MACHINE LEARNING OVERVIEW

BROOKE WENIG

M.S. Computer Science (Distributed Machine Learning)

Fluent in 中文

machinelearningninja@gmail.com

LinkedIn

WHEN I'M NOT WORKING...

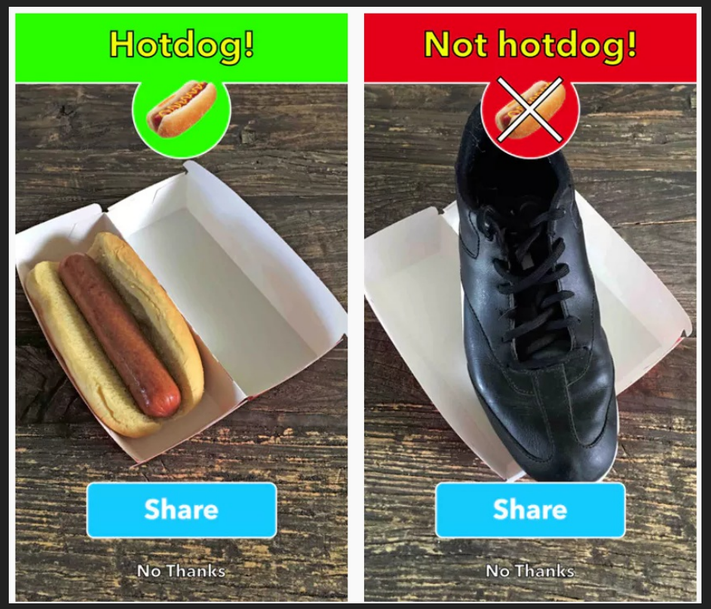
WHAT DOES ML DO?

