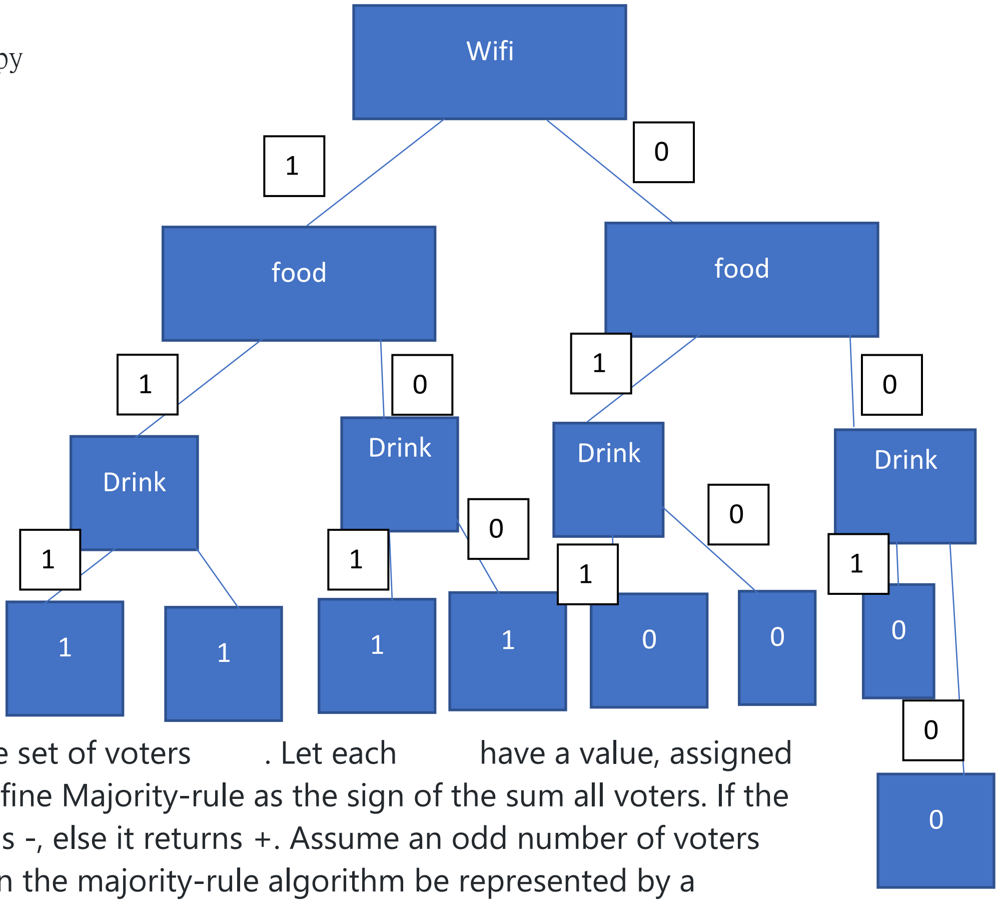
1. Assume you have a deterministic function that takes a fixed, finite number of Boolean inputs and returns a Boolean output. Can a decision tree be built to represent any such function? Give a simple proof for your answer.

   Yes

   Happy(food, drink, wifi) = 1 or 0

Food	drink	wifi	happy
1	1	1	1
1	0	1	1
1	0	0	0
1	1	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0



2. Let be a voter in the set of voters . Let each have a value, assigned to either -1 or 1. Let's define Majority-rule as the sign of the sum all voters. If the sum is negative, it returns -, else it returns +. Assume an odd number of voters (so there are no ties). Can the majority-rule algorithm be represented by a decision tree that considers a single voter at each decision node? Why or why not?

   Yes.

   If we build a tree based on voters' value.

   When we traverse from root to leaf, in the meanwhile we can count the sum. Therefore, when we arrive the leaf, we know the sum and we can tell it is + or – based on the sum.

3. In the coding section of this assignment, you trained a decision tree with the ID3 algorithm on several datasets (`candy-data.csv`, `majority-rule.csv`, `ivy-league.csv`,

and `xor.csv`). For each dataset, report the accuracy on the testing data, the number of nodes in the tree and the maximum depth (number of levels) of the tree.
My code has bugs. Unfortunately, I have to skip this question.

4. What is the bias of the ID3 algorithm in the way it searches the hypothesis space of possible decision trees?

   A closer approximation to the inductive bias of ID3: shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

   ID3 searches a complete hypothesis space. It searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met. Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy. Its hypothesis space introduces no additional bias.

5. What is the the thing that `majority-rule.csv` and `xor.csv` have in common? How does this thing interact with the bias in the way ID3 builds a decision tree?
   No matter in majority-rule.csv or xor.csv, the information gain of all attributes at all levels are the same. ID3 always choose the shorter tree by greedy algorithm. ID3 will always choose the first attribute.

6. Explain what overfitting is and describe how one can tell it is happening.

   Given a hypothesis space H, a hypothesis h belongs to H is said to overfit the training data if there exists some alternative hypothesis h' belongs to H, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

7. Explain how Reduced Error Pruning is done, the reason for doing it, and what the tradeoffs are in applying Reduced Error Pruning.

   Consider each of the decision nodes in the tree to be candidates for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set. This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set. Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set. Pruning of node continues until further pruning is harmful of the tree over the validation set.

The major drawback of this approach is that when data is limited, withholding part of it for the validation set reduces even further the number of examples available for training.

