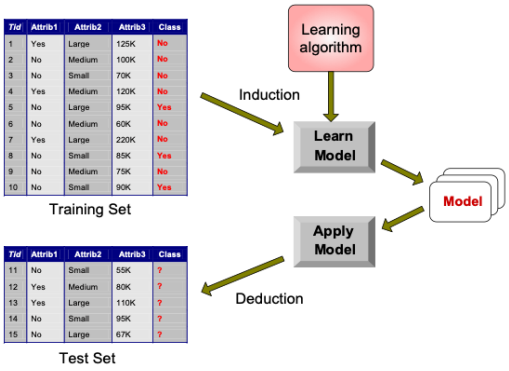
Classification and Decision Trees Activity

Activity 1 : Supervised Learning

Given a set of example data, which we will call *training data*, the goal is to learn a function/model that will compute a target value for new data. This is known as **supervised** learning. An example of supervised learning is predicting the sales price of a home. Example data for this task is shown in the table below.

SQFT	# of bedrooms	zip code	Sales price
1000	2	22999	160,000
2000	4	20111	450,000
2200	3	90210	850,000



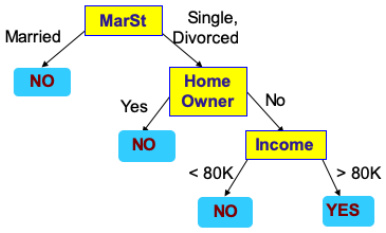
Introduction to Data Mining. Pang-Ning Tan et. al. 2019.

When the predicted value is a real number (sales price), we call this *regression*. When the predicted value is a label, this is called *classification*, for example, identifying an email as **spam** or **not spam**. When only two labels are possible (as is the case with spam), this is called *binary classification*. A general approach to classification to shown on the left.

1. List 3 possible uses of **regression**.

2. List 3 possible uses of **classification**.

Activity 2 : Tree Structures for Classification



A tree structure can be created to perform classification. For example, the tree on the left is used to predict whether someone who borrows money from a bank will default on the loan.

1. Use the tree to classify a newly arrived example with the following features:

Rowid	Homeowner	Marital Status	Income	Loan Will Default?
1	no	married	80K	??

Trace the pathway through the tree. What label (yes or no) does the tree assign?

Decision Tree Construction

The following is a subset of the Titanic Survivors dataset available from Kaggle `https://www.kaggle.com/c/titanic/`. Draw a decision tree that correctly classifies each item.

Pclass	Sex	Age	Survived
1	female	30	Yes
3	male	25	No
3	male	33	No
2	male	34	No
2	female	45	Yes
2	male	44	No
1	male	27	Yes
2	female	31	Yes
2	male	34	No
3	male	23	No

Now use your tree to classify the items in this test set:

Pclass	Sex	Age	Survived
1	female	17	
3	female	41	
3	male	17	
3	male	1	

Tree Reflection

In this case, it was straightforward to develop a decision tree that could perfectly classify the training set by "eyeballing" the data. Consider the case where we have thousands of elements in our training set, each with dozens of attributes.

- How could you automate the process of tree construction?

- Given that there are potentially many possible trees that fit the training data, how might you go about deciding between them? What constitutes a "good" tree?

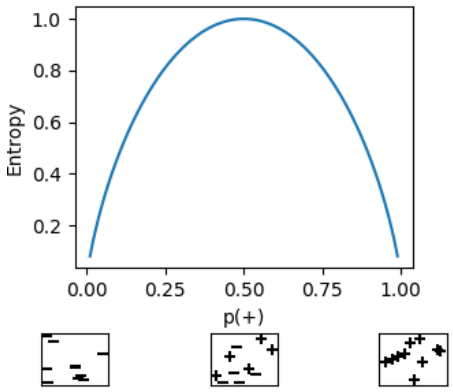
Decision Tree Construction

This type of classifier is known as a **decision tree**. How do we build this structure from training data? The goal is to build a tree that both labels the training data accurately and also performs well on data that the model has not seen. Performing well on data that was not used to build the model is important, and is known as **generalization**.

There are usually many possible trees that will perfectly classify the training data. It is impossible to know, in advance, which tree will have the best performance on unseen data points. A valuable principle here, and throughout machine learning, is **Occam's Razor**, the principle that, all else being equal, the simpler explanation is likely to be best. In the case of decision trees, the "simpler explanation" corresponds to *smaller* trees, either in terms of height, or the total number of nodes.

You may not be surprised to learn that the problem of building an optimal decision tree is NP-Hard. Following a common theme, we will build a high-quality tree using a greedy approach known as **Hunt's Algorithm**. This approach works by first grouping all the data into a single node. If all the examples in the node are from a single class, we are done. Otherwise, we iterate through each feature to see how well that feature splits/partitions the data by class, and then select the feature providing the best split (we will formalize the best split in the next activity). Child nodes are created using the best split, and the algorithm is applied recursively to those nodes. There is no guarantee that this greedy strategy will result in an optimal tree, but in practice, it works well.