> *It's tough to make predictions, especially about the future.*

- Yogi Berra

SUPERVISED MACHINE LEARNING

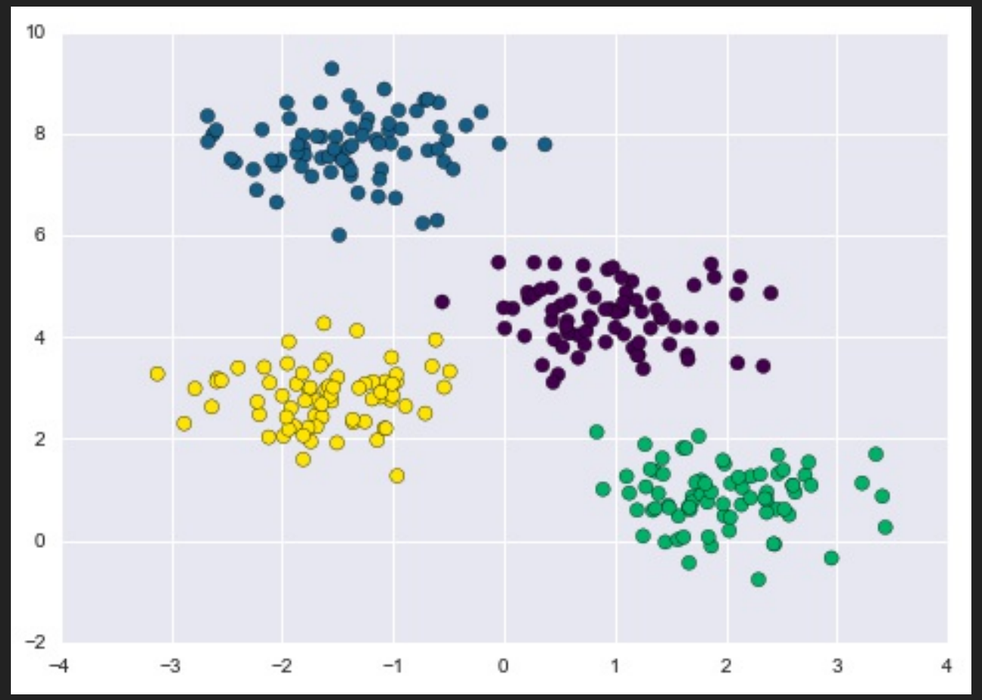
Classification



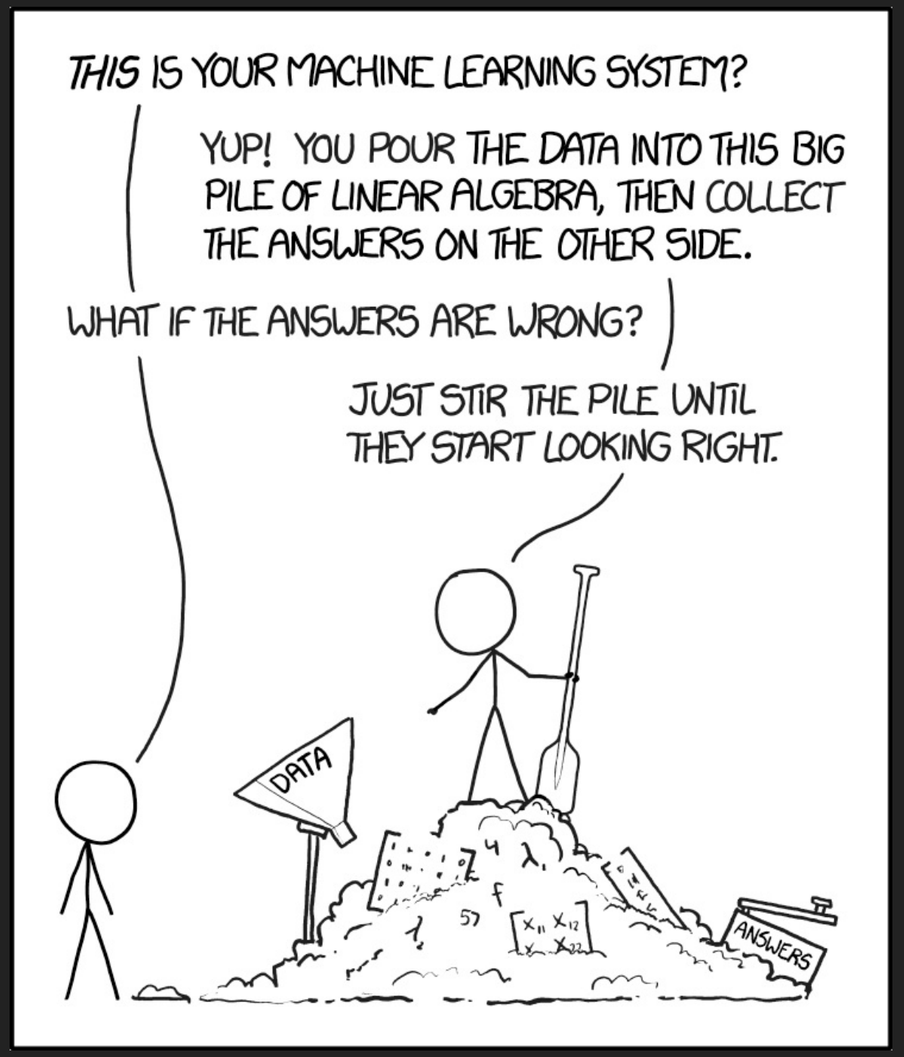
Regression

UNSUPERVISED MACHINE LEARNING

Clustering



> *"All models are wrong; some models are useful."*



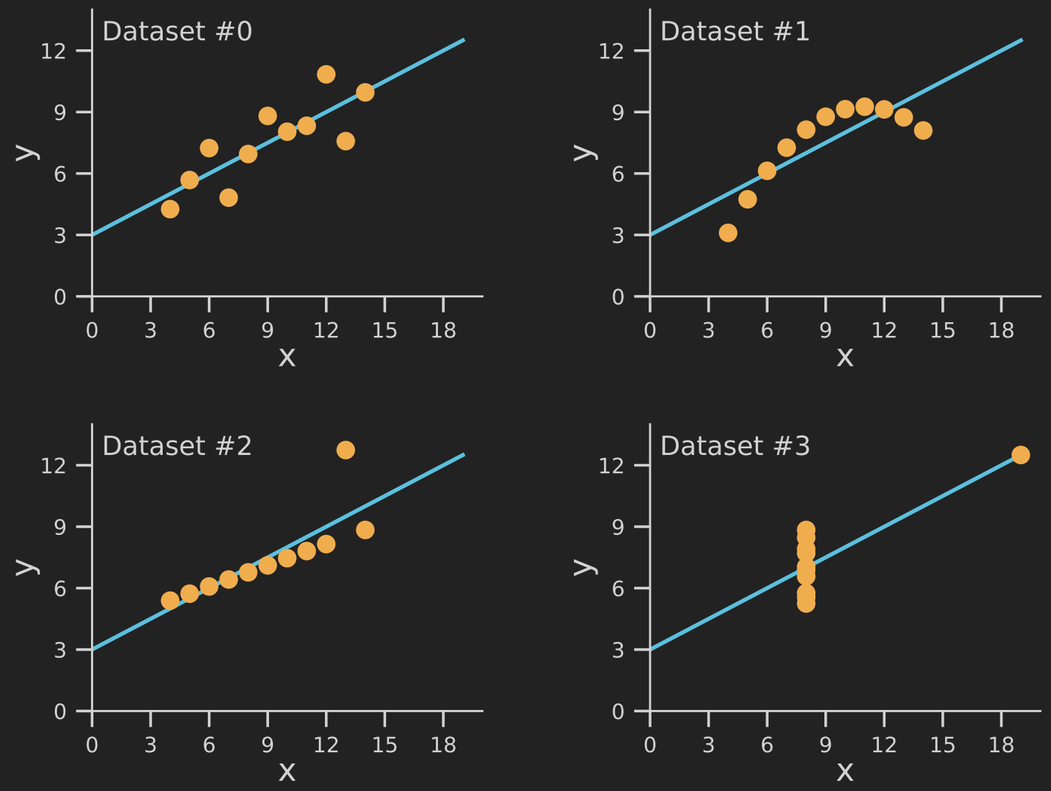
SHOULD WE TRUST NUMBERS?

ANSCOMBE'S QUARTET

Dataset #0		Dataset #1		Dataset #2		Dataset #3	
