COMP6237 Data Mining

Modelling with Decision Trees

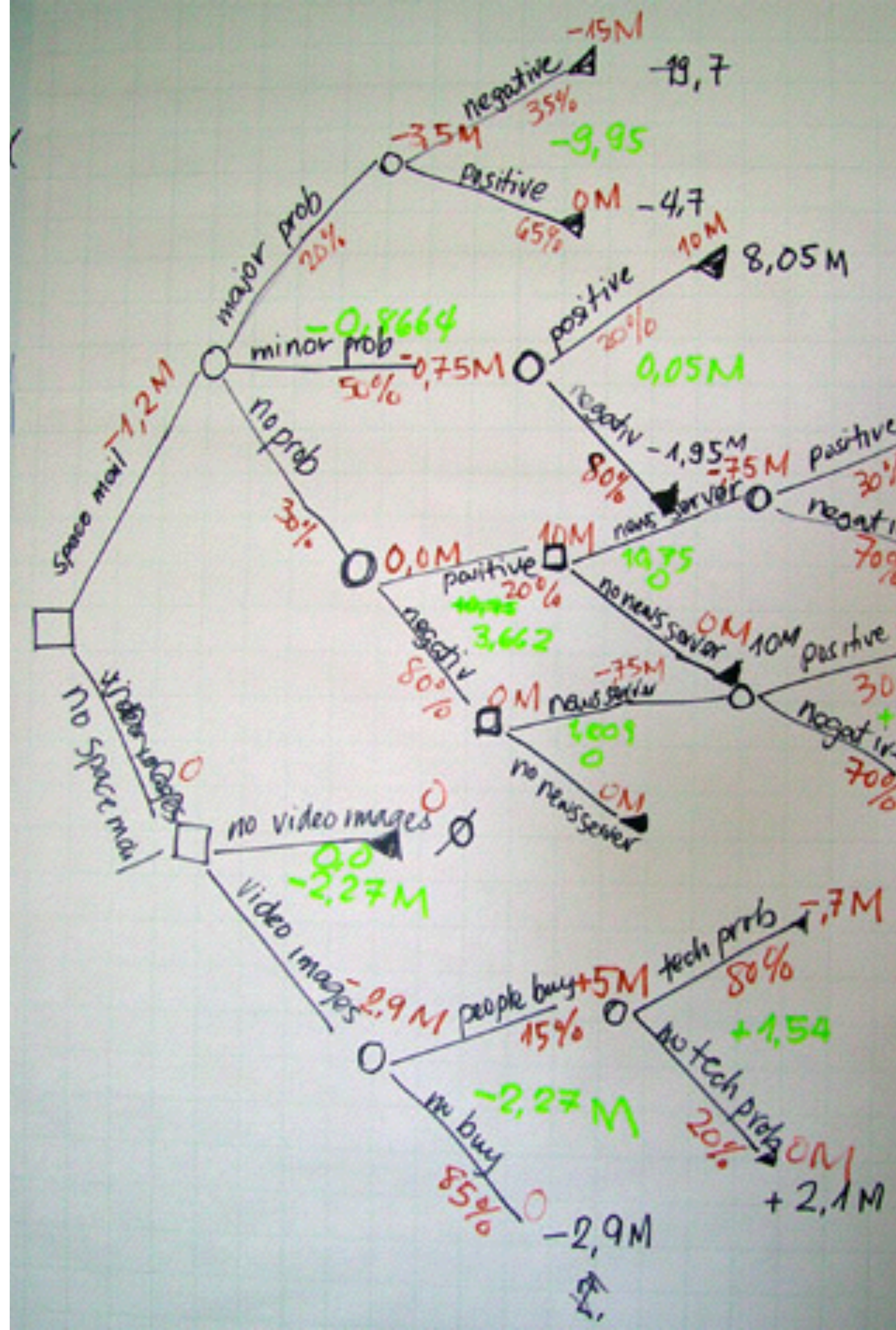
Jonathon Hare
jsh2@ecs.soton.ac.uk

Introduction

- What are decision trees?
- Classification trees
- Regression trees
- Ensembles of trees

