QBUS6850 Week 6

Ensemble Methods 1 - Foundations

Dr Stephen Tierney

The University of Sydney Business School

Ensembles

An ensemble is a collection of different models.

Ensembles often have better predictive performance than a single model.

Decision Trees are the most commonly model used in ensembles.

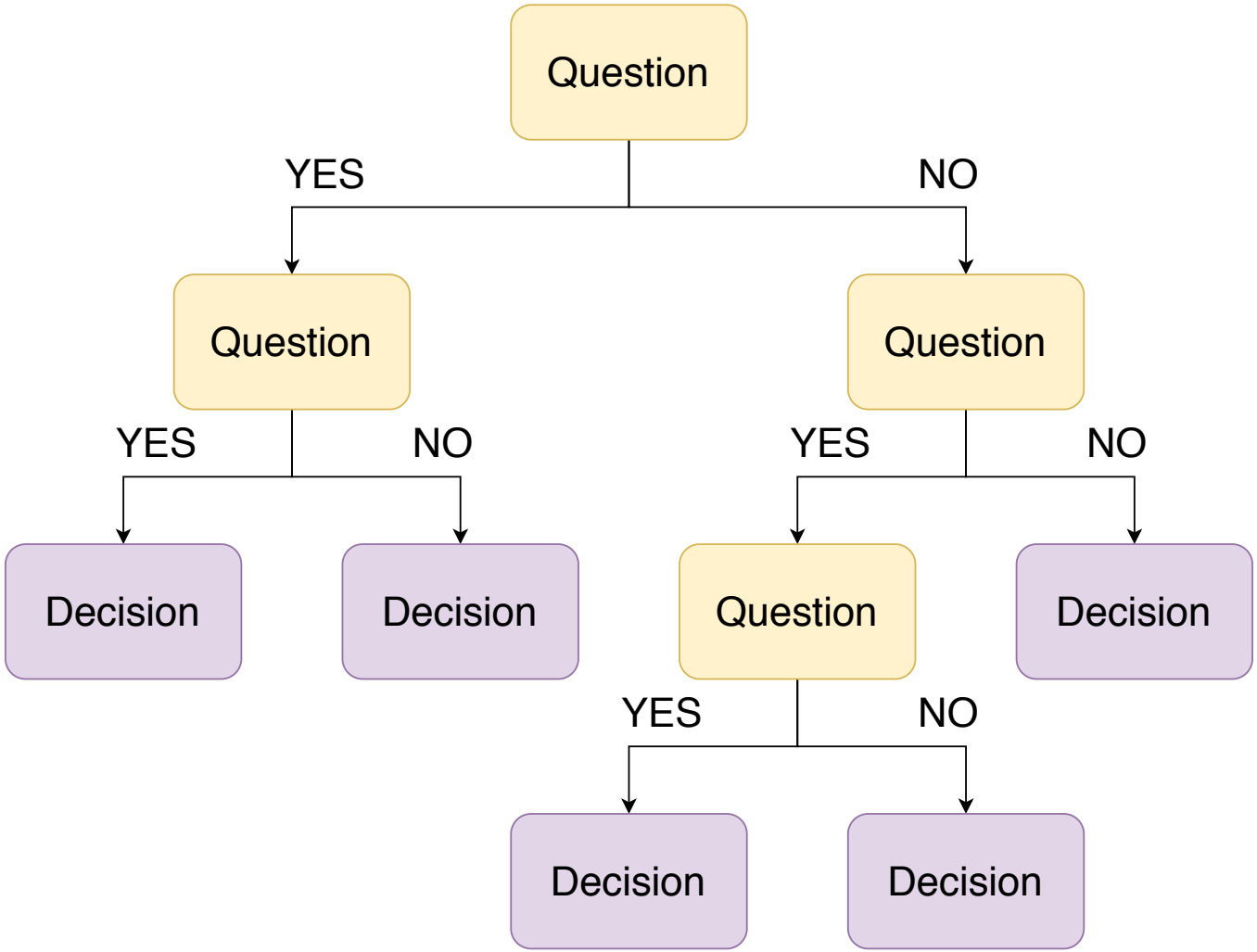
The Plan

- **Part 1 (Week 6): Understand decision trees and How to make an ensemble (forests)**
- Part 3 (Week 8): Advanced ensembles 1 (boosting)
- Part 4 (Week 9): Advanced ensembles 2 (gradient boosting)

Decision Tree

A Decision Tree is a sequence of if-then-else rules which describe a decision making process.