x	y	x	y	x	y	x	y
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.1	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.1	4	5.39	19	12.5
12	10.84	12	9.13	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89

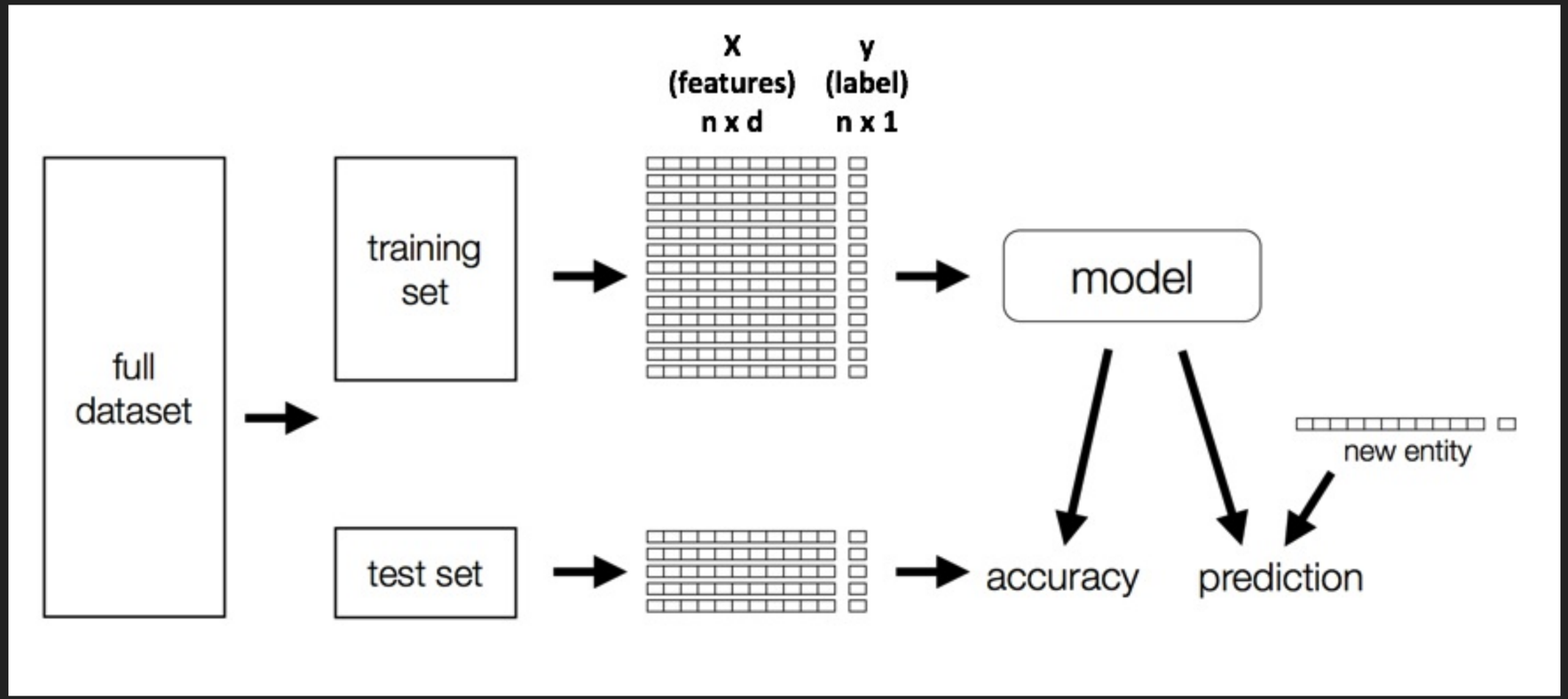
Mean	9	7.5	9	7.5	9	7.5	9	7.5
Variance	11	4.1	11	4.1	11	4.1	11	4.1
Correlation	0.86		0.86		0.86		0.86	
Regression line	y = 3 + 0.5x		y = 3 + 0.5x		y = 3 + 0.5x		y = 3 + 0.5x	