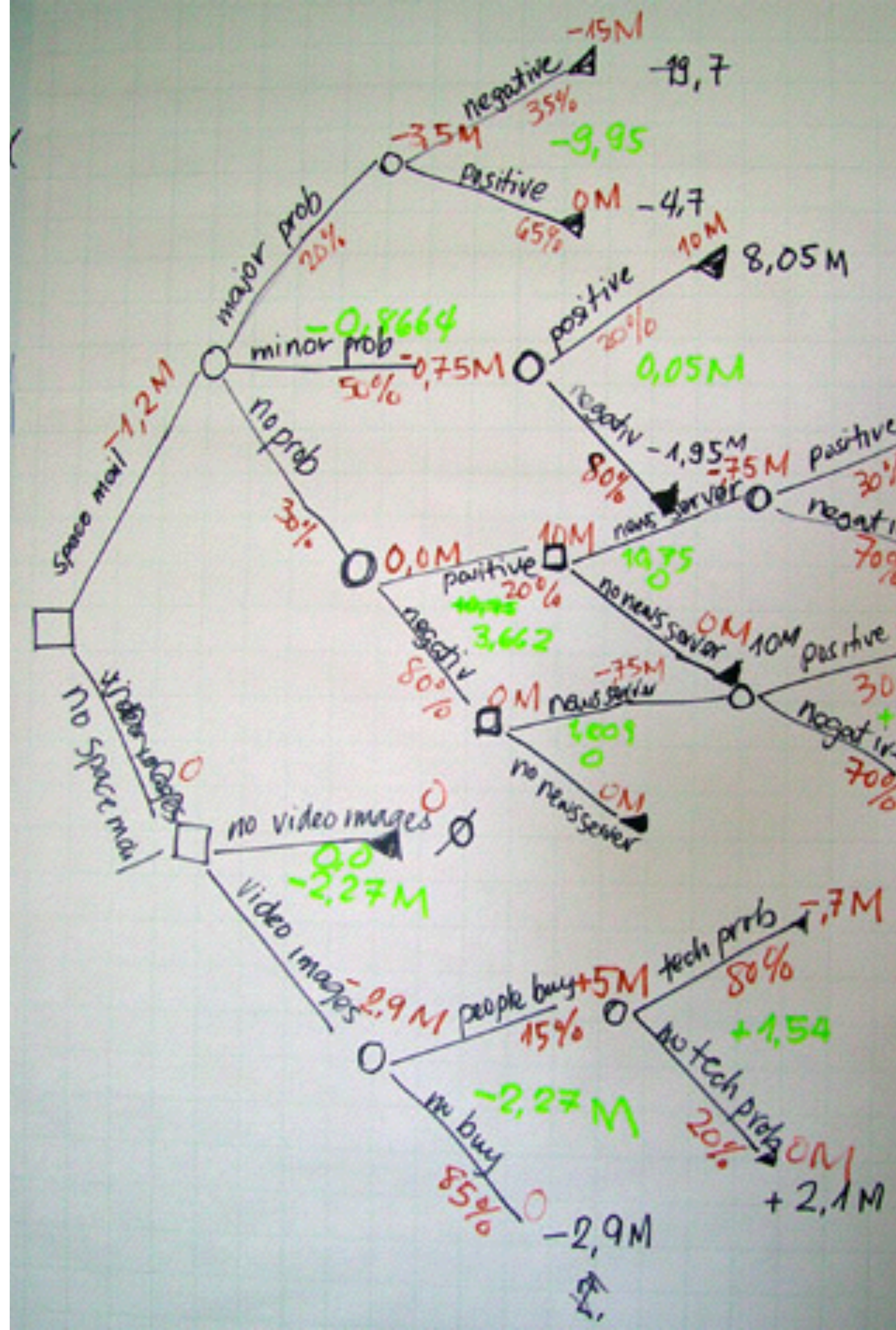
What is a decision tree?

- Flowchart-like structure
  - Each internal node represents a “test” on a feature
  - Each branch represents an outcome of the test
  - Each leaf node represents a class label
  - Path from root to leaf represents a conjunction of tests that lead to a class label



What is a decision tree?

- Decision Tree Analysis
  - Trees explicitly represent *decisions* and *decision making*
  - Often used in operational research for “decision analysis”
  - Typically trees hand crafted
- **Decision Tree Learning**
  - Tree describes data
    - Can be used as an input to *decision making*
  - Use machine learning to learn trees that can be used as **predictive models**
    - Both **classification** and **regression** are possible



Why model with decision trees?

- **Interpretability**
  - Bayes classifiers could tell us about importance of words, but have to do computation to see actual result
  - Weights in a neural network really difficult to understand
  - What about the hyperplane of a linear classifier?
- **Decision trees make the reasoning process explicit**

Decision trees in the Real World

- Very popular in medicine
  - Doctors want to be able to understand and check the reasoning of automated classifiers
- Financial analysis
  - e.g. Decision support for managing hedge funds using the *buy-write* strategy
- Astronomy
  - e.g. determining start field counts; discovering quasars; …
- and many more…

Decision trees in the Real World

- Typically used in applications where a domain expert is involved

- Often tree is created automatically and expert will

  - use it to understand key factors and

  - refine to match their own beliefs (domain knowledge)

Problem statement: Predicting Signups

- Suppose that we're running an online application that offers a free trial
  - We would like to be able to identify the users who are most likely to become paying customers
    - Rather than spamming everyone, we could target our marketing at these people
    - To minimise annoyance when people sign-up for a trial we don't ask too many questions
      - Prediction will be done on the basis of tracking user behaviour from server logs

Sample Data:

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Google	France	Yes	23	Premium
Digg	USA	Yes	24	Basic
Kiwitobes	France	Yes	23	Basic
Google	UK	No	21	Premium
(direct)	New Zealand	No	12	None
(direct)	UK	No	21	Basic
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	USA	No	18	None
Google	UK	No	18	None
Kiwitobes	UK	No	19	None
Digg	New Zealand	Yes	12	Basic
Google	UK	Yes	18	Basic
Kiwitobes	France	Yes	19	Basic