Decision Tree - Example



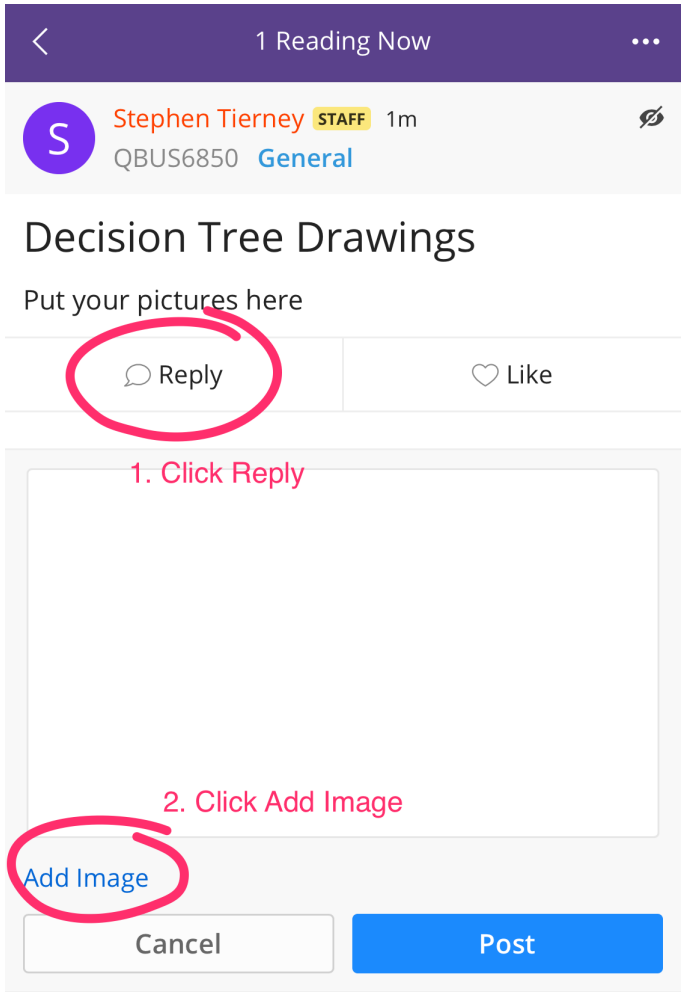
Decision Tree - Exercise

Take 5 minutes to draw your own decision tree.

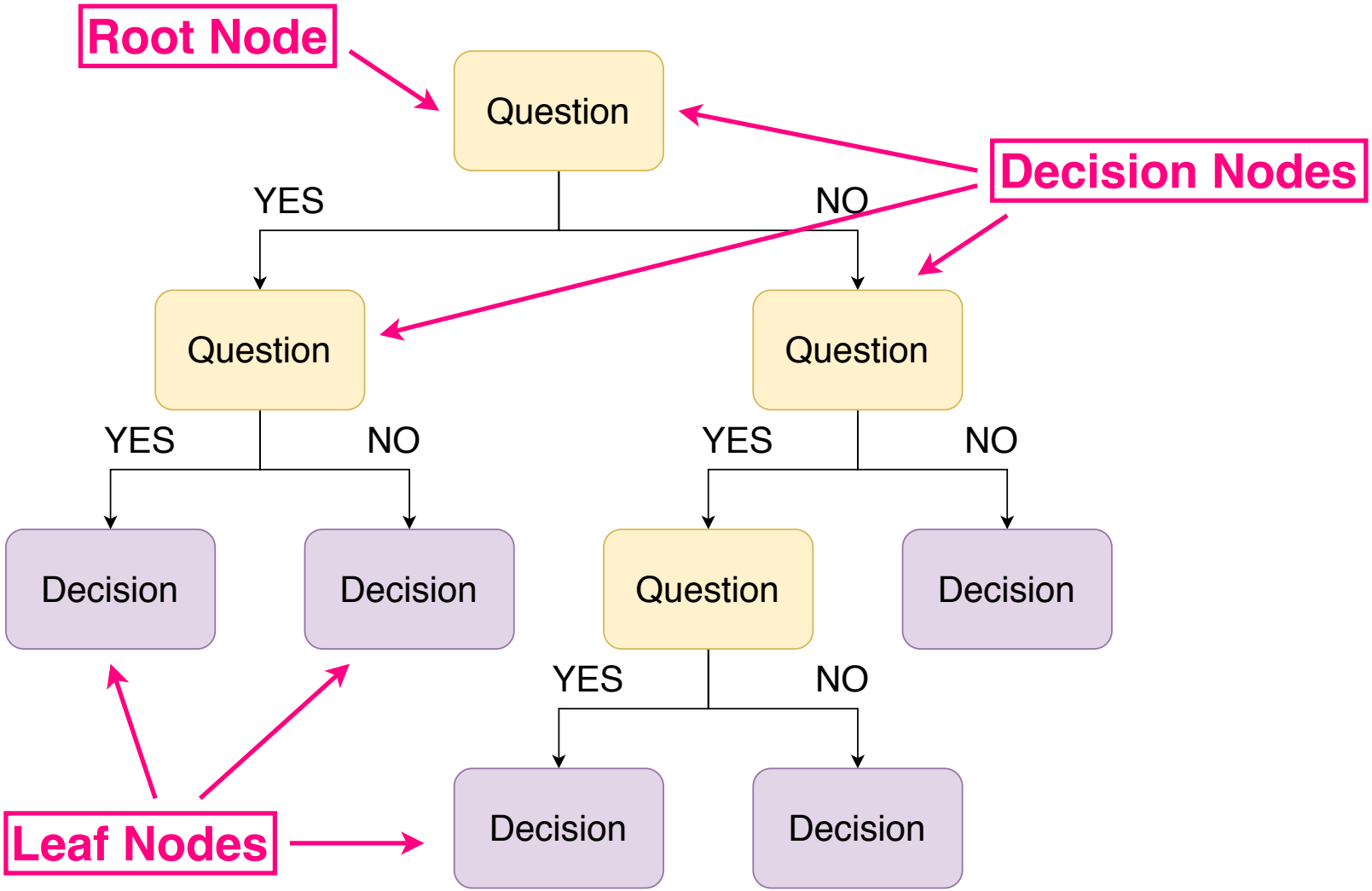
It can be about anything you like!

Take a picture with your phone and add them to the Ed Thread

https://edstem.org/au/courses/6467/discussion/582077



Decision Tree - Definitions



Classification With Decision Trees

Classification With Decision Trees

Decision Trees can be used as classifiers.

First we need to build the tree using a dataset.

