Decision Trees

Q Estimate in-game win probability for American Football.

↳ as a function of game-state

* Why? — Betting
  - player valuation
  - Strategic decision making: make the decision which maximizes win probability
    ↳ fourth-down decision making: read my paper!
* Mathematical Models:
  - dynamic programming state-space models popular 20+ years ago
* Statistical Models:
  - approach: in "similar" situations across football history, what proportion of the time did the team with possession win the game?
  - play-by-play NFL data is easily accessible today (nflfastR) machine learning models can be fit quickly today Hence today, everyone does this
  - we will focus on statistical models

* **Task** Estimate in-game win probability $WP(x)$ as a function of game-state x

Row index i i^{th} play in our dataset of NFL plays

outcome var. y_i whether the team with possession on this play won or lost the game

game-state variables x_i of play i:

yardline, down, distance,
score differential, game seconds remaining,
off. and def. team quality → use Point Spread,
Receive 2nd half Kickoff, timeouts

Model form $P(y_i=1) = f(x_i)$; logistic model $\log \frac{P(y_i=1)}{P(y_i=0)} = g(x_i)$

* Previous lectures: logistic Regression

$$P(\text{win}=1 \mid \underset{x}{\text{game-state}}) = \frac{1}{1+\exp\left(-\left(\beta_1 \cdot ydl + \beta_2 \cdot down + \beta_3 \cdot dist + \beta_4 \cdot (\text{time rem}) + \dots\right)\right)}$$

ydl: spline$(ydl \mid \vec{\beta}_1)$

$down$: $\beta_{21}\cdot(down=1) + \beta_{22}\cdot(down=2) + \beta_{23}\cdot(down=3) + \beta_{24}\cdot(down=4)$

What's wrong with this?

Logistic Regression model assumes additive relationships between variables, but these game-state variables interact in complicated ways and have complex non-linearities.

ex Score diff & time Remaining certainly interact.

You can model interactions in additive Regression settings but this can be extremely difficult with 2 numeric/continuous vars (but easy with binary vars)

$$\beta_1 \cdot \text{score diff} + \beta_2 \cdot \text{time Rem.} + \beta_3 \cdot (\text{score diff}) \cdot (\text{time Rem})$$

$$\text{OR} \quad + \beta_3 \cdot \frac{\text{score diff}}{\sqrt{1 + \text{time Rem}}}$$

Since many of these variables are interacting and nonlinear, logistic Regression is probably a bad idea.

→ **Machine Learning**

Learn an arbitrarily complex relationship b/t the variables from the data.

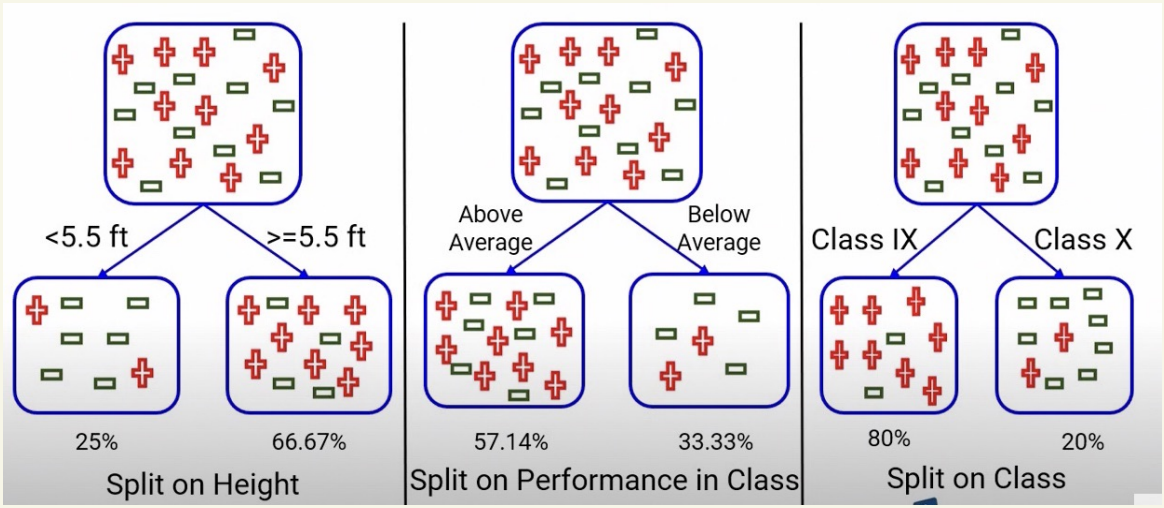
* We will focus on tree machine learning models, which are easy to use in R, fairly quick to fit, and are widely used for NFL win probabilities.