Building trees

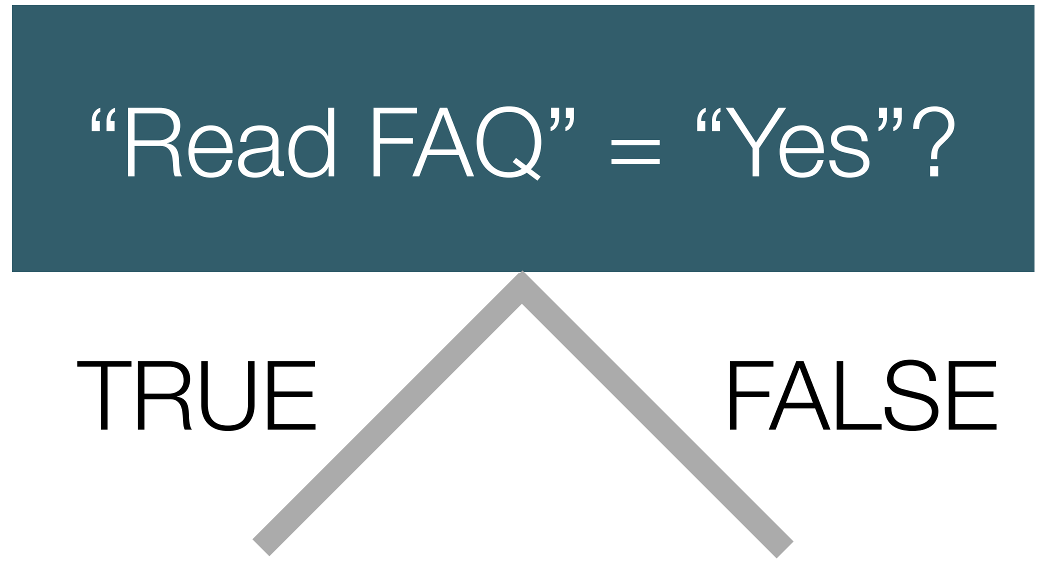
- Lots of different algorithms:
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 (ID3's successor)
  - **CART** (Classification And Regression Trees)
  - CHAID (CHi-squared Automatic Interaction Detector)
  - …

CAR**T**

- Conceptually simple idea
- Recursively split dataset by choosing a *feature* and a *value* to split on
  - For numeric features splits can be *feature* >= *value*
  - For categorical features split can be *feature* == *value*

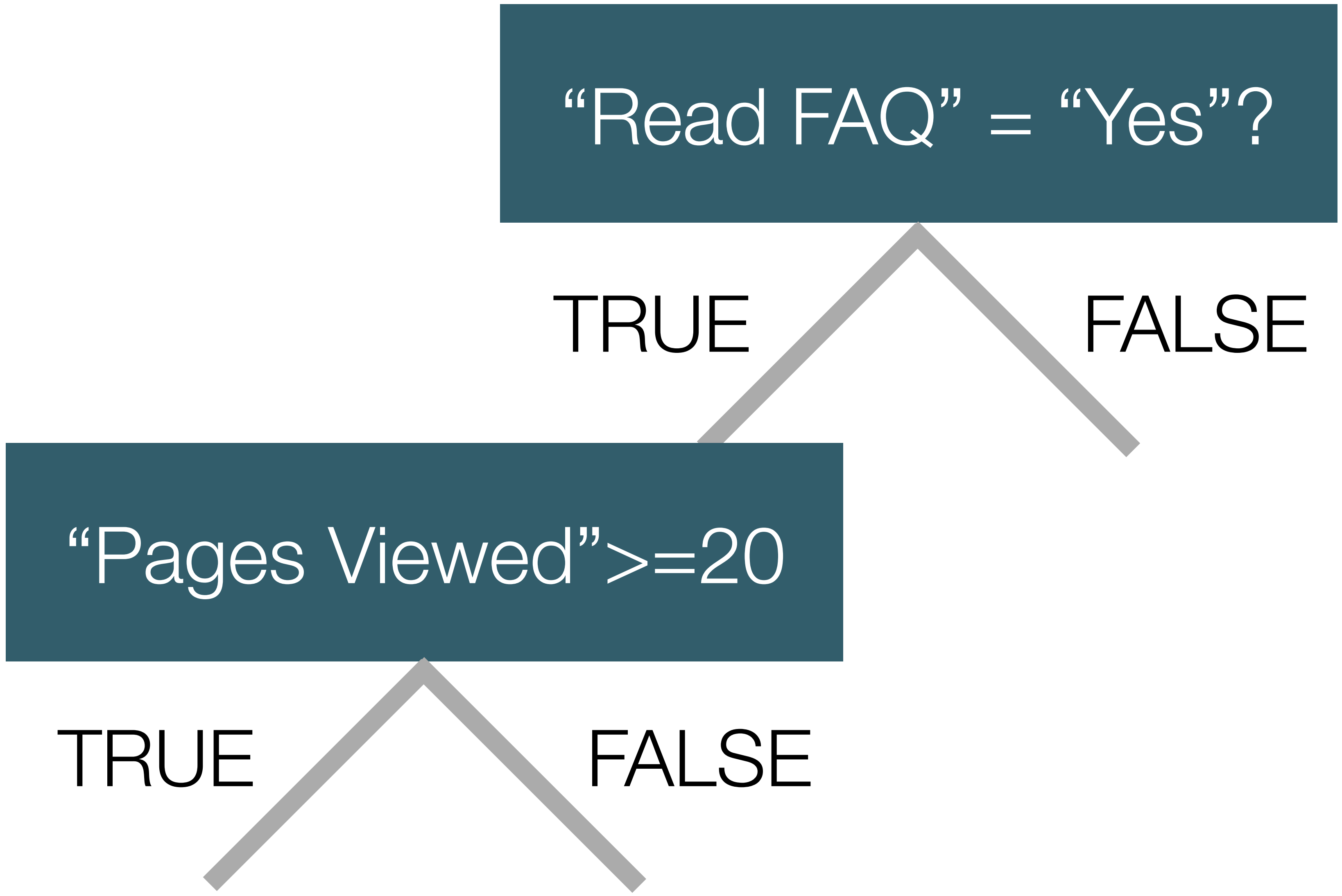
Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Google	France	Yes	23	Premium
Digg	USA	Yes	24	Basic
Kiwitobes	France	Yes	23	Basic
Google	UK	No	21	Premium
(direct)	New Zealand	No	12	None
(direct)	UK	No	21	Basic
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	USA	No	18	None
Google	UK	No	18	None
Kiwitobes	UK	No	19	None
Digg	New Zealand	Yes	12	Basic
Google	UK	Yes	18	Basic
Kiwitobes	France	Yes	19	Basic

chosen feature = "Read FAQ"; value="Yes"


