Decision Trees and Random Forests

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

* 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 machine learning models can be fit quickly today Hence today, everyone does this
- we will focus on statistical models

* <u>outcome var.</u> whether the team with possession on this play won or lost the game

<u>Variables</u> yardline
down
distance
score differential
game seconds remaining
off. and def. team quality → use point spread
Receive 2nd half kickoff
timeouts

* Do these variables have a linear or additive relationship?

No → Nonlinear and Interacting variables

→ $\boxed{\text{Machine Learning}}$

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

* Going to focus on strategic decision making

Q What fourth down decision should I make in {Punt, Go, FG} given the game-state?

* The entire fourth down decision making process can be defined in terms of the win probability if you have a first down with ten yards to go. Ignoring down & yards to go will simplify our modeling process.

Fourth down decision model

$V_1(x)$ = value (win probability) of a 1st and 10 at game-state x (or, for simplicity, yardline y)

* Value of punting:

$$V_{Punt} = -\sum_{y'} V_1(\text{yardline } y') \cdot P(\text{Yardline after punting is } y')$$

$$= -\mathbb{E}_{punt}\left[V_1(\text{yardline } y')\right]$$

$$\approx -V_1(\text{yardline } \mathbb{E}_{punt}[y'])$$

linear regression function of yardline (spline)

* Value of Kicking a FG:

$$V_{FG} = \underbrace{\mathbb{P}\left(\begin{smallmatrix}\text{make}\\ \text{FG}\end{smallmatrix}\right)}_{\substack{\text{logistic Regression}\\ \text{function of kicker quality and yardline}}} \cdot V\left(\begin{smallmatrix}\text{make}\\ \text{FG}\end{smallmatrix}\right) + \left(1-\mathbb{P}\left(\begin{smallmatrix}\text{make}\\ \text{FG}\end{smallmatrix}\right)\right)\cdot\left(-V_1\left(\begin{smallmatrix}\text{yardline}\\ 100-y\end{smallmatrix}\right)\right)$$

$$= -\mathbb{E}_{\text{kickoff}}\left[V_1(\text{yardline})\right]$$

$$\approx -V_1(\text{yardline } 75)$$

* Value of going for it:

4th down and z yards to go at yardline y

$$V_{GO} = \sum_{\Delta \geq z} V_1(\text{yardline } y-\Delta)\cdot\mathbb{P}(\text{gain } \Delta \text{ yards})$$

$$- \sum_{\Delta < z} V_1(\text{yardline } 100-(y-\Delta))\cdot\mathbb{P}(\text{gain } \Delta \text{ yards})$$

$$\approx \underbrace{\mathbb{P}(\text{convert})}_{\substack{\text{logistic Regression}\\ \text{function of team}\\ \text{qualities, yardline,}\\ \text{and yards to go}}}\cdot V_1(\text{yardline } y-z) + (1-\mathbb{P}(\text{convert}))\cdot V_1\left(\begin{smallmatrix}\text{yardline}\\ 100-y\end{smallmatrix}\right)$$

* So, to obtain V_{GO}, V_{FG}, V_{Punt} we need
$V_1(x)$ = value (W.P.) of a 1st and 10 at game-state x
and $\underbrace{\mathbb{E}_{punt}[y'],\ \mathbb{P}\binom{\text{make}}{\text{FG}},\ \mathbb{P}(\text{convert})}_{\text{HW: estimate these 3 functions of game-state}}$

HW: estimate these 3 functions of game-state

Task: Estimate $V_1(x) = WP(x \mid 1^{st} \text{ down and } 10)$.
using Machine Learning.

Game-state X: yardline, score differential, timeouts, game seconds remaining, point spread, receive 2nd half kickoff

Model Setup: i = index of i^{th} play in dataset of NFL plays
$y_i = 1$ if team with possession on play i wins, else 0
x_i = game-state vector of play i
$y_i = \text{Logistic}(f(x_i)) + \varepsilon_i$
Obtain the best possible $\hat{f}(x_i)$ (best predictive performance (logloss))