WHY VISUALIZE YOUR DATA?



HOW TO BUILD/EVALUATE MODELS?

BUILD A MODEL

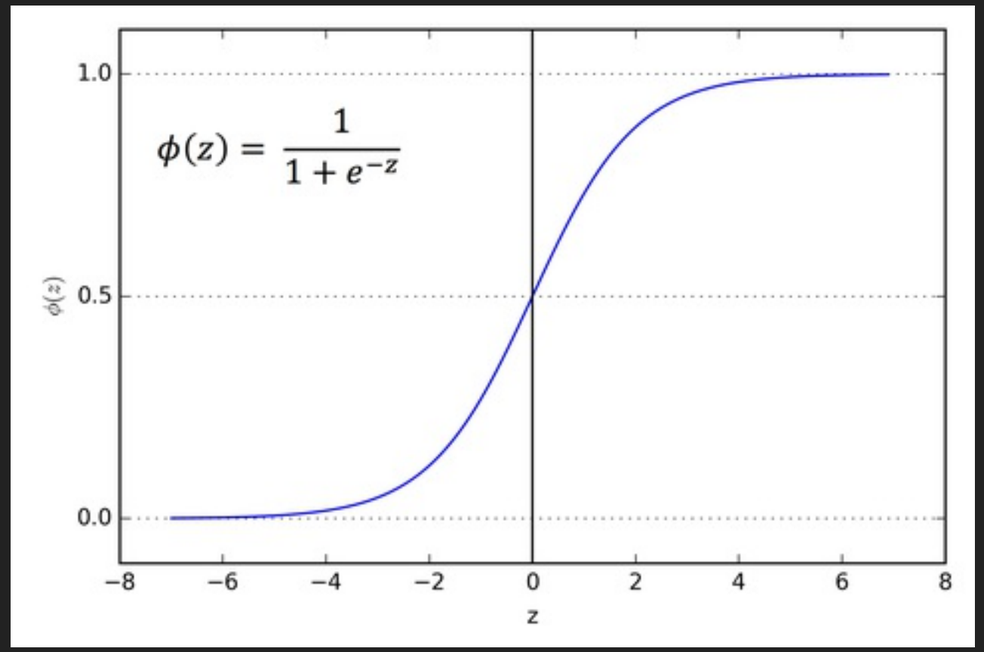


LOGISTIC REGRESSION

BINARY CLASSIFICATION

- Given a set of features, does this data point belong to $y = 1$ or $y = 0$?
- Reminder: Probabilities bounded between 0 and 1
- If P[y = 1 | x] > 0.5, then $\hat{y} = 1$, else $\hat{y} = 0$