"Read FAQ" = "Yes"?
TRUE
FALSE

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Google	France	Yes	23	Premium
Digg	USA	Yes	24	Basic
Kiwitobes	France	Yes	23	Basic
Google	UK	No	21	Premium
(direct)	New Zealand	No	12	None
(direct)	UK	No	21	Basic
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	USA	No	18	None
Google	UK	No	18	None
Kiwitobes	UK	No	19	None
Digg	New Zealand	Yes	12	Basic
Google	UK	Yes	18	Basic
Kiwitobes	France	Yes	19	Basic

chosen feature = “Pages Viewed”; value=20


“Read FAQ” = “Yes”?
TRUE
FALSE
“Pages Viewed”>=20
TRUE
FALSE

Choosing the best split

- For the tree to be useful we ideally want it to separate the classes as effectively and efficiently as possible
    - i.e. we want a split to minimise the amount of mixing of different classes in its two children
- Need measures of amount of mixing (*impurity measures*)
    - Gini Impurity
    - Entropy

Gini Impurity

- a measure of how often a randomly chosen item from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset

Number of labels in set

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i \neq k} f_i f_k$$

Fraction of items in set labelled with label i

- if all items have the same label, Gini impurity is 0

- if there are 4 labels with equal likelihood, Gini impurity is 0.75

Entropy

- Measure the amount of disorder in the set:

Number of labels in set

$$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$$

Fraction of items in set labelled with label i

- if all items have the same label, entropy is 0 bits
- if there are 4 labels with equal likelihood, entropy is 2 bits

Splitting a node

- Given a node, *N*, in the tree with *n* items
  - Compute its impurity *I(N)*
  - **Search** for the predicate that splits the data in such that it maximises the improvement in impurity:
    $\boldsymbol{I(N) - ((n_L/n)I(L) + (n_R/n)I(R))}$
    - *n_L and n_R is the number of items that would fall into the left and right branches*
    - *I*(*L*) and *I*(*R*) are the impurity of the left and right subsets formed from the branches

Splitting a node

- Given a node, *N*, in the tree with *n* items
  - Compute its impurity *I(N)*
  - **Search** for the predicate that splits the data in such that it maximises the improvement in impurity:
    $\boxed{I(N) - ((n_L/n)I(L) + (n_R/n)I(R))}$
    - *n_L and n_R is the number of items that would fall into the left and right branches*
    - *I*(*L*) and *I*(*R*) are the impurity of the left and right subsets formed from the *branches*

If we're using entropy, this is known as the information gain

Example

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i \neq k} f_i f_k$$

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Digg	USA	Yes	24	Basic
Google	UK	No	21	Premium
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	New Zealand	Yes	12	Basic

Overall Gini Impurity: 0.666

Example

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i \neq k} f_i f_k$$

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Digg	USA	Yes	24	Basic
Google	UK	No	21	Premium
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	New Zealand	Yes	12	Basic

Overall Gini Impurity: 0.666

Assume split on `Referrer=Slashdot`:

I_G(Left) = 0

I_G(Right) = 0.5

Gain = 0.666-(2/6)*0-(4/6)*0.5 = 0.333

Example

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i \neq k} f_i f_k$$

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Digg	USA	Yes	24	Basic
Google	UK	No	21	Premium
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	New Zealand	Yes	12	Basic

Overall Gini Impurity: 0.666

Assume split on `Referrer=Digg`:

I_G(Left) = 0

I_G(Right) = 0.5

Gain = 0.666-(2/6)*0-(4/6)*0.5 = 0.333

Example

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i \neq k} f_i f_k$$

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None
Digg	USA	Yes	24	Basic
Google	UK	No	21	Premium
Google	USA	No	24	Premium
Slashdot	France	Yes	19	None
Digg	New Zealand	Yes	12	Basic

