of a random variable
b. Adversaries can cause negative information gain because they can use information against you
c. The information gain between two random variables is zero if and only if the two variables are independent.
d. In the recursive DT algorithm, splitting on the feature with the largest information gain is equivalent to splitting on the feature with the lowest entropy.

9. Fill in the missing pseudo-code for the base cases in the pseudo code for the `DECISION-TREE-LEARNING` function:

```
"""data_set is a nxk matrix for the n data samples at the current node, and outcomes
is a list of known outcomes for each data sample. Assume that unique(list) is a
function that returns the number of unique
objects in a list. Let majority_rule(list) be
a function that returns the object in a list
with the greatest occurance. """

function DECISION-TREE-LEARNING(data_set, outcomes)
 #create a new tree
 tree = new node()
 #base case 1
 if unique(outcomes) == 1
  tree.set_label(YOUR ANSWER PART A)
  return tree
 #base case 2
 else if unique(get_features_list(data_set)) == 1
  tree.set_label(YOUR ANSWER PART A)
  return tree
 else
  #select feature that maximizes information gaing
  best_feature = argmax(information_gain)
  for value v in best_feature:
   indices = [index where feature_value(data, best_feature) == v]
   subDataSet = data_set[indices]
   subOutcomes = outcomes[indices]
   subtree = DECISION-TREE-LEARNING(subDataPoints, subOutcomes)
   tree.add_child(subtree)
  return tree
```

a.
b.

10. The above Decision-Tree Learning algorithm is:

a. Optimal only
b. Complete only
c. Both optimal and complete
d. Neither optimal nor complete