LOGISTIC FUNCTION



- Large positive inputs => 1
- Large negative inputs => 0

CONDITIONAL PROBABILITY

Logistic regression uses logistic function to model this conditional probability

- $\mathbb{P}[y = 1 \,|\, \mathbf{x}] = \sigma(\mathbf{w}^\top \mathbf{x})$
- $\mathbb{P}[y = 0 \,|\, \mathbf{x}] = 1 - \sigma(\mathbf{w}^\top \mathbf{x})$

For notational convenience we now define $y \in \{0, 1\}$

MAKE PREDICTIONS

To make class predictions, we need to convert probabilities to values in $\{0, 1\}$

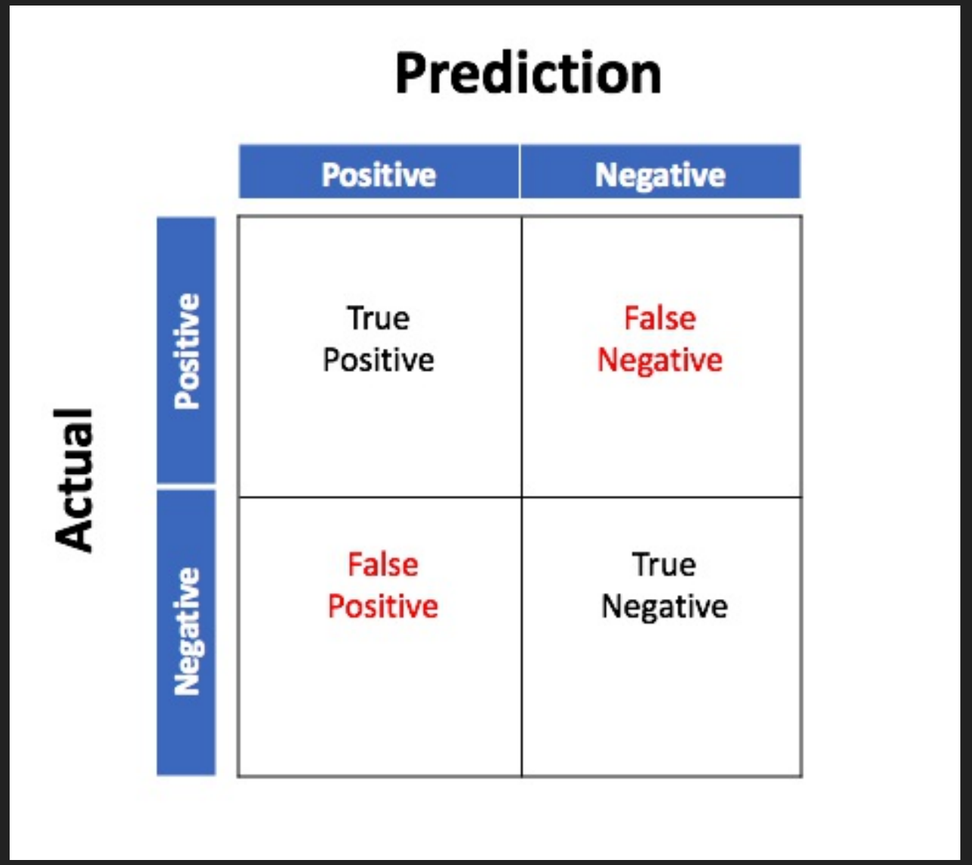
We can do this by setting a threshold on the probabilities

- Default threshold is 0.5
- $\mathbb{P}[y = 1 \,|\, \mathbf{x}] > 0.5 \implies \hat{y} = 1$