Overall Gini Impurity: 0.666

Assume split on `Pages viewed>=21:`

I_G(Left) = 0.444

I_G(Right) = 0.444

Gain = 0.666-(3/6)*0.444-(3/6)*0.444 = 0.222

Stopping

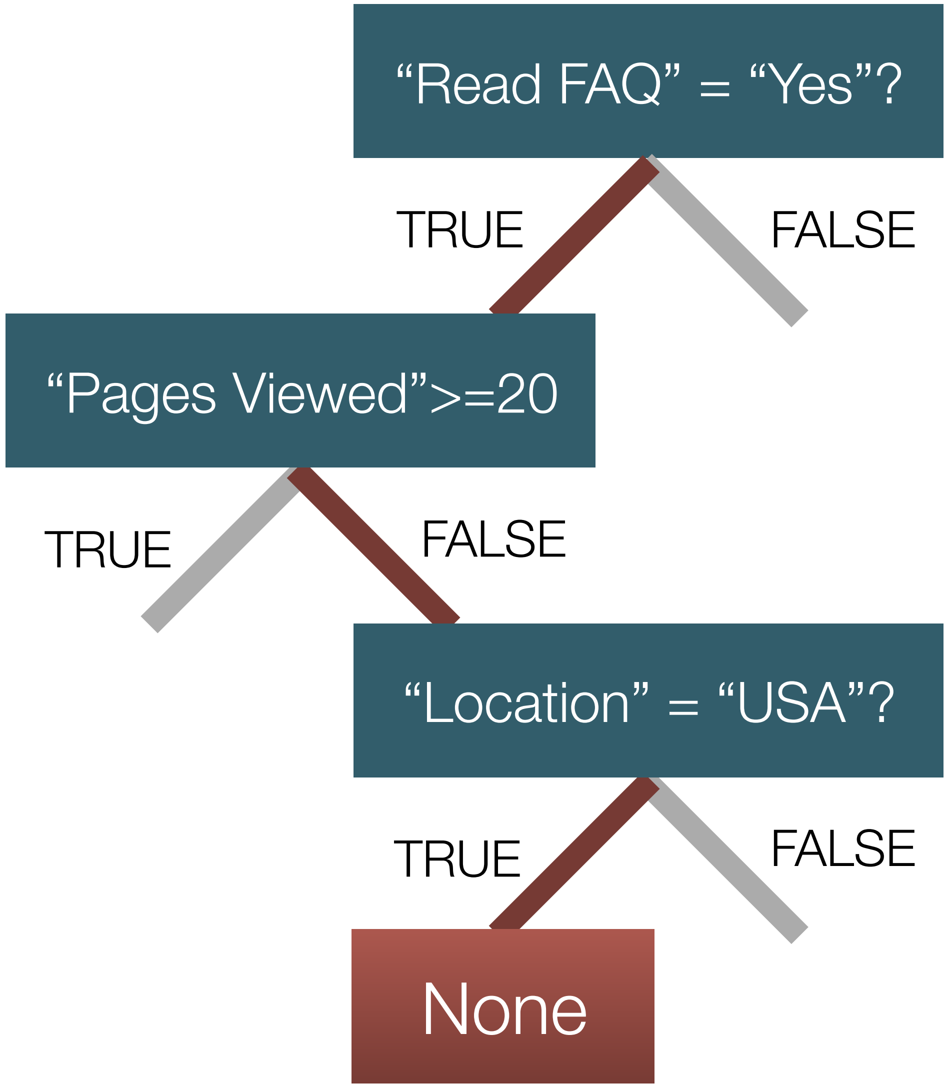
- Stop when the node is completely pure or can't be split further
  - *Note: might be identical instances with different classes*
- Leaf node is either
  - labelled with the majority class of the children data items
  - labelled with all classes, together with counts of items

Tree Building Demo

Making classifications

- Simple!
  - Given a data item, walk down the tree by comparing the predicate at each node against the item's features
  - When you hit a leaf node, the label of that node is the predicted class of the item
    - or the most likely label of that leaf

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	18	None



Tree Classification Demo

Overfitting

- Decision trees trained using the CART algorithm (and other similar approaches) have a big problem
  - They tend to **overfit** the to the data
  - Tend to **generalise poorly** to new data
  - Tend to create trees that are **too complex**