Activity 3 : Decision Boundaries for Decision Trees



The greedy part of this algorithm is selecting the "best" split for partitioning the data into nodes that are as homogeneous as possible. One measure of homogeneity is *entropy*. Entropy is defined as follows:

$$-\sum_{i=0}^{c-1}\Big(p_i(t)\ \log_2\ (p_i(t))\Big)$$

Where $p_i(t)$ is the relative frequency of class i in node t. The figure on the left shows how entropy changes based on a binary classification problem of $+$ and $-$.

1. For a binary classifier, calculate the entropy of the SET (contents of a node) where 13 examples are of the class "+" and 20 examples are of class "−".

Activity 4 : Define how to Partition/Split Data

Following the example from out textbook, the metadata for the dataset is:

Feature	Domain	Type (Nominal, Ordinal, Interval, Ratio)
Homeowner	{True, False}	
Married	{Single, Married, Divorced}	
Income	[0,∞]	
Defaulted borrower	{Yes, No}	

(Fill in the table with the types of each attribute.)

Let's first consider how to enumerate the possible splits. Next, we will evaluate each potential split, with the goal of picking the "best" split.

For **nominal types**, we can evaluate a multi-way split (each nominal value has its own child) or make binary splits. **Ordinal** values can also be split this way, but care should be taken to maintain the order property (one split could be "small, medium" and another "large", but you would avoid splits of "small, large" and then "medium").

Continuous attributes are usually handled by sorting the points according to the attribute value, then using the midpoints between attribute values as possible split points.

The goal of a partitions/split is to maximize the increase in homogeneity between the parent node and its children. This is known as *gain*, or in the case of entropy, *information gain*.

$$Gain(Split) = Entropy(Parent) - \sum \frac{N(v_j)}{N} Entropy(Child) \quad (1)$$

Where $N(v_j)$ is the number of training examples within a child node and N is the is the number of examples in the parent node that is being split. Each entropy calculation is *weighted* by the number of examples it contains (and the weights sum to 1). This gain is sometimes denoted Δ_{info}.

Here is a small training set:

Home Owner	Martial Status	Annual Income	Defaulted Borrower
Yes	Single	120,000	No
No	Married	100,000	No
Yes	Single	70,000	No
No	Single	150,000	Yes
Yes	Divorced	85,000	No
No	Married	80,000	Yes
No	Single	75,000	Yes

1. For the example data provided, identify the best first split point (from a greedy perspective) and calculate the information gain from the root node (all the data) to the child nodes with respect to each candidate split point. In other words, create split points for home owner, martial status, and annual income and pick the best one (highest Δ_{info}). Show your work.
   (a) **Home Owner**

   (b) **Marital Status** (For the purposes of this example, assume a three-way split.)

   (c) **Annual Income** The table below is provided to help you keep track of the possible split points for the Annual Income attribute.

Class	No	Yes	Yes	No	No	No	Yes
	Annual Income (In thousands)						
	70	75	80	85	100	120	150

Split Points	72.5		77.5		82.5		92.5		110		135	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
Yes	0	3	1	2								
No	1	3	1	3								
Weighted Entropy	.857											

(d) What is the best overall split point and what is the information gain associated with that split?

Activity 5 : Run Time Analysis of Splitting

1. (a) Consider the run time cost of evaluating the splits for a single continuous feature like *annual income*. For a dataset of size n, how many operations does it take? You can consider the cost of computing the entropy for a single split point as $O(1)$.

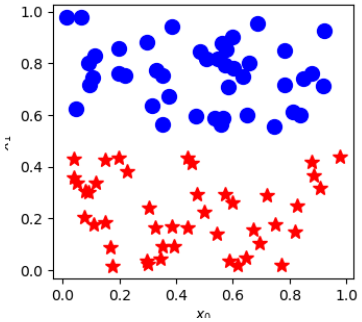
   (b) Given a dataset with d attributes, what is the cost of a determining a single split, considering all possible attributes?

   (c) What is the overall cost of the greedy decision tree construction algorithm, assuming a reasonably balanced tree?

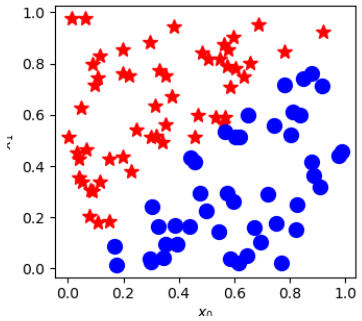
   (d) What is the worst-case cost of classifying a point using a balanced decision tree?

Activity 6 : Example Data and Applicability of Decision Trees

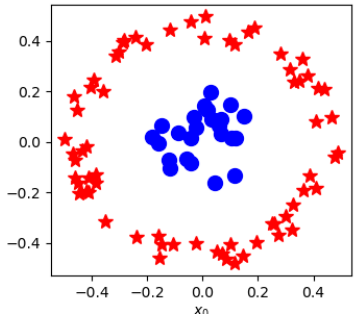
Considering the case that only *binary* splits are possible (each node can have at most two children), consider how decision trees divide up the feature space. Below are three sets of training data with 2 features. Make some general comments on how decision trees might perform given this input data.



a



b



c

Activity 7 : Quantifying Your Model's Performance

1. We will be discussing several types of classification algorithms in this class. While the accuracy and error rate of your classifier are certainly important ways to judge quality, discuss other factors that you should weigh when evaluating these algorithms.