DS 7331 – Machine Learning I

LIVE SESSION - 7

Intro to Trees

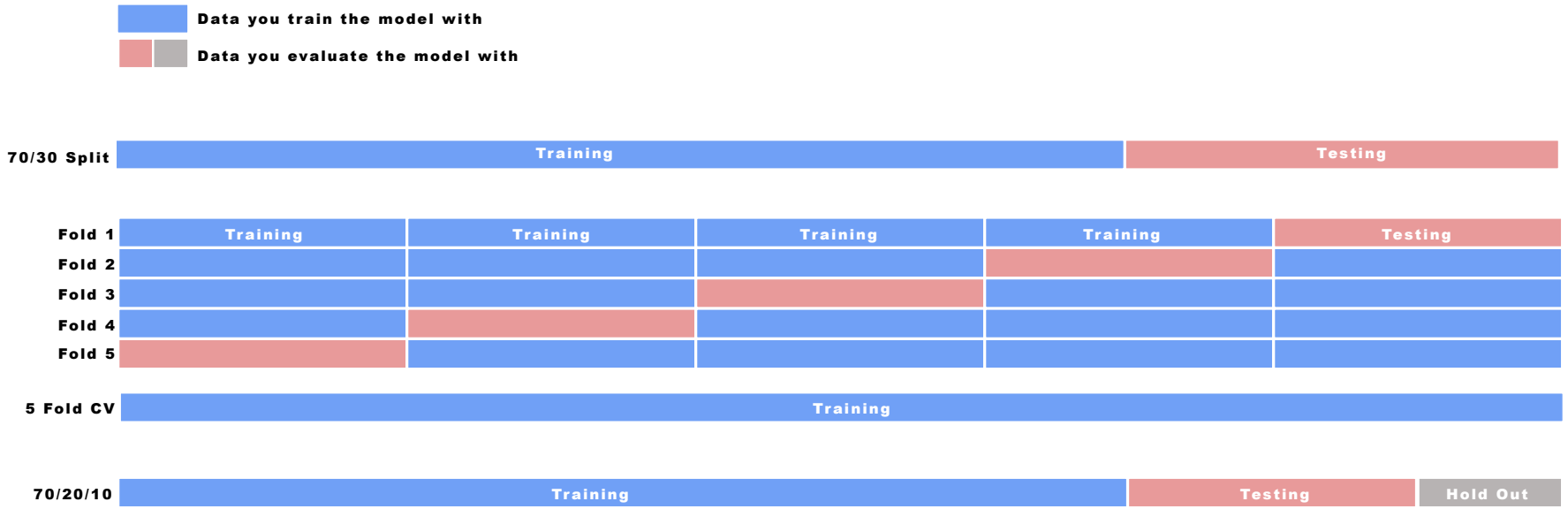
World Changers Shaped Here

SMU

Quick Outline

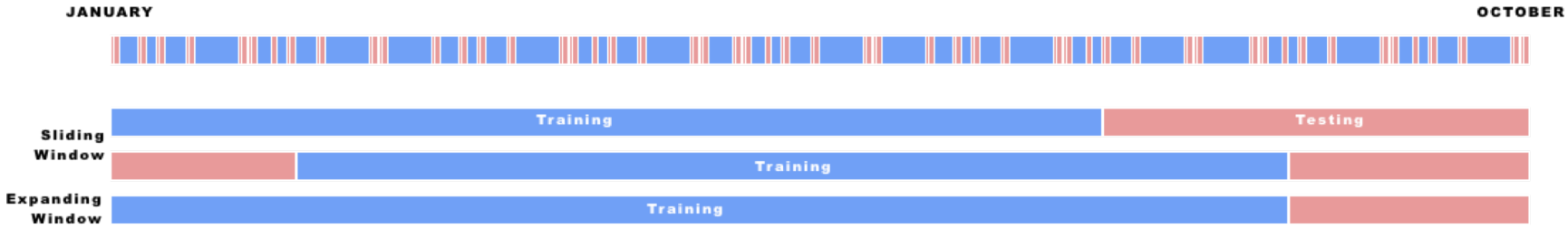
- Sampling & Data Prep
- Simple Trees

Sampling….oh, let's count the ways.