Tree Pruning

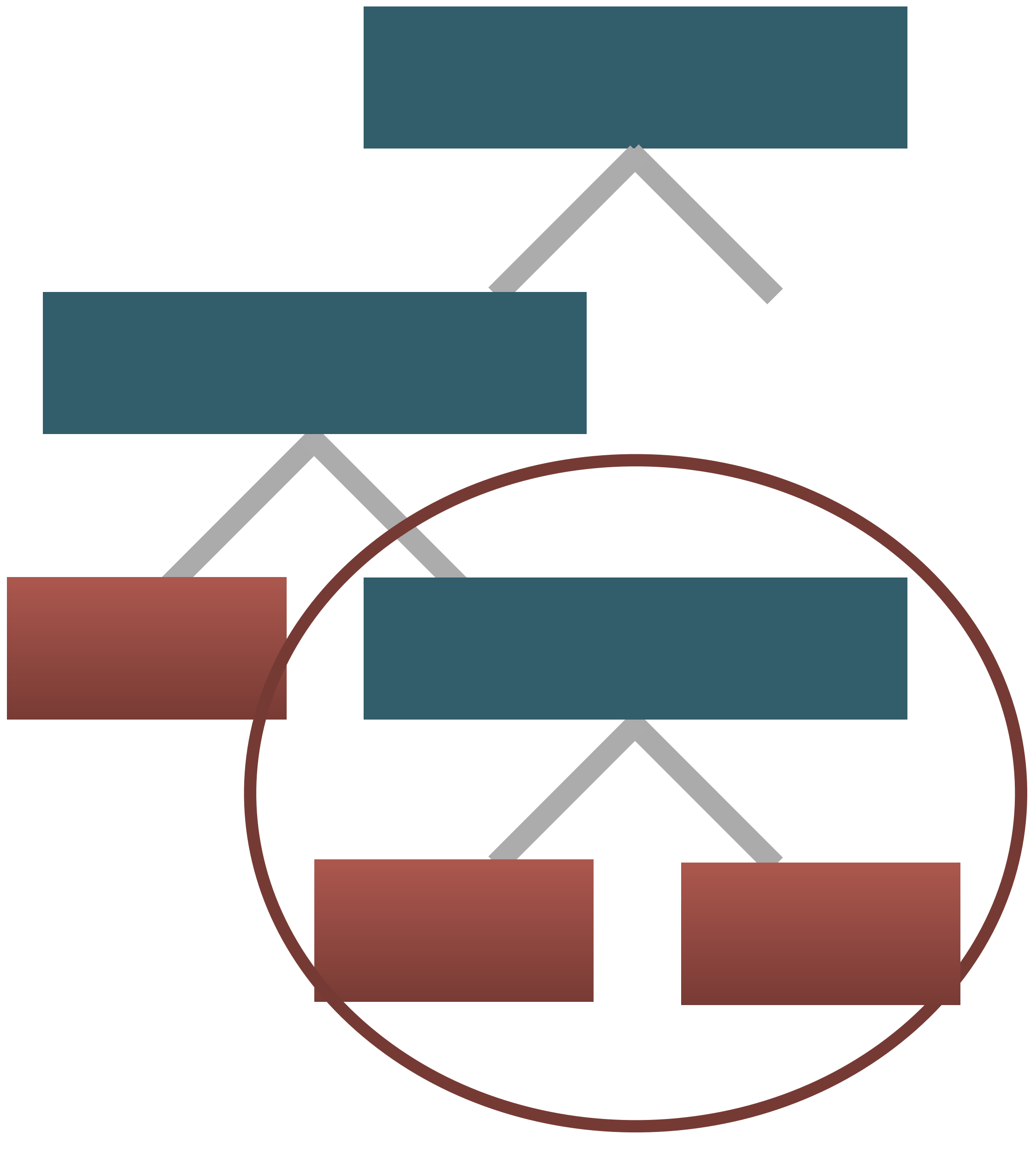
- One solution to overfitting is to grow the tree fully, and then **prune** it back
- Pruning should reduce the size of the tree
  - typically without reducing predictive accuracy as measured using a *validation set*
- Many different techniques
  - usually varying with respect to *measurement* used to optimise performance

Reduced Error Pruning

- Starting at the leaves, each node is replaced with its most popular class.
    - If the prediction accuracy (tested on the validation set) is not affected then the change is kept.
    - Naive, but both simple and fast

Merging based on Entropy

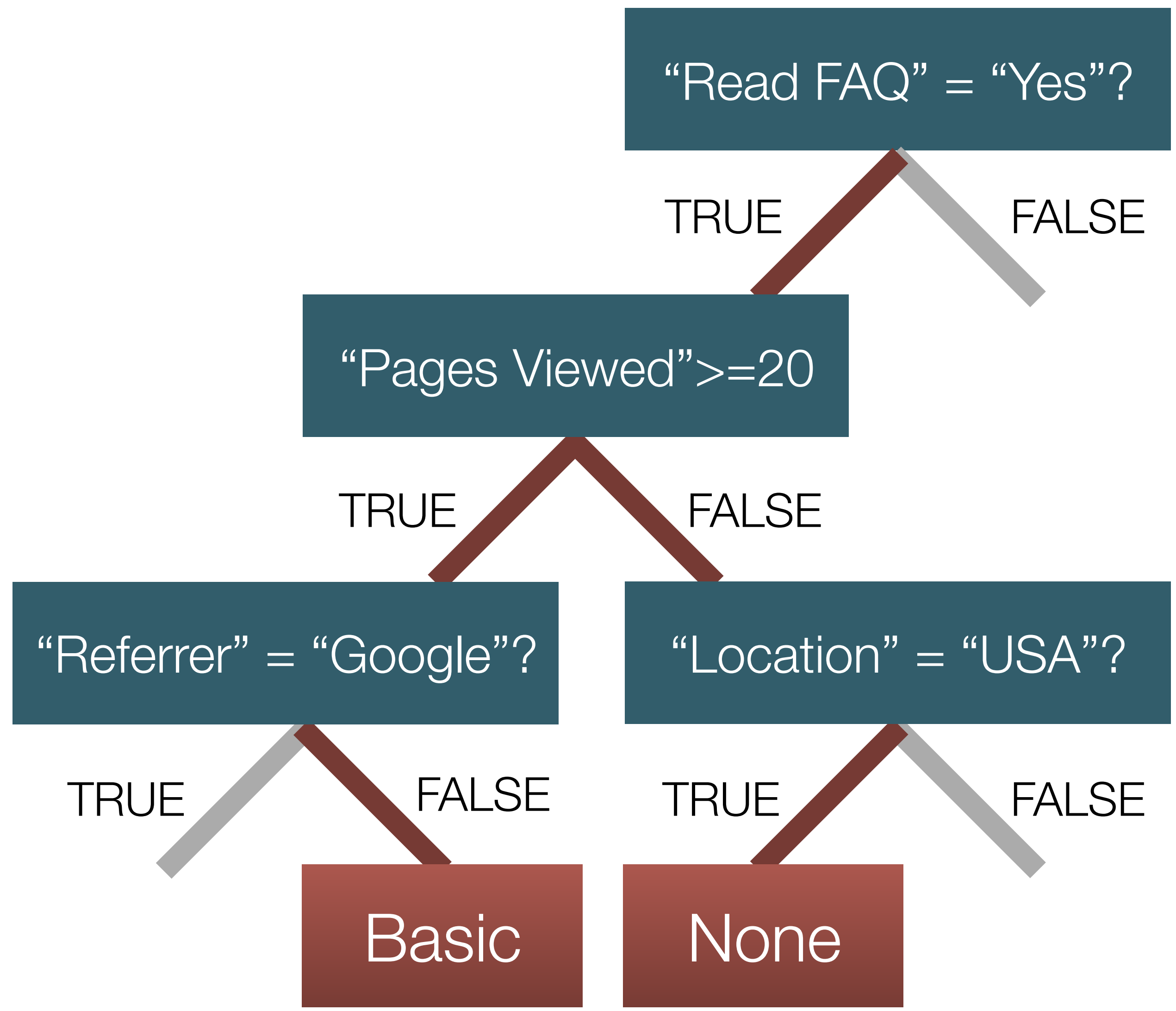
- Check pairs of leaf nodes that have a common parent to see if merging them would increase entropy by less than a threshold.
    - If so, then merge nodes
- Doesn't require additional data