Data Example

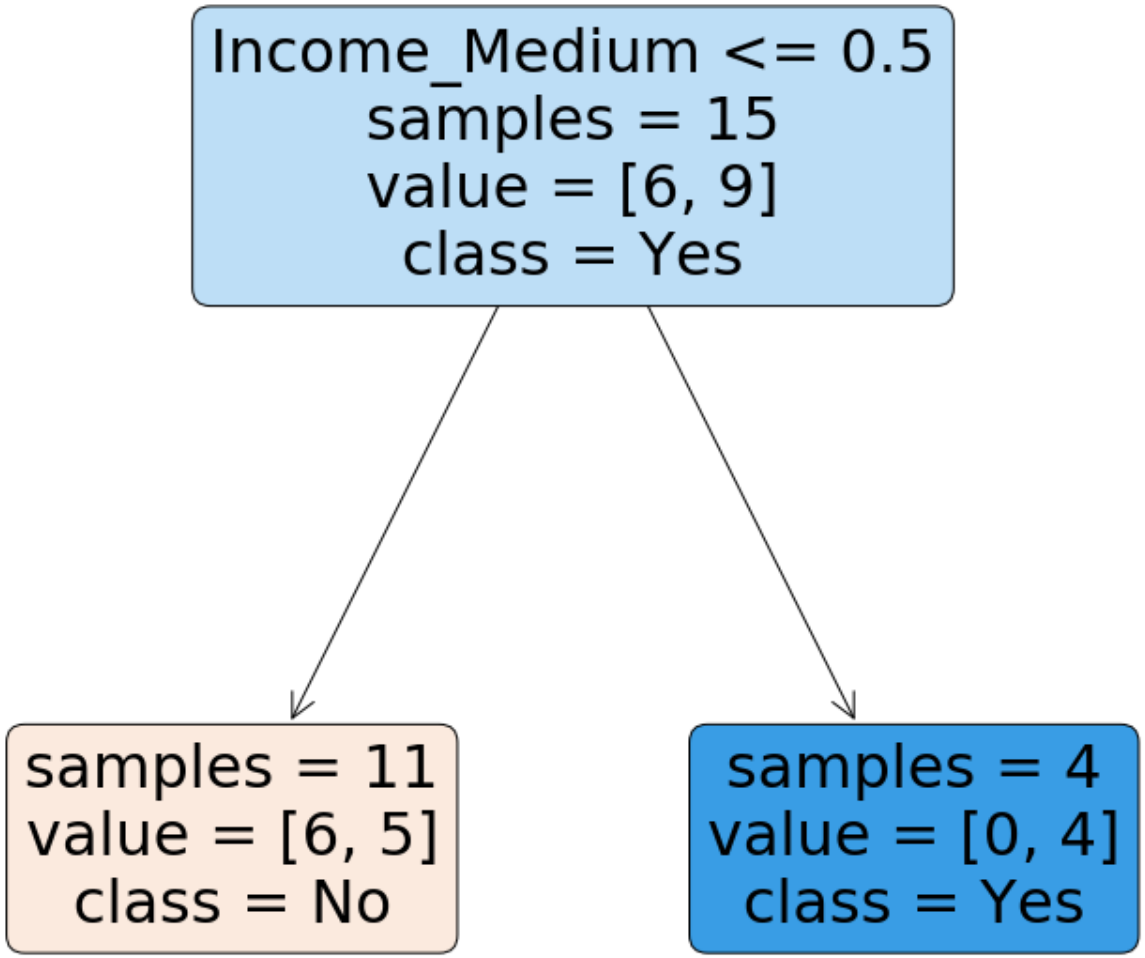
Customer	Income	Education	Marital Status	Purchase
1	Medium	University	Single	Yes
2	High	University	Single	No
3	High	University	Married	No
4	Low	University	Single	Yes
5	Low	High school	Single	Yes
6	Low	High school	Married	No
7	Medium	High school	Married	Yes
8	High	University	Single	No
9	High	High school	Single	Yes
10	Low	High school	Single	Yes
11	High	High school	Married	Yes
12	Low	University	Married	No
13	High	University	Single	No
14	Medium	University	Married	Yes
15	Medium	High school	Single	Yes

Data Example

Conversion to Dummies

Customer	Income_Low	Income_Medium	Education_University	Marital Status_Single	Purchase_Yes
1	0	1	1	1	1
2	0	0	1	1	0
3	0	0	1	0	0
4	1	0	1	1	1
5	1	0	0	1	1
6	1	0	0	0	0
7	0	1	0	0	1
8	0	0	1	1	0
9	0	0	0	1	1
10	1	0	0	1	1
11	0	0	0	0	1
12	1	0	1	0	0
13	0	0	1	1	0
14	0	1	1	0	1
15	0	1	0	1	1

Tree Example

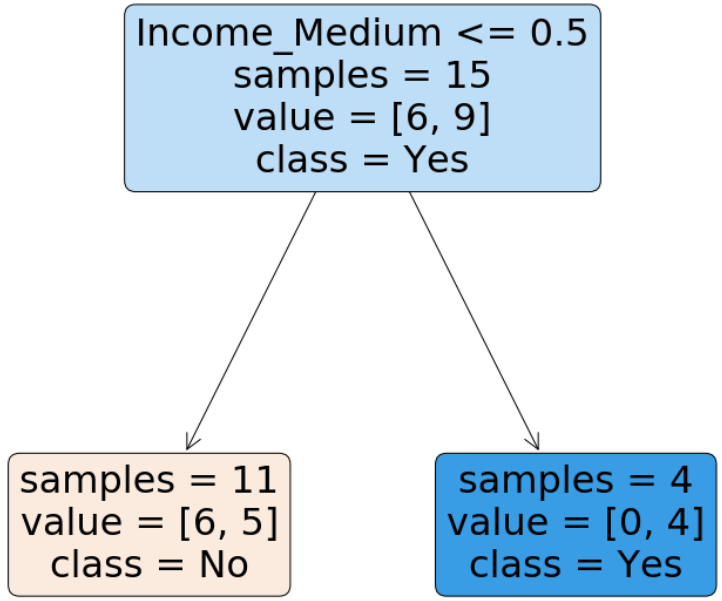


Prediction

Then to make a classification (prediction):