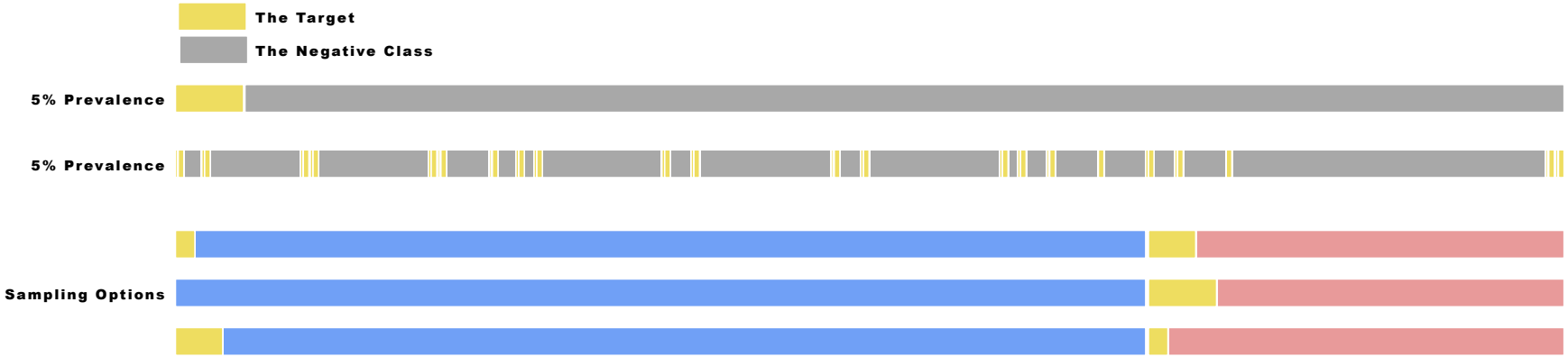
- The simple 70/30-split or a 5/10-fold cross validation will account for practically all use cases; I only mention the 70/20/10 for transparency

Dealing with 'temporal' data can be tricky….see why?



- If we used random sampling to train a model impacted by time, we may use the historical 'future' to evaluate performance on the historical 'past'
- In data science terminology, that is one component of "data leakage" (i.e., leaking data about the target into your training data)
- Ultimately, be super careful when dealing with time-oriented data

Oh yeah, and another note on sampling….



- Stratification allows us to maintain our target's "prevalence" rate throughout our sampling sets (ex. Our target class is equally represented in both the training and testing sets, or throughout the cross validation process)
- Sometimes, you may want to include additional strata beyond the target (ex. Demographics) to aid in the creation of equitable and fair models

The ‘gist’ of Cross Validation

- Splits your data into 3-5 sets (more is fine, but takes time) or partitions
- Sets aside each set as a test data split, builds a model on the other sets.
  - Repeats for each step (in essence it builds your model 3-5 times)
  - Evaluate by averaging your results (or look for consistency in results)
  - Primary aim is to prevent overfitting

Validation ‘does’

- Shuffle your data (where it makes sense)
- Make your shuffle repeatable
  - Set a seed or create a “sampling” feature
- Balance or weight your classes (or pick another algorithm) – choose wisely!
  - Class imbalance becomes a problem around 10:1
  - Stratification is important

Metrics: Classification

- **Accuracy** – how many did I get right?
- **Precision** – how many correct “true predictions” divided by how many in total I predicted to be true
- **Recall** – how many correct “true predictions” divided by all true records
- **F1** – Balanced precision and recall
- **Type I error** (false positive)
- **Type II error** (false negative)

Metrics: Classification (Illustration)

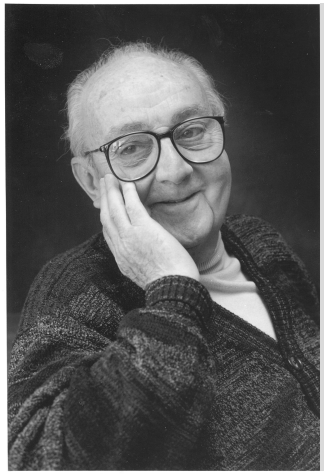
		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Source: Wikipedia