Dealing with missing data

- Possible to modify classification algorithm to deal with features with unknown values
  - Follow both branches
  - Weight each branch based on fraction of counts of items in order to compute which result to prefer

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	?	None



CA**RT**: Dealing with numerical outcomes

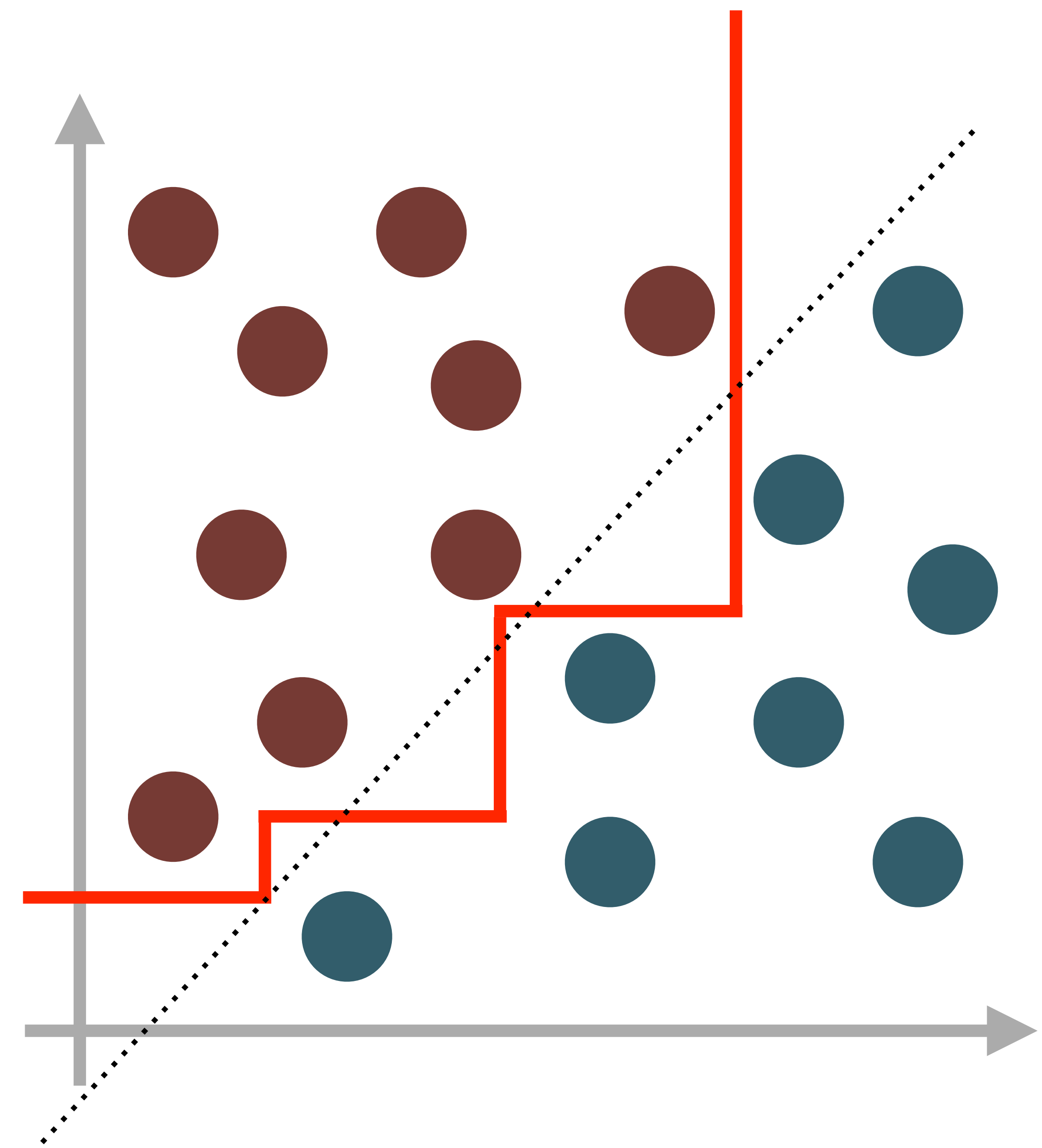
- Can we modify how we build trees so that rather than performing classification they predict numerical outcomes?
  - i.e. perform regression
- Could use the same approach as for classification, but…
  - each numeric outcome would be considered to be a single class, with no regard for ordering or similarity

