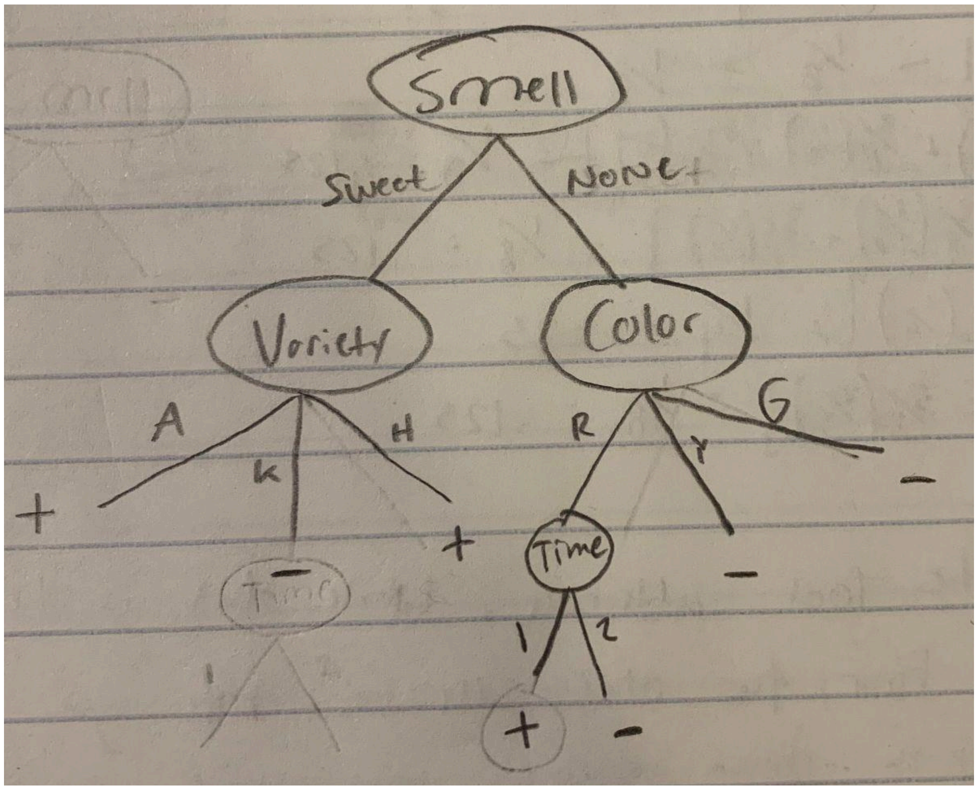
CS5350_Hw1

Part 1 : Decision Trees

1.
   a. There are 36 possible functions to map the four features to a boolean decision.Of those, <write a function to list all the possible functions>, then analyze.<probs just write a function to see which ones worked as well>
   b. 1
   c. Variety: .156, Color: .061, Smell: .189, Time: .049
   d. Smell
   e. See graph



   f. ⅔.

2.
   a. Gain(S,A) = MajorityError(S) - Sum (for each v in Values(A)) { (|Sv|/|S| Majority(Sv) }
   b.

Feature	Information Gain (using majority error)
Variety	.125
Color	.25
Smell	.125
Time	.125

c. Smell should be the root attribute. While we reach that same conclusion and majority error, the other attributes info gain using majority error is the same, which doesn't give us any indication as for how to construct the rest of the tree. Entropy has different values for info gain for each attribute so we'd end up with a different tree.

Part 2 : Experiments

1. So e is the most common label in the training data. If a classifier always picks this label (e), the accuracy would be 3382/6530 = 51.8%. For the test data, a classifier that always picks e would be 826/1594 = 51.8%.
2. ID3 algorithm explained- I first determine what the attribute is with the best information gain and use that as the current root node. For each of the possible values of the attribute we make a branch connecting to another node. The node being branched to will have either the attribute of the next best information gain attribute or be given the majority label if there are no more attributes with info gain (aka the attribute has e or p = 0).
   Design choices/data structures: the ID3 algorithm uses bestAttribute which eventually ends up using label_counts, col_label_counts, entropy, col_entropy, info_gain. The main design choice other than separating code was the vals_dict in col_label_counts. The vals_dict contains counts of e and p for each possible value of the selected column. I then use that dictionary in col_entropy to calculate entropy for the column.
   a. Spore-print-color
   b. .485
   c. 8
   d. .983
   e. .982
3.
   a.

```
Cross-validation with depth limit: 1 ----------
   Average accuracy: 0.5283307810107198
   Standard deviation: 0.22324756072455645
Cross-validation with depth limit: 2 ----------
   Average accuracy: 0.8214395099540581
   Standard deviation: 0.17127559162968609
Cross-validation with depth limit: 3 ----------
```

```
    Average accuracy: 0.9514548238897398
    Standard deviation: 0.09611299806653004
Cross-validation with depth limit: 4 ----------
    Average accuracy: 0.9655436447166922
    Standard deviation: 0.08157096893368165
Cross-validation with depth limit: 5 ----------
    Average accuracy: 0.9644716692189894
    Standard deviation: 0.08278413917282185
Cross-validation with depth limit: 10 ----------
    Average accuracy: 0.9638591117917304
    Standard deviation: 0.08346822677856157
Cross-validation with depth limit: 15 ----------
    Average accuracy: 0.9638591117917304
    Standard deviation: 0.08346822677856157
```

From this data, we see that the most accurate cross validation is with a depth limit of 4 which gives 96.55% accuracy. This makes sense as lower depth limits do not generate specific enough trees and greater depth limits start to overfit the data slightly. The accuracy levels out after depth limit 4.

b. When iterating through the possible values for an attribute, I don't add a branch unless the current depth is less than the depth limit. I pass current_depth+1 to the recursive ID3 call when branches are added.
c. best_tree_accuracy: 0.972396486825596
d. The full decision tree appears to be more accurate than the pruned one, which makes me think its possible I did something wrong/my ID3 algorithm is slightly different in the depth limiting version.. However, the pruned tree is half the depth of the full tree, and is almost as accurate.It would make sense that a pruned tree would be more accurate to avoid overfitting and also takes up much less space.