1. traverse the tree from top (root) to bottom (leaf)
2. find the most common class found in the leaf node

Prediction Example



Customer IDs in each leaf node

No: 2, 3, 6, 8, 12, 13

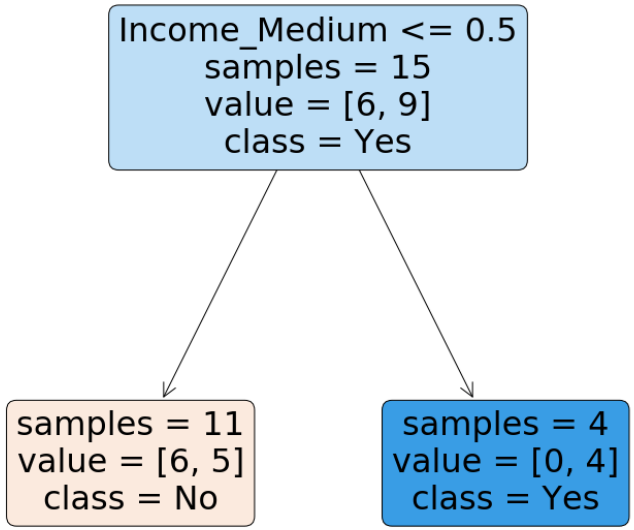
Yes: 4, 5, 9, 10, 11

Yes: 1, 7, 14, 15

Prediction Example

What is the purchase prediction for a new customer with the following attributes:

- married
- high income
- university educated



Customer IDs in each leaf node

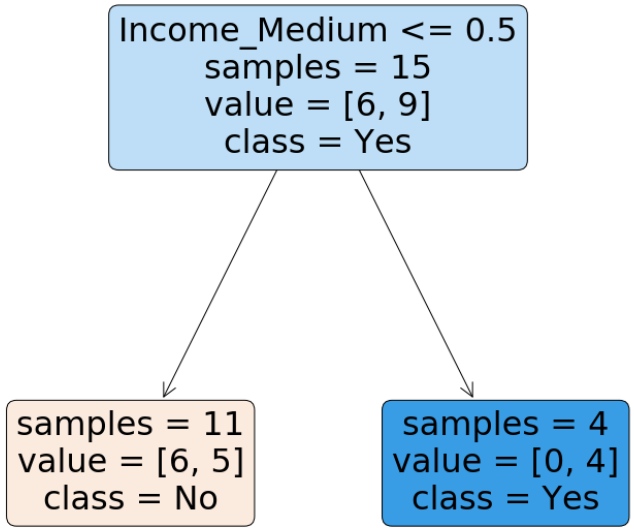
No: 2, 3, 6, 8, 12, 13
Yes: 4, 5, 9, 10, 11

Yes: 1, 7, 14, 15

Prediction Example

Answer: No Purchase.

Because we end at the left node. In the left node there are more 'No' values than 'Yes' (6 to 5 respectively).



Customer IDs in each leaf node

No: 2, 3, 6, 8, 12, 13
Yes: 4, 5, 9, 10, 11

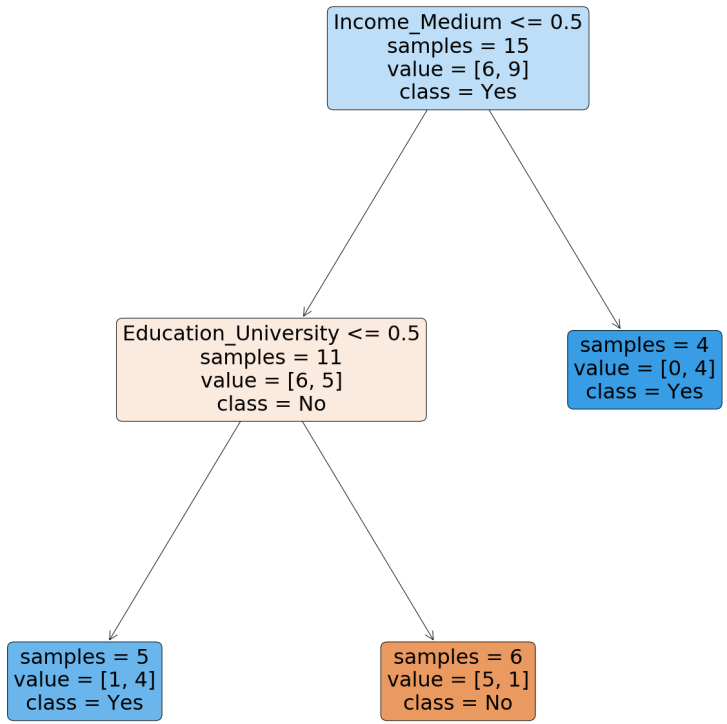
Yes: 1, 7, 14, 15

Prediction Example

Let's try again with a deeper tree.

