Chapter 1: Decision Trees

1.1 The Decision Tree Model Of Learning

Decision tree is a natural model of learning, but specifically, we will be talking about **binary classification** where you only have two choices: yes or no.

You need to ask questions and based on user responses, you make predictions. The questions that you can ask is called **features** and the responses to these questions are **feature values**. An **example** is just a set of feature values.

To start, first you need to look at the **histogram** for each feature. Each feature is supposed to have yes or no. Each yes and no is supposed the percentage of the time it guessed correctly. For example, the feature of "course AI" is supposed to show "yes" guessing 70 percent of the time right and 30 percent it guessed wrong. This means that when you pick "yes", you will guess 70 percent of the time that the algorithm will guess correctly and 30 percent otherwise and that is based on the training set.

So, if you had a feature where if they said no to the feature, they would hate the course. In the case of the AI course, if you pick that you hated AI course, according to the histogram, 100 percent of the time, it will say that you will not enjoy computer systems. However, if you said that you enjoyed AI, 80 percent of the time will say that you will enjoy computer systems, meaning 20 percent it will incorrectly say that you will enjoy computer systems when you do not.

Thus, the error is 18/20, and let's say this is highet among all features. Thus, you pick this feature among the others. You will repeat seperately on the YES part and the NO part until it won't make sense to go on.

But now, you must compute the error for your model that you just created, which is why we have the **loss function** of two arguments, the actual value y and the predicted value $\hat{y}$.

For regression, we have the **squared loss**: $l(\hat{y}, y) = (y - \hat{y})^2$ and then we have the **absolute loss**: $l(\hat{y}, y) = |y - \hat{y}|$.

For binary classification as well as multiclass classification, we have 0 for predicting right and 1 for predicting wrong.

Now, we have the **data generating distribution** D which defines what sort of data we expect to see.

Based on the training data, we are expected to **induce** a function f that maps new inputs $\hat{x}$ to corresponding prediction $\hat{y}$. We also expect our function to do well on future examples that are also sampled from D. Therefore, we should minimize **expected loss** as small as possible.

We also have **training error** which is simply the average error over the *training set*. The expected loss, I think, goes over the test set.

Therefore, we have the following summary: Given (i) a loss function and (ii) a sample D from some unknown distribution D, you must compute a function f that has low expected error over D with respect to the loss function.