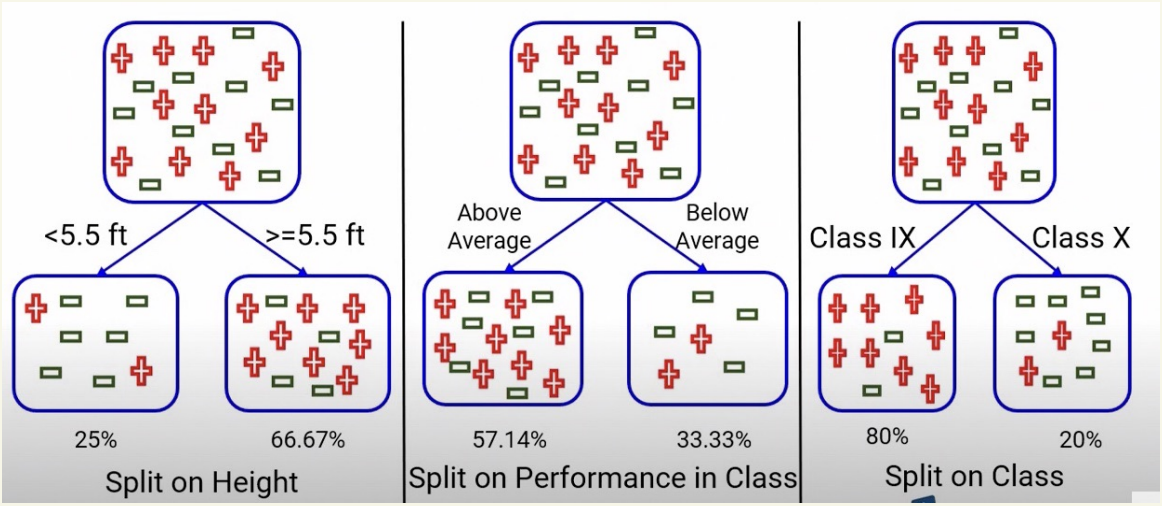
* We will focus on tree machine learning models, which are easy to use in R, fairly quick to fit, and are widely used for NFL win probabilities.

Decision Trees

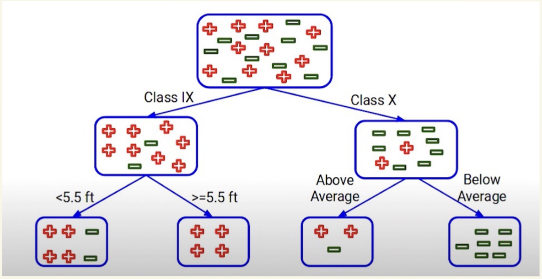
* credit to Analytics Vidhya on Youtube

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?

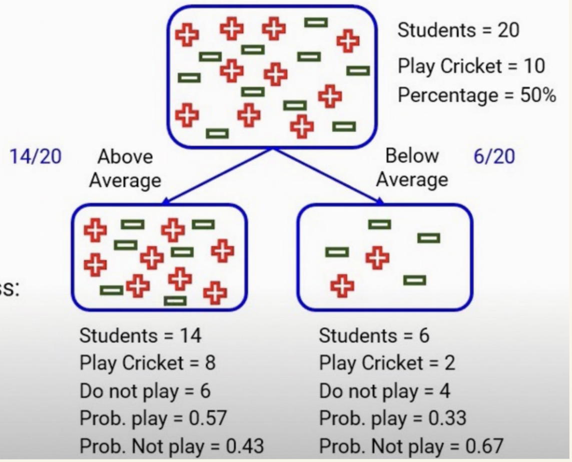
* How to select the best split point?

→ 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

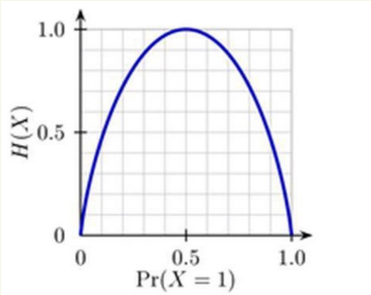
= weighted Gini Impurity of both sub-nodes of that split

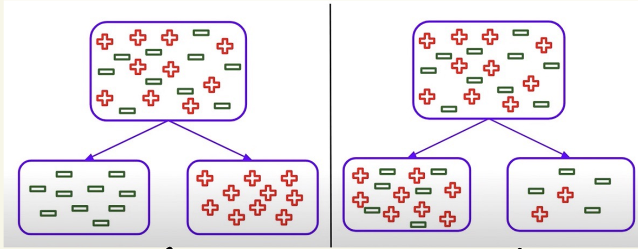
$$= W_L(1-Gini_L) + W_R(1-Gini_R)$$

Split	Weighted Gini Impurity
Performance in Class	0.475
Class	0.32

→ split on Class first!