Tips and Tricks (General)

- **Scale your data**
  - Your training data only (use the scales on your testing)
  - helps when interpreting variable importance
- **Ensembles outperform single models (but come with a price)**
  - Non-correlated ensembles are better than correlated ensembles
- **There is no magic formula – "*There is no free lunch!*"**

Remember



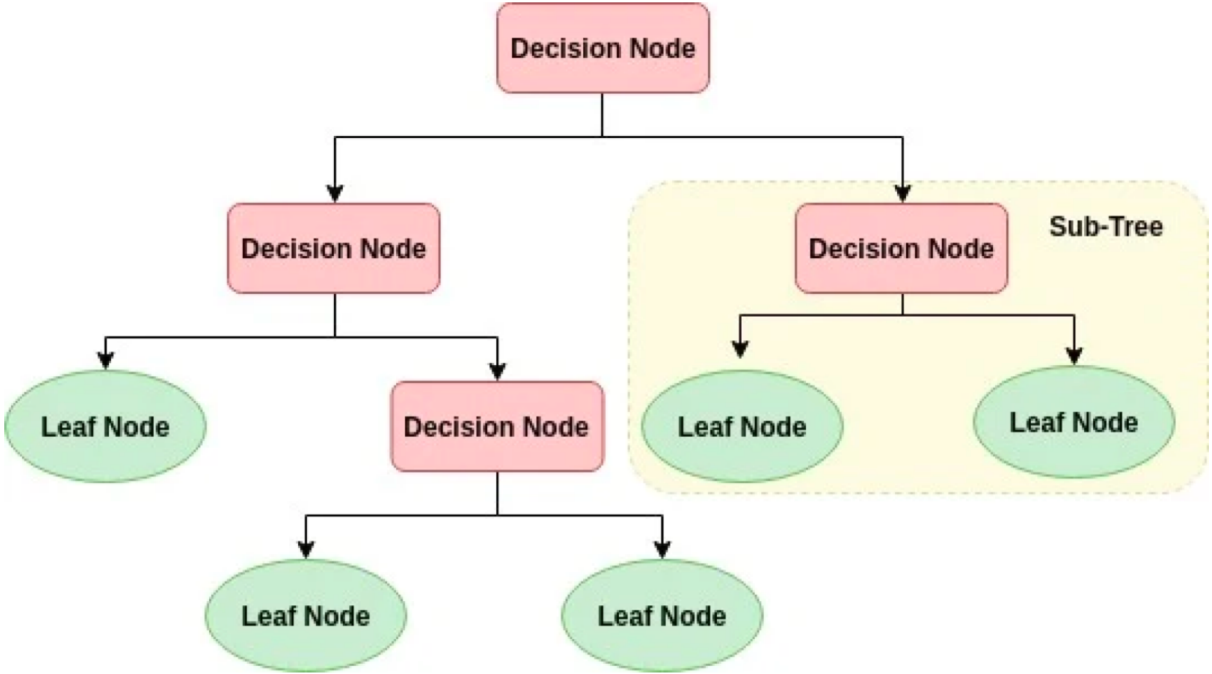
“All models are wrong but some are useful.”

George Box, famous statistician

So how do we make a useful model

- Choose an appropriate sampling method (CV works well)
- Folds are stratified and randomized (unless dealing with temporal data)
- Data is scaled
- Don’t reinvent the wheel – use packages!
  - ”Science experiments” take time and don’t scale
  - Much faster methods to solve/approximate solution (get to a baseline)
  - How the problem is solved is less important than what the model says

What is a tree-based model?