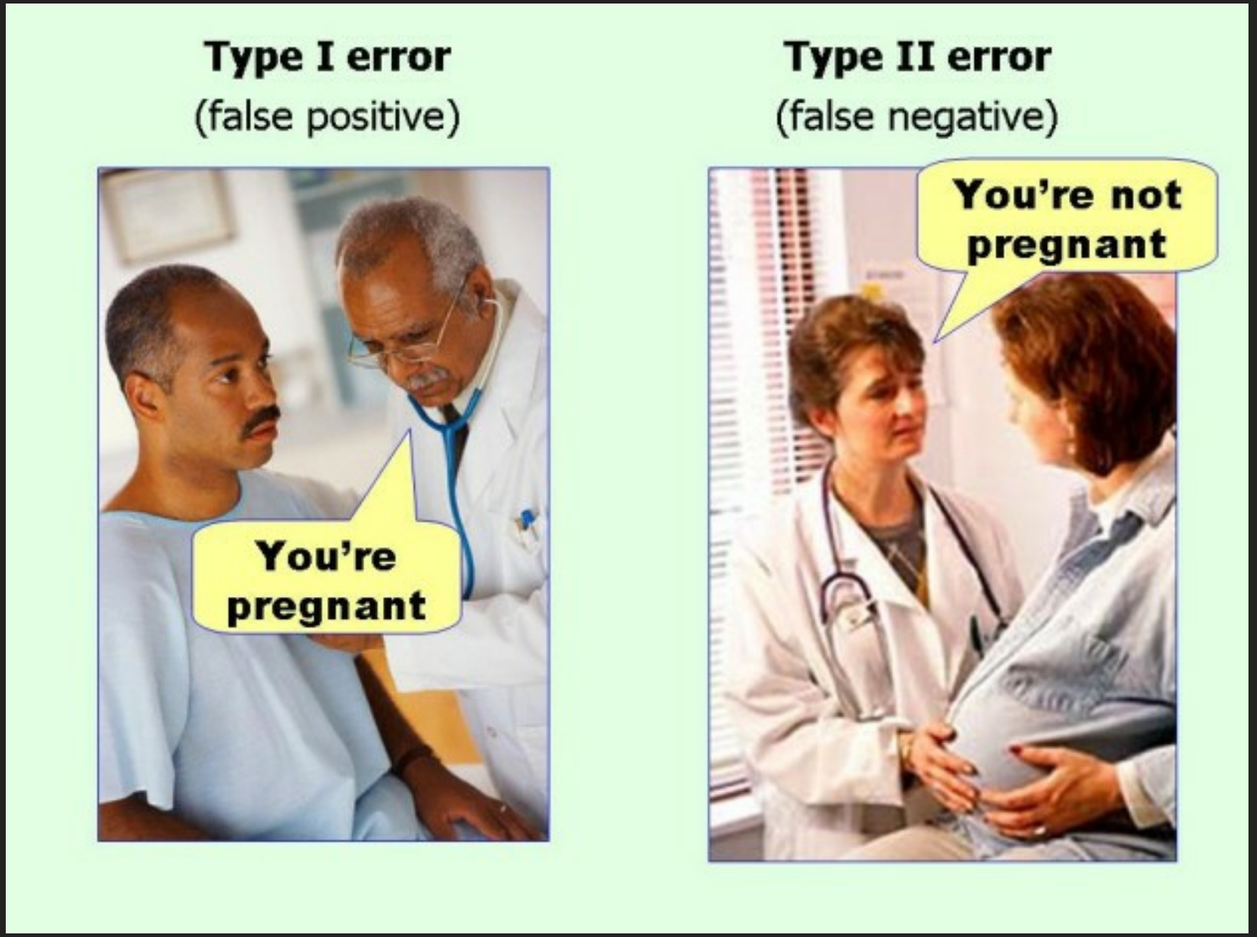
Example: Predict rain from **t**emperature, **c**loudiness, **h**umidity

- $\mathbb{P}[y = \text{rain} \,|\, \text{t} = 14°\text{F}, \text{c} = \text{LOW}, \text{h} = 2\%] = .05 \qquad \hat{y} = 0$
- $\mathbb{P}[y = \text{rain} \,|\, \text{t} = 70°\text{F}, \text{c} = \text{HIGH}, \text{h} = 95\%] = .9 \qquad \hat{y} = 1$

CONFUSION MATRIX



CHANGE THRESHOLD?



NON-NUMERIC FEATURES

Categorical

- No intrinsic ordering
- e.g. Gender, Country, Occupation

Ordinal

- Relative ordering, but inconsistent spacing between categories
- e.g. Excellent, good, poor

HOW TO HANDLE NON-NUMERIC FEATURES?

Option 1: Use methods (like decision trees) that support these features

Option 2: Convert these features to numeric features

How do we do this?

ONE IDEA

Create single numerical feature to represent non-numeric one