8. One can modify the simple ID3 algorithm to handle attributes that are real-valued (e.g. height, weight, age). To do this, one must pick a split point for each attribute (e.g. height > 3) and then determine information gain, given the split point. How would you pick a split point automatically? Why would you do it that way?

   We would like to pick a threshold, c, that produces the greatest information gain. By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding value of A.it can be shown that the value of c that maximizes information gain must always lie at such a boundary. These candidate thresholds can then be evaluated by computing the information gain associated with each.

9. Assume each person in a population is has two real-valued measured attributes: height, weight. In a two-dimensional plot, illustrate the decision boundary line for

   the concept . Those with get a 0, those with get a 1. Now, sssume you have a decision tree that uses a single real-valued attribute (plus an ideally-chosen split point value) at each decision node. Can you represent the

   concept with such a tree? In other words, you want to always return 0 if someone's weight is greater than their age and 1 otherwise. Can you do it? If so, specify the decision tree. If not, say why not.

   No, we can't. we only have information of height and weight and we don't have the relationship between age and height or age and weight. Therefore, we can't use height and weight to determine if weight is greater than age.

10. Ensemble methods are learning methods that combine the output of multiple learners. The hope is that an ensemble will do better than any one learner, because it reduces the overfitting a single learner may be suceptible to. One of the most popular ensemble methods is the random forest. The high level idea is to build multiple decision trees from the training data. The way one builds different decision trees from the same data is to train decision trees on different

random subsets of the attributes. If, for example, there are 20 measurable attributes, you randomly pick 10 of the attributes and build a tree on those 10, then you randomly pick another 10 attributes and build a tree using those 10

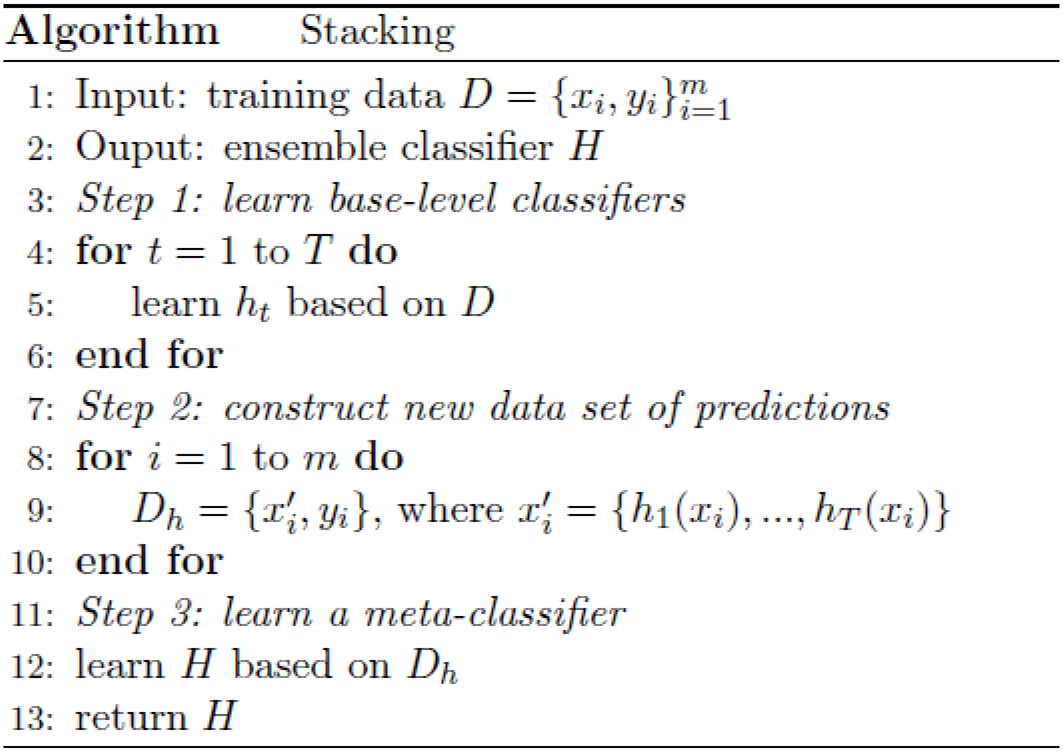
attributes. If you were building an ensemble of trees this way, how would you combine their decisions in the end? Explain why you would choose this method. Feel free to provide a citation for your choice (if you cite something, please ALSO provide a hyperlink), but also explain the reason this is your choice.

We can use stacking.

Stacking

Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

The base level often consists of different learning algorithms and therefore stacking ensembles are often heterogeneous. The algorithm below summarizes stacking.



```
Algorithm Stacking
Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier $H$
Step 1: learn base-level classifiers
for $t = 1$ to $T$ do
    learn $h_t$ based on $D$
end for
Step 2: construct new data set of predictions
for $i = 1$ to $m$ do
    $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
end for
Step 3: learn a meta-classifier
learn $H$ based on $D_h$
return $H$
```

The stacking consists of k-NN, Random Forest, and Naive Bayes base classifiers whose predictions are combined by Logistic Regression as a meta-classifier.

Stacking achieves higher accuracy than individual classifiers and based on learning curves, it shows no signs of overfitting.

This is my choice because it trains a lot of model based on different training data, and using these models to produce training data for our final model. Therefore, I think this makes good use of all feature of Ensemble methods.

Ref: https://blog.statsbot.co/ensemble-learning-d1dcd548e936