Source: DataCamp

Classification & Regression Trees (CART)

- Inherently non-linear
- Approximate value by a hypercube
- Very interpretable
- Methods for building trees can be obscure
  - Linear models are easy to understand/solve
  - Most people don't know the "under the hood" methodology for trees

Building a Tree

- Introduce “Entropy” (aka randomness, related to order)
- Introduce “Information Gain” (a loss of entropy or introduction of order)
- Remember – even though we are going to rules based decision trees, those rules are based on statistics!
- Low entropy is good! Entropy Zero -> no randomness, no “unpredictability”
- High entropy (~1) – much “unpredictability”
- Information Gain is the negative change in entropy, if you DECREASE unpredictability, you increase information extracted

Building a Tree (50/50 example)

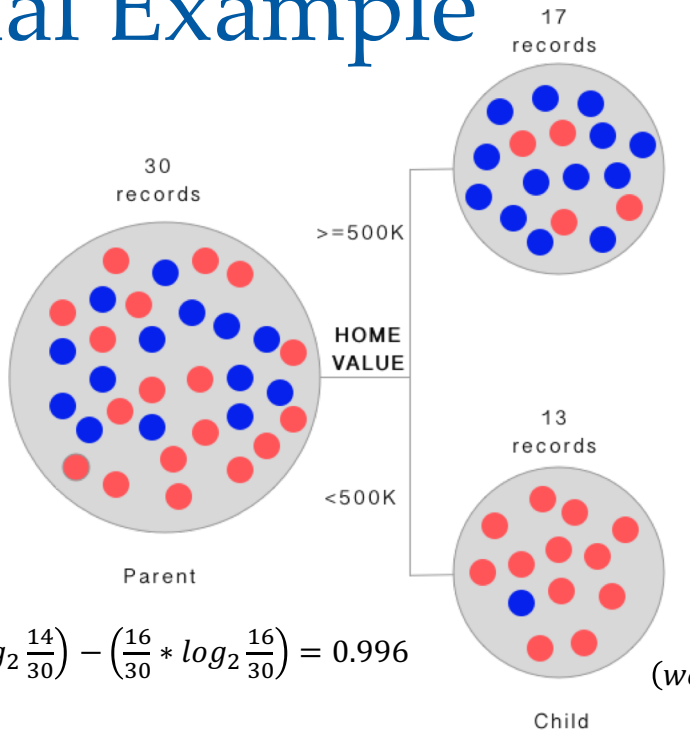
- Entropy (H)

$$H = -\sum_{1}^{n} P(x_i) log_2 P(x_i)$$

$$H = -\sum_{1}^{n} P(x_i) log_2 P(x_i) = \underbrace{-0.5(-1)}_{\text{Heads}} \underbrace{- 0.5(-1)}_{\text{Tails}} = 1; random\ results$$

- Why “Entropy”? The story goes that Claude Shannon didn't know what to call his new information measure, so he asked von Neumann, who said `You should call it entropy ... [since] ... no one knows what entropy really is, so in a debate you will always have the advantage’ (Source)

A Visual Example



$$impurity = -\left(\frac{13}{17} * log_2 \frac{13}{17}\right) - \left(\frac{4}{17} * log_2 \frac{4}{17}\right) = 0.787$$

$$impurity = -\left(\frac{14}{30} * log_2 \frac{14}{30}\right) - \left(\frac{16}{30} * log_2 \frac{16}{30}\right) = 0.996$$

$$impurity = -\left(\frac{1}{13} * log_2 \frac{1}{13}\right) - \left(\frac{12}{13} * log_2 \frac{12}{13}\right) = 0.391$$

$$\begin{matrix}(weighted)\ impurity \\ of\ children\end{matrix} = -\left(\frac{17}{30} * 0.787\right) - \left(\frac{13}{30} * 0.391\right) = 0.615$$

$$information\ gain = 0.996\ - 0.615 = 0.38$$

So how do I build a Tree

- Recursively partition your data (split your data) and find the maximum REDUCTION in entropy (information gain)
  - Go through your features 1 by 1 and split them to find the max info gain
    - The algorithm bins your data
    - Calculates splits based on the bias; calculates info gain for each split
    - Picks the split that lowers entropy most (note: surrogacy)
    - To prevent overfitting, induce rules to govern split criterion
  - Repeat this procedure until you can no longer increase the info gain by a fixed amount: the **Complexity Parameter**