Decision Trees

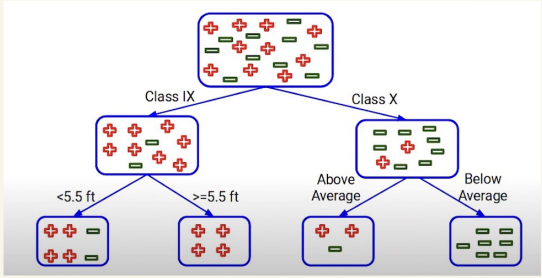
* credit to Analytics Vidhya on Youtube

Ex 20 students, 10 play cricket
Variables: Height, grades, class
Fit a model to predict whether a student plays cricket.

Idea split on an X variable to classify the y variable



Which split is best? The Rightmost one.
Want to separate the classes as best as possible.



Can have multiple splits in a decision tree!

* decision trees allow variables to interact
* but how to fit such a tree from data?

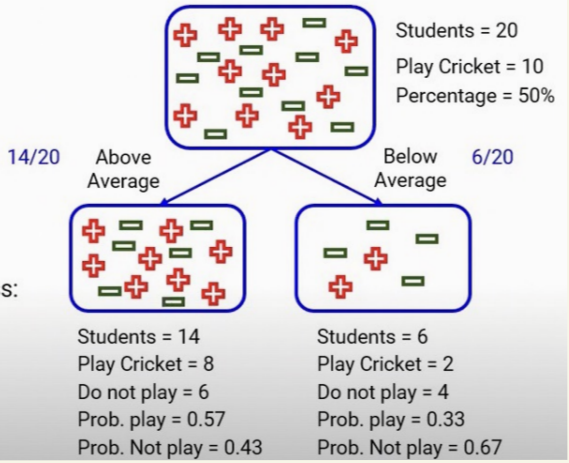
* How to select the best split point?

→ select the split which results in the most homogeneous sub-nodes

1. Gini Impurity

Split on Performance in Class

- Gini Impurity: sub-node Above Average:
  1 - [(0.57)*(0.57) + (0.43)*(0.43)] = 0.49
- Gini Impurity: sub-node Below Average:
  1 - [(0.33)*(0.33) + (0.67)*(0.67)] = 0.44
- Weighted Gini Impurity: Performance in Class:
  (14/20)*0.49 + (6/20)*0.44 = 0.475



Gini = sum of square probabilities for each outcome category, within a node

$= P_+^2 + P_-^2$

Gini Impurity of a split

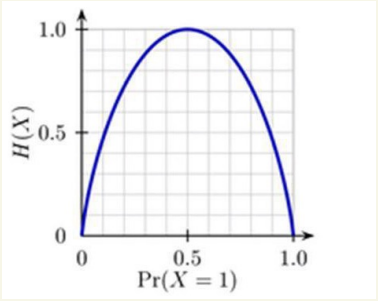
= weighted Gini Impurity of both sub-nodes of that split

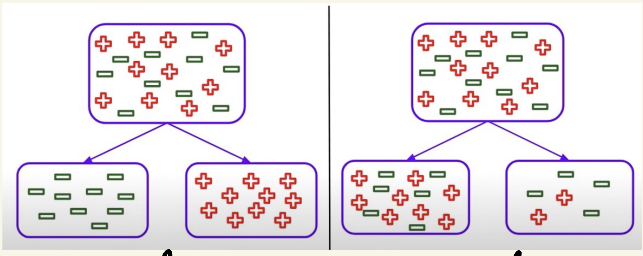
$= W_L(1 - Gini_L) + W_R(1 - Gini_R)$

Split	Weighted Gini Impurity
Performance in Class	0.475
Class	0.32

→ split on Class first!