Variance Reduction

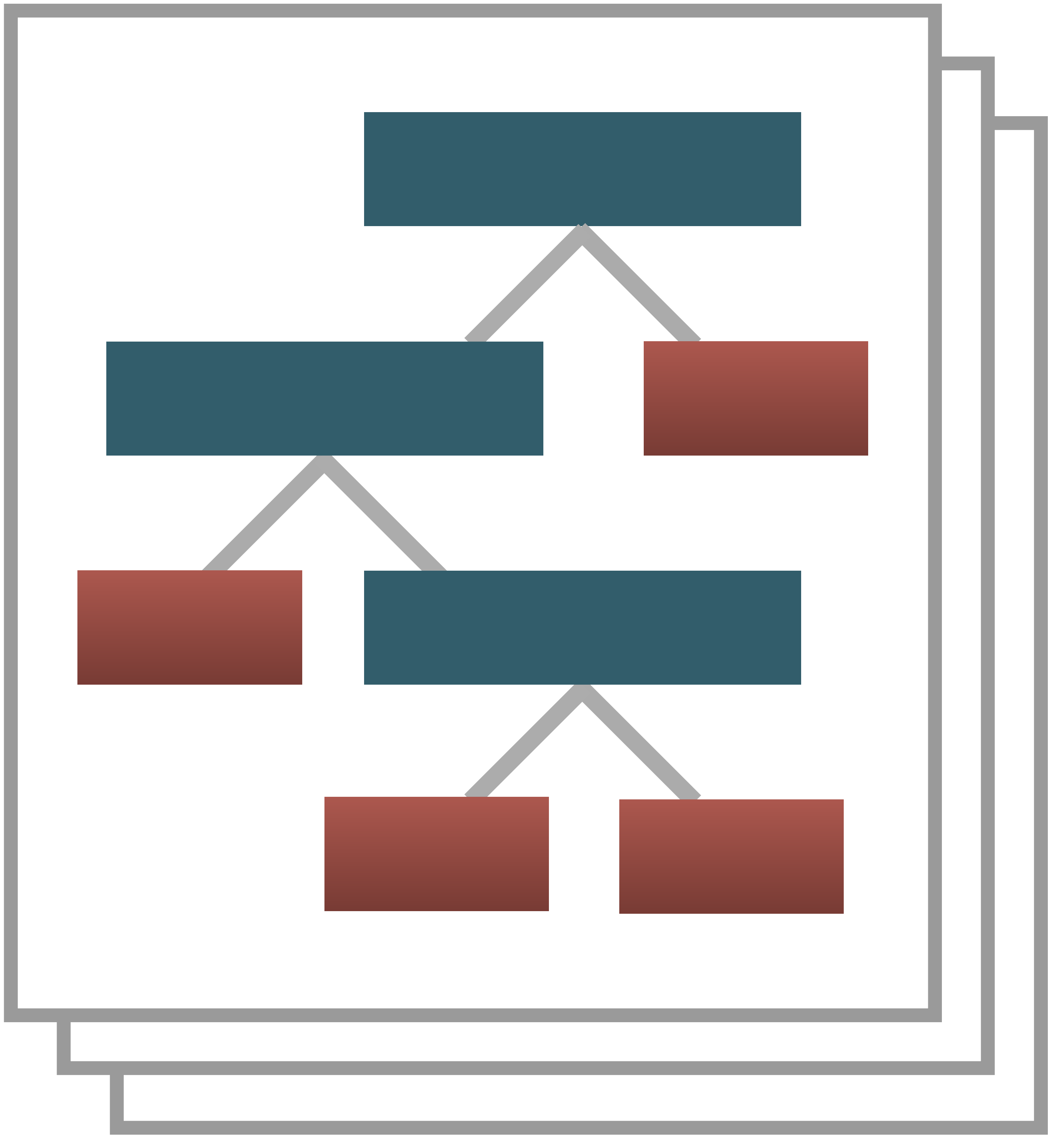
- Instead of using entropy or Gini, use variance instead
    - (e.g. best feature-value is the one that maximises variance-gain)
    - Resultant split will try to ensure numbers with similar magnitudes are grouped together
        - low numbers of one side of the split
        - high numbers on the other

Problems with CART-like trees

- Learning optimal tree is NP-complete!
- Generalisation/overfitting
    - hence need to prune (or use a different algorithm [with it's own problems])
- Information gain is biased by features with more categories
- Splits are performed in an axis-aligned manner…



Ensemble methods



Bagging

- **B**ootstrap **agg**regat**ing**
  - **Uniformly** sample initial dataset **with replacement** into *m* subsets
  - train a classifier/regressor (e.g. decision tree) for each subset
  - To perform classification apply each classifier and choose by voting (i.e. take mode)
    - For regression take mean
- Leads to better performance - decreases variance without increasing bias

Boosting

- Can a set of weak learners create a single strong learner?
  - Learn a **sequence** of weak trees (i.e. fixed size trees)
  - **Gradient Tree Boosting**

Random Forests

- Applying bagging
  - but for each subset when learning the tree choose the split searching over a random sample the features rather than all of them
- Improves bagging by reducing overfitting

Summary: When should you use a decision tree?

- Advantages
  - Interpretability
  - Ability to work with numerical and categorical features
- Disadvantages
  - Might not effectively scale to large numbers of classes
  - Problems learning from features that interact