1 Naive Bayes

1. \( \hat{C} = \text{argmax}_C (P(C|G, H)) = \text{argmax}_C (\frac{P(G|C)P(H|C)P(C)}{P(C,G,H)}) \)

2. \( P(C=\text{Oak}) = P(C=\text{Pine}) = 1/2 \)
   \( P(G=\text{Small}|C=\text{Oak}) = 1/4 \)
   \( P(G=\text{Large}|C=\text{Oak}) = 3/4 \)
   \( P(G=\text{Small}|C=\text{Pine}) = 3/4 \)
   \( P(G=\text{Large}|C=\text{Pine}) = 1/4 \)
   \( P(H=\text{Hard}|C=\text{Oak}) = 1/2 \)
   \( P(H=\text{Soft}|C=\text{Oak}) = 1/2 \)
   \( P(H=\text{Hard}|C=\text{Pine}) = 1/2 \)
   \( P(H=\text{Soft}|C=\text{Pine}) = 1/2 \)

3. \( P(C=\text{Oak}|\text{Small, Hard}) \propto \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{1}{2} \)
   \( P(C=\text{Pine}|\text{Small, Hard}) \propto \frac{3}{4} \cdot \frac{1}{2} \cdot \frac{1}{2} \)
   \( P(C=\text{Oak}|\text{Small, Hard}) = 1/4 \rightarrow \text{Pine} \)
   True label is also Pine
   0/1 loss = 0
   sq. loss = \((1/4)^2 = 1/16 \)

2 Decision Rules

2.1 Construct a set of decision rules

We’d only mentioned decision rules briefly, and noted how they could be extracted from a decision tree. One way to do this would be to learn a decision tree, then take each path from root to a leaf and write it down as a decision rule (e.g., \( A_1 \land \neg A_2 \land A_3 \land A_4 \Rightarrow Y \)).

But since you’ll be learning a decision tree in the next question, let’s think about how to do this directly. First we could identify any attributes that lead to a decision 100% of the time. For example, \( A_1 = 1 \Rightarrow Y \) holds. We have now covered \( x_1, x_2, \) and \( x_5 \). We now look for rules with two attributes that cover the remaining cases. We note that \( A_3 \land A_4 \Rightarrow Y \) (covering \( x_2 \) again, and \( x_7 \)), and \( \neg A_3 \land \neg A_4 \Rightarrow Y \), we cover \( x_3 \) (and \( x_2 \) again). We’ve now covered all our \( Y = 1 \) cases, so we can add a default rule of “if no other rule applies, \( Y = 0 \)” and cover all cases.

Because there are fewer cases of \( Y = 0 \), it would probably be better to start with rules that identify those cases. But if you inspect, you’ll see that this gives more complicated rules.
2.2 Precision and Recall

From the first rule, we get $x_{11} = 1$ (true positive)). The second rules gives $x_{10} = 1$ (false positive). The third rules gives $x_9 = 1$ (true positive) and $x_{12} = 1$ (false positive).

$$\text{Precision} = \frac{\text{TruePos}}{\text{FalsePos} + \text{TruePos}} = \frac{2}{2+2} = 0.5$$
$$\text{Recall} = \frac{\text{TruePos}}{\text{TruePos} + \text{FalseNeg}} = \frac{2}{2} = 1$$

Not very good. You need a lot more data to expect to learn good rules...

3 Decision Trees

3.1 Build a Naive Bayes Classifier

$$P(Z=0) = 7/16$$
$$P(Z=1) = 9/16$$
$$P(X=0|Z=0) = 4/7$$
$$P(X=0|Z=1) = 4/9$$
$$P(Y=0|Z=0) = 3/7$$
$$P(Y=0|Z=1) = 4/9$$

The remaining four conditional probabilities can be found by taking complement of above conditional probabilities.

3.2 Build a Decision Tree

If (X=1 and Y=1) or (X=0 and Y=0), then Z=1.

3.3 Evaluate, and compare

If you’ve used the entire dataset to build the models above, then you can only evaluate on the training data, which is not a good measure of how well this will work on unseen data. The proper approach would be to split the dataset into training set and test set. Then compare the accuracy, precision, recall and F-1 score of both methods. (For this part, the approach used could vary as students may divide their dataset in different ways. But you should recognize that the decision tree is better.)

3.4 Why?

Decision tree is better than Naive Bayes as it is able to model the non-linear decision boundaries which a Naive Bayes classifier can’t do as it is linear. The given data distribution resembles a XOR function which can’t be linearly separated with a single decision boundary. Hence, decision trees will be better due to the ability to generalize over non-linearities in data.