2. Information Gain





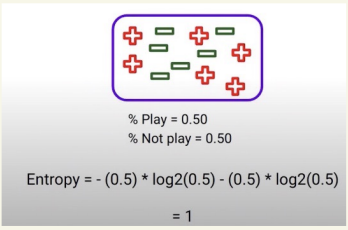
more info gained

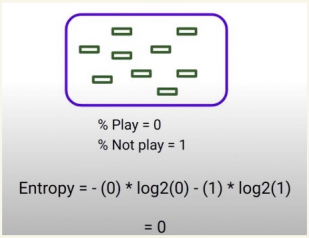
less info gained

Information Gain = 1 − Entropy

Entropy = $-P\log_2 P - (1-P)\log_2(1-P)$

where $P = P(+) = P(y=1)$

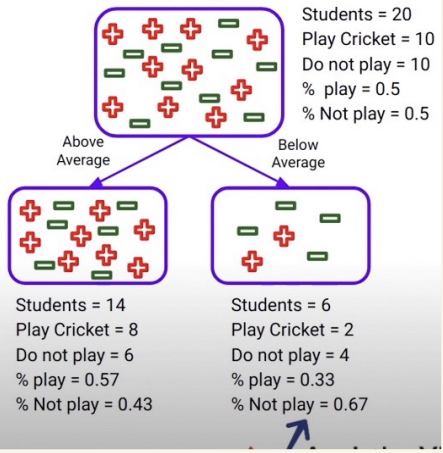




Entropy of a split = weighted average of entropy of each resulting sub-node

Split on Performance in Class

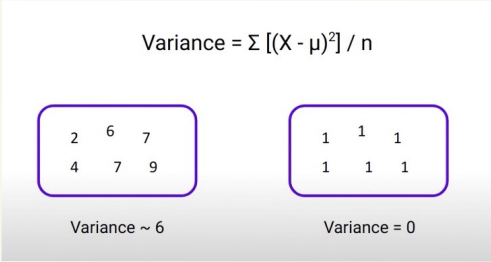
- Entropy for Parent node:
  $-(0.5)*\log_2(0.5) -(0.5)*\log_2(0.5) = 1$
- Entropy for sub-node Above Average:
  $-(0.57)*\log_2(0.57) -(0.43)*\log_2(0.43) = 0.98$
- Entropy for sub-node Below Average:
  $-(0.33)*\log_2(0.33) -(0.67)*\log_2(0.67) = 0.91$
- Weighted Entropy: Performance in Class:
  $(14/20)*0.98 + (6/20)*0.91 = 0.959$



Split	Entropy	Information Gain
Performance in Class	0.959	0.041
Class	0.722	0.278

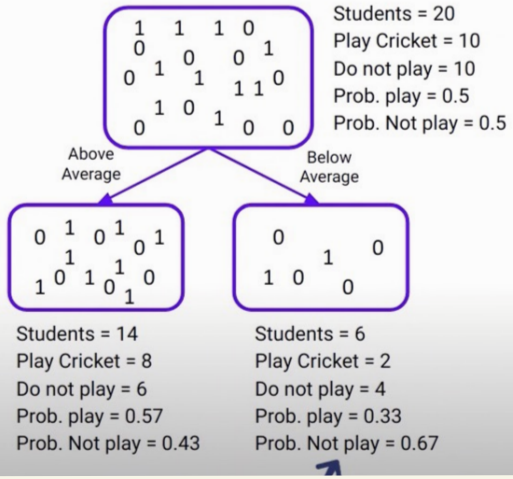
→ split on Class first!

* our outcome variable (cricket vs. no cricket / win vs. loss) is categorical, so Gini Impurity and Information gain work well. If outcome were continuous (e.g. height or money), need another way to measure splits: 3. Reduction in Variance.



split with lower variance is selected.

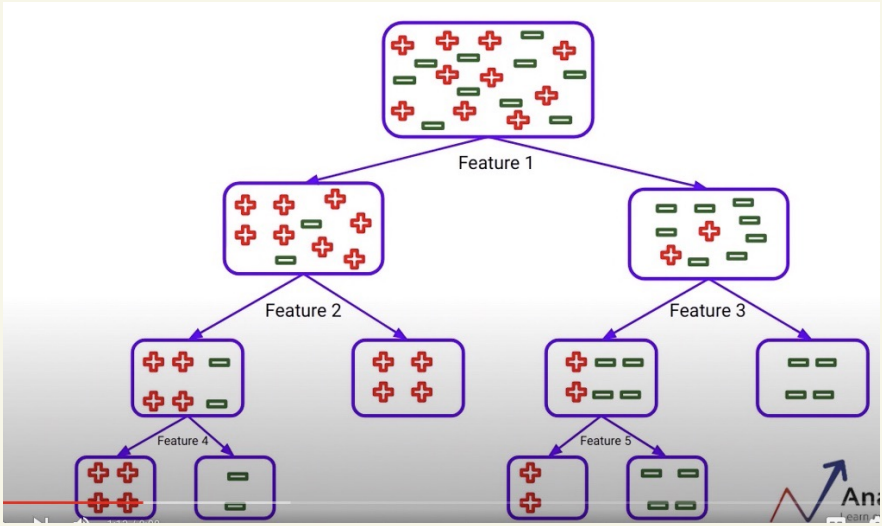
- Above Average node:
  - Mean = (8*1 + 6*0) / 14 = 0.57
  - Variance = [8*(1-0.57)² + 6*(0-0.57)²] / 14 = 0.245
- Below Average node:
  - Mean = (2*1 + 4*0) / 6 = 0.33
  - Variance = [2*(1-0.33)² + 4*(0-0.33)²] / 6 = 0.222
- Variance: Performance in Class: (14/20)*0.245 + (6/20)*0.222 = 0.238



Split	Variance
Performance in Class	0.238
Class	0.16

→ split on class first!

* Now, we know how to select the best split point!



* Could iteratively make splits until all nodes are "pure" but this would overfit (memorize noise in the training data)
* Tuning parameters (to control overfitting & underfitting)
  - max depth of tree
  - min. samples for node split
  - min. samples for a terminal node
  - max number of terminal nodes
* Straightforward to fit a decision tree in R

* Decision trees are unstable & prone to overfitting.
* How to Reduce overfitting? Ask Leo Breiman...