2. Information Gain





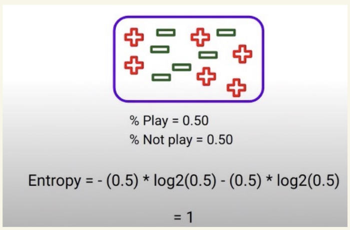
More info gained

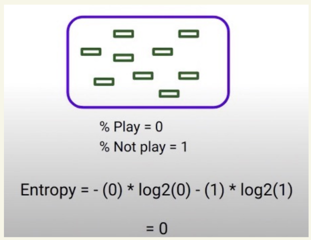
Less info gained

Information Gain = 1 − Entropy

$$\text{Entropy} = -p\log_2 p - (1-p)\log_2(1-p)$$

where $p = \mathbb{P}(+) = \mathbb{P}(y=1)$

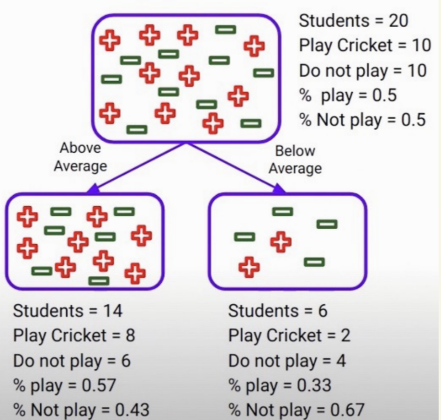




Entropy of a split = weighted average of entropy of each resulting sub-node

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

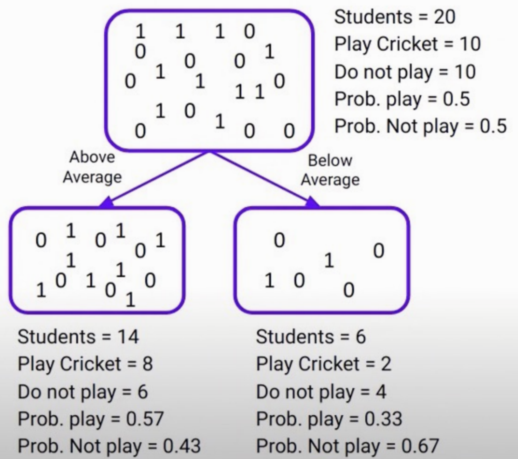
→ split on Class first!

* 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.

$$\text{Variance} = \Sigma [(X - \mu)^2] / n$$

split with lower variance is selected.

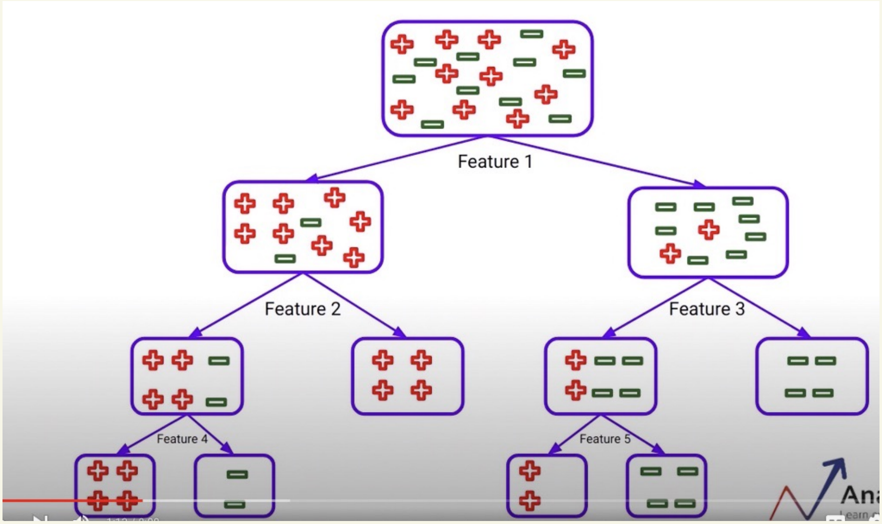
- Above Average node:
  - Mean = (8*1 + 6*0) / 14 = 0.57
  - Variance = $[8*(1-0.57)^2 + 6*(0-0.57)^2] / 14 = 0.245$
- Below Average node:
  - Mean = (2*1 + 4*0) / 6 = 0.33
  - Variance = $[2*(1-0.33)^2 + 4*(0-0.33)^2] / 6 = 0.222$
- Variance: Performance in Class:
  (14/20)*0.245 + (6/20)*0.222 = 0.238



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
* HW: fit a decision tree NFL win prob. model. What goes wrong?
* Decision trees are unstable & prone to overfitting.
* How to reduce overfitting? Ask Leo Breiman...