Predict the purchase status for a new customer with the following attributes:

- married
- high income
- have university education



Building a Decision Tree - Classification

Intuition

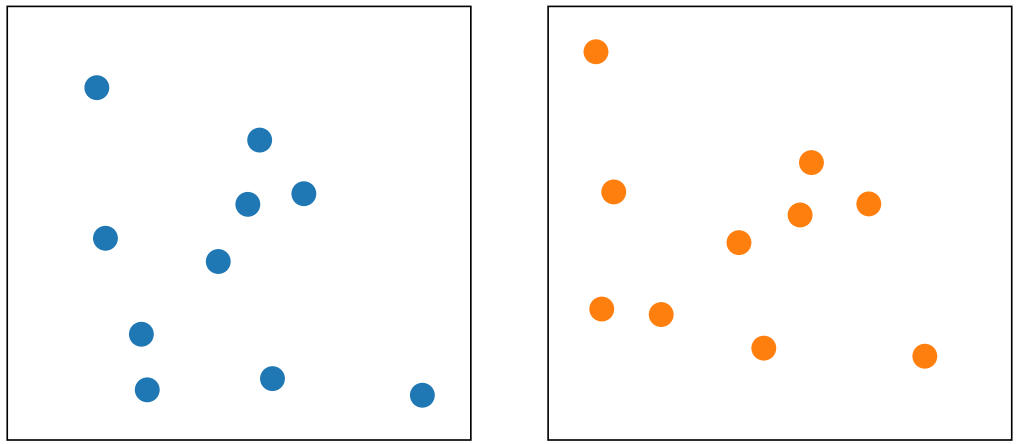
At each decision node we split our data based on one feature.

Pick the the feature that gives the **purest** leaf nodes.

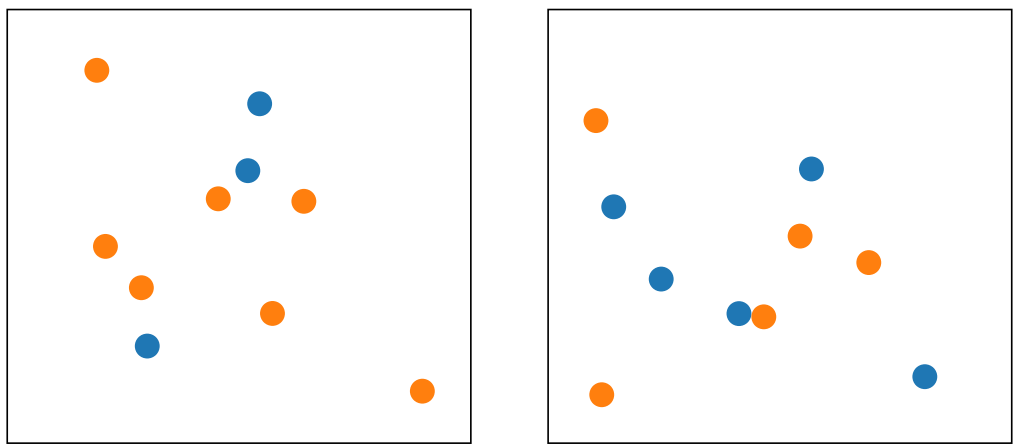
Purity

If we have two classes (blue and orange), a split might be pure, impure or somewhere in between.

Pure



Impure



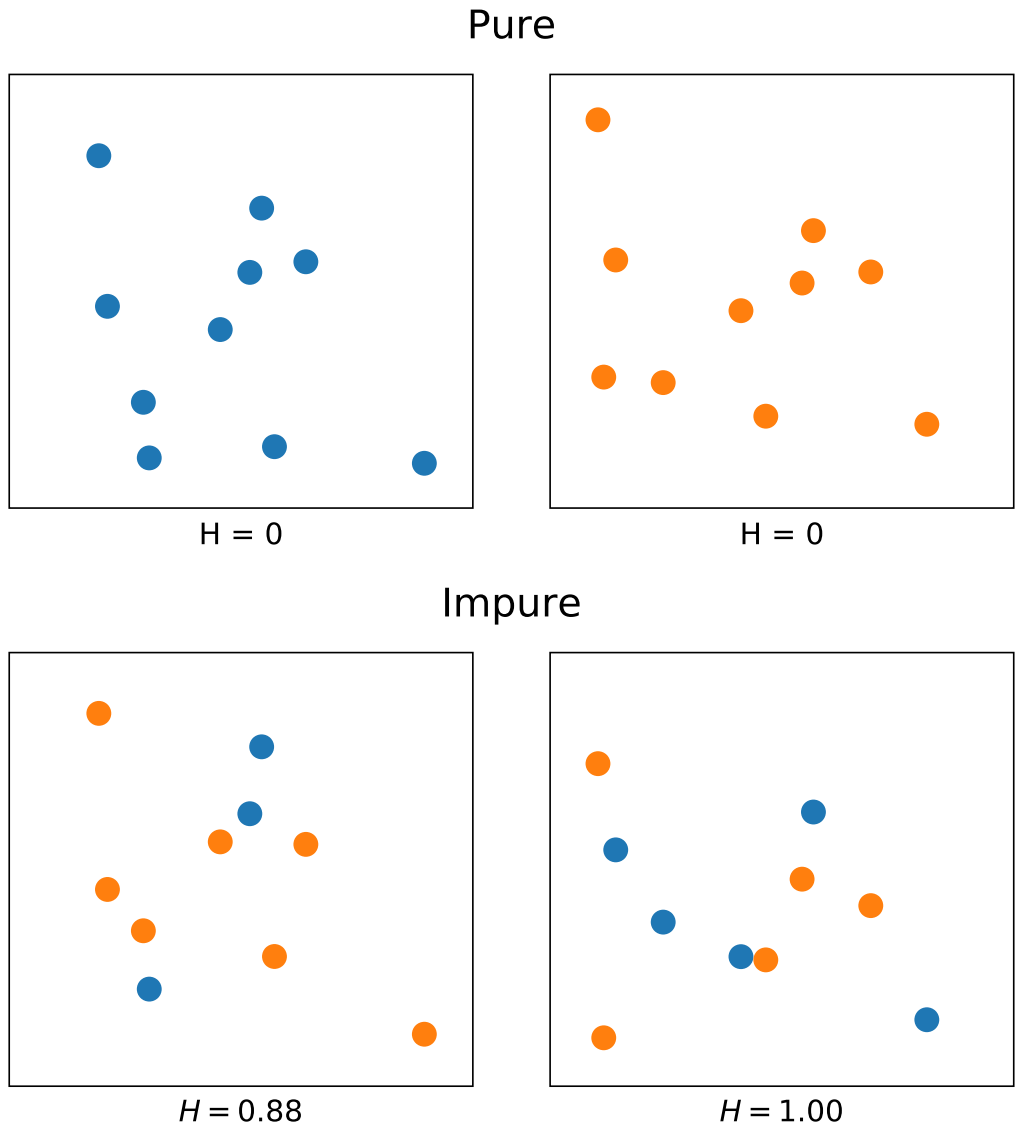
Entropy

We measure purity using **Entropy**, which we denote by H.

Entropy ranges from 0 to 1.

Smaller entropy $\mapsto$ higher PURITY
Larger entropy $\mapsto$ higher IMPURITY

Entropy



Entropy

Definition

Let $\mathcal{D}$ be a dataset with K classes, p_k be the proportion of class k in the dataset, then the entropy is

$$H(\mathcal{D}) = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

Note:

1. We calculate entropy for each leaf node
2. We are going to assume that $K = 2$ for this lecture
3. $\sum_{k=1}^{K} p_i(x) = 1$
4. $p_k \log_2(p_k) = 0$ when $p_k = 0$

Entropy Calculation Example

$\texttt{Education\_University} = 0$

Customer	Income	Education	Marital Status	Purchase
5	Low	High school	Single	Yes
6	Low	High school	Married	No
7	Medium	High school	Married	Yes
9	High	High school	Single	Yes
10	Low	High school	Single	Yes
11	High	High school	Married	Yes
15	Medium	High school	Single	Yes

$\texttt{Education\_University} = 1$

Customer	Income	Education	Marital Status	Purchase
1	Medium	University	Single	Yes
2	High	University	Single	No
3	High	University	Married	No
4	Low	University	Single	Yes
8	High	University	Single	No
12	Low	University	Married	No
13	High	University	Single	No
14	Medium	University	Married	Yes

$$H(\mathcal{D}_1) = -\frac{1}{7}\log_2(\frac{1}{7}) - \frac{6}{7}\log_2(\frac{6}{7})$$

$$H(\mathcal{D}_1) = 0.59$$

$$H(\mathcal{D}_2) = -\frac{5}{8}\log_2(\frac{5}{8}) - \frac{3}{8}\log_2(\frac{3}{8})$$

$$H(\mathcal{D}_2) = 0.95$$

Average Entropy: $\frac{H(\mathcal{D}_1)+H(\mathcal{D}_2)}{2} = 0.77$

Entropy Calculation Example

$\texttt{Income\_Medium} = 0$

Customer	Income	Education	Marital Status	Purchase
2	High	University	Single	No
3	High	University	Married	No
4	Low	University	Single	Yes
5	Low	High school	Single	Yes
6	Low	High school	Married	No
8	High	University	Single	No
9	High	High school	Single	Yes
10	Low	High school	Single	Yes
11	High	High school	Married	Yes
12	Low	University	Married	No
13	High	University	Single	No

$\texttt{Income\_Medium} = 1$

Customer	Income	Education	Marital Status	Purchase
1	Medium	University	Single	Yes
7	Medium	High school	Married	Yes
14	Medium	University	Married	Yes
15	Medium	High school	Single	Yes

$$H(\mathcal{D}_1) = -\frac{6}{11}\log_2(\frac{6}{11}) - \frac{5}{11}\log_2(\frac{5}{11})$$

$$H(\mathcal{D}_1) = 0.99$$

$$H(\mathcal{D}_2) = -\frac{0}{4}\log_2(\frac{0}{4}) - \frac{4}{4}\log_2(\frac{4}{4})$$

$$H(\mathcal{D}_2) = -0 - 0$$

$$H(\mathcal{D}_2) = 0$$

Average Entropy: $\frac{H(\mathcal{D}_1)+H(\mathcal{D}_2)}{2} = 0.49$

Entropy Calculation Example

Which variable gives us the purest split?

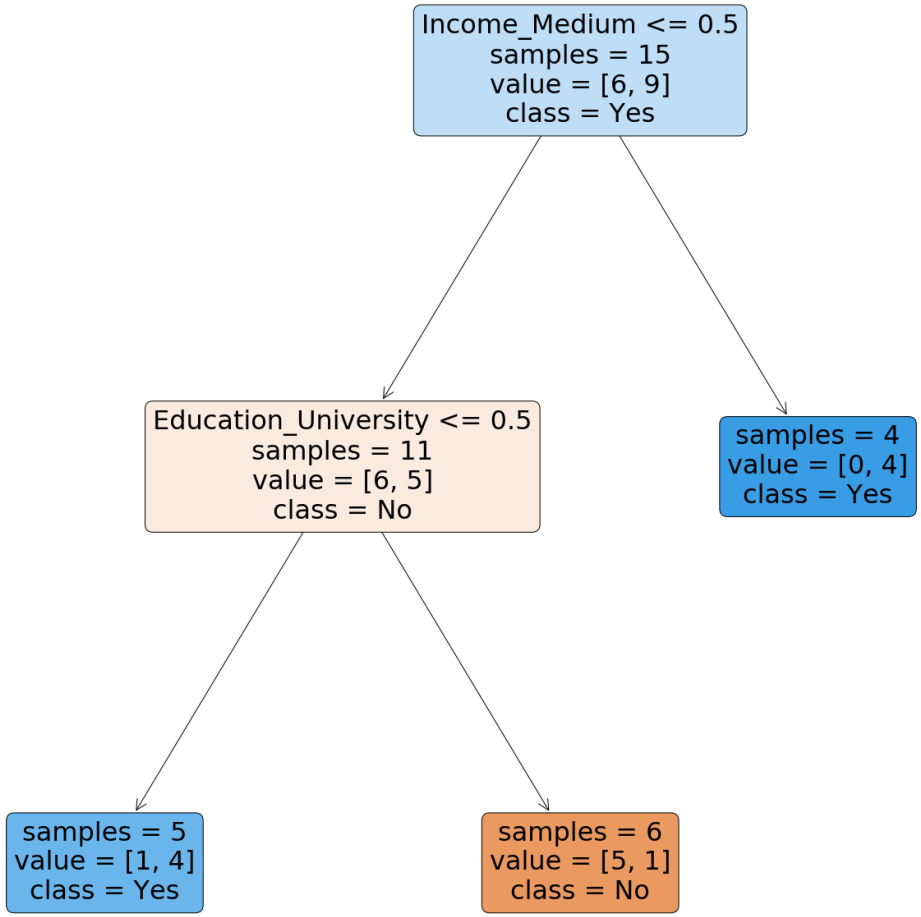
Answer: `Income_Medium`

Because the average entropy is lower.

Entropy Calculation Example

This matches our tree from before.

At the root node we split based on `Income_Medium`



Gini

In practice we often use Gini instead of Entropy because it is computationally faster since we don't need to calculate a log. Gini values closely follow Entropy.

Definition (Gini)

Let $\mathcal{D}$ be a dataset with K categories, p_k be the proportion of class k in the dataset, then the Gini index is

$$G(\mathcal{D}) = \sum_{k=1}^{K} p_k(1 - p_k)$$

Tree Building Algorithm

Recursively do the following on each node

1. Loop through every feature of the set $\mathcal{D}$ and calculate the entropy H for the respective splits [1]
2. Select the feature which has the lowest entropy
3. Split $\mathcal{D}$ by the selected feature, to produce new leaf nodes

[1] This is a simplification, we will revisit later

Stopping Criteria

Inherent

- All observations in the subset belongs to the same class (Yes or No, 1 or 0, + or -)
- All observations have constant feature values

Heuristic

- Depth of tree reaches pre-specified limit
- Number of training observations in the subset are less than a threshold
- Entropy is less than a pre-defined threshold

Take a Break

Continuous Variables

Continuous variables are split based on thresholds, for each numeric variable we construct multiple temporary new **feature-threshold pairs**.

The same rules apply, we select the pair that gives the lowest entropy.

Continuous Variables

Customer	Income	Education	Marital Status	Age	Purchase
1	Medium	University	Single	22	Yes
2	High	University	Single	21	No
3	High	University	Married	27	No
4	Low	University	Single	21	Yes
5	Low	High school	Single	27	Yes
6	Low	High school	Married	27	No
7	Medium	High school	Married	27	Yes
8	High	University	Single	25	No
9	High	High school	Single	22	Yes
10	Low	High school	Single	27	Yes
11	High	High school	Married	32	Yes
12	Low	University	Married	22	No
13	High	University	Single	32	No
14	Medium	University	Married	27	Yes
15	Medium	High school	Single	25	Yes

Continuous Variables

After creating "less than or equal" thresholds from the unique values of Age

Income_Low	Income_Medium	Education_University	Marital Status_Single	Age_22	Age_21	Age_27	Age_25	Age_32	Purchase_Yes
0	1	1	1	1	0	1	1	1	1
0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	0	1	0	1	0
1	0	1	1	1	1	1	1	1	1
1	0	0	1	0	0	1	0	1	1
1	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	1	0	1	1
0	0	1	1	0	0	1	1	1	0
0	0	0	1	1	0	1	1	1	1
1	0	0	1	0	0	1	0	1	1
0	0	0	0	0	0	0	0	1	1
1	0	1	0	1	0	1	1	1	0
0	0	1	1	0	0	0	0	1	0
0	1	1	0	0	0	1	0	1	1
0	1	0	1	0	0	1	1	1	1

Continuous Variables

Thresholds can be determined in two ways:

- **Intervals**: either evenly spaced or distributed according to each features distribution, bounded by the minimum and maximum values observed
- **Samples**: construct thresholds using the unique values that appear for each feature or if the data is large enough, by drawing a representative sample of feature values

See Ross Quinlan's book "C4.5: Programs for Machine Learning" for more details.

Overfitting

Trees can very easily overfit to training data. Some possible cases that will lead to overfitting:

- keep growing the tree until a perfect classification for the training set.
- keep splitting the true until each leaf node contains 1 example.

In both cases our tree will likely have poor generalisation ability.

Fixing Overfitting

- **During Training:** Use stopping criteria to terminate early.
- **Postpruning:** Fully grow the tree, then prune back leaf nodes until a condition is met.

Fixing Overfitting - Post Pruning

We can define the cost-complexity of a tree as

$$R(T) = M(T) + \lambda * |T|$$

where $M(T)$ is the misclassification rate of tree T, $|T|$ is the number of leaf nodes and λ is a hyper-parameter indicating our preference for smaller or larger trees.

Given a fully grown tree we then find the sub-tree which minimises $R(T)$.

Alternatively we evaluate sub-tree performance on a validation set.

Information Gain

Information Gain

When we select features based on minimum entropy, we are maximising information gain (IG), which is defined as

$$IG(\mathcal{D}, A) = \text{Entropy before splitting} - \text{Entropy after splitting}$$

The information gain tells us how much more we can be certain about the target value after we split based on a feature.

Information Gain

Definition

The information gain after splitting data set $\mathcal{D}$ on feature A is formally defined as

$$IG(\mathcal{D}, A) = H(\mathcal{D}) - H(\mathcal{D}|A)$$

where $H(\mathcal{D}|A)$ is the expected entropy when split by feature A, which is

$$H(\mathcal{D}|A) = \sum_{j=1}^{J} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} H(\mathcal{D}_j)$$

where $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_J$ are all the split subsets on feature A and $|\mathcal{D}_j|$ is the number of data points in $\mathcal{D}_j$, and $|\mathcal{D}|$ is the total number of data points.

Information Gain

For each split we calculate the IG as our metric

$$IG(\mathcal{D}, A) = H(\mathcal{D}) - H(\mathcal{D}|A)$$

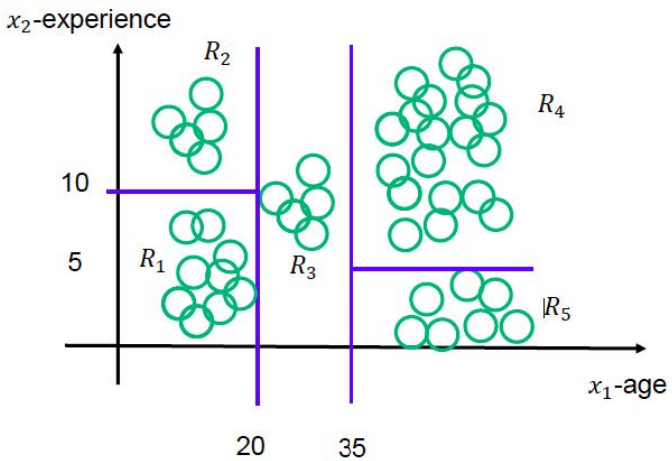
However note when we are comparing splits then $H(\mathcal{D})$ is a constant which we can ignore.

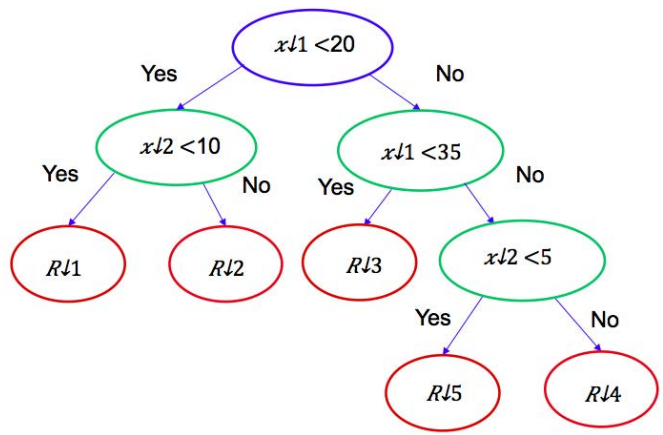
Regression

Regression Trees

For a new data point $x^* = (x_1^*, x_2^*)$, we can traverse the tree to arrive one of five regions R_1, R_2, R_3, R_4 and R_5.

The prediction is the average response values of all the training data in the same region.





Building a Regression Tree

For regression trees, our goal is to find regions $R_1, R_2, ..., R_J$ that minimise the following **Loss Function**,

$$\min_{R_1,R_2,...,R_J} \sum_{j=1}^{J} \sum_{x_i \in R_j} (y_i - \overline{y}_{R_j})^2$$

where $\overline{y}_{R_j}$ is the average response of training examples inside region R_j.

The variables (or parameters) in the optimisation problem are $R_1, R_2, ..., R_J$. Finding these regions is a difficult problem. It is **computationally infeasible** to consider every possible partition of the feature space into J regions.

Building a Regression Tree

The earlier example implies that the regions are defined by separating lines.

Each line can be described by a cut-point, which we denote as θ_i, where i is the feature index.

The problem of building the regions is now transformed into finding the θ's that describe the regions.

Building a Regression Tree

Within each region recursively repeat the following

1. Select the feature i and cut-point θ_i that yields the lowest loss function value in the proposed regions.

$$R_1(k,\theta) = \{\mathrm{x}|x_k < \theta\} \quad \text{and} \quad R_2(k,\theta) = \{\mathrm{x}|x_k \geq \theta\}$$

$$\text{Loss} = \sum_{\mathrm{x}_i \in R_1(k,\theta)} \left(y_i - \overline{y}_{R_1}\right)^2 + \sum_{\mathrm{x}_i \in R_2(k,\theta)} \left(y_i - \overline{y}_{R_2}\right)^2$$

2. Split the feature space with the best feature and cut-point pair.

Similar to handling numeric values in classification trees, the cut-points can be found by iterating over the range or using values from the data.

Wrapping Up

Decision Trees Summary

Advantages

- Learning and classification is fast
- Closely mirror human decision making process.
- Easy to interpret as sets of decision rules
- Can be used as a benchmark before more complicated algorithms are attempted
- Handles categorical variables easily

Disadvantages

- Trees can easily overfit to training data
- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches, especially regression tree

History

In 1986, Ross Quinlan proposed the **ID3** algorithm, which uses entropy to select features.

In 1993, Quinlan advanced on his original ideas with the **C4.5** algorithm, which added the ability to handle numeric features, missing data and he suggested a post-pruning technique.

CART (Classification and Regression Trees) is similar to ID3/C4.5 with an extension of to support regression and was developed around the same time.

In this lecture we discussed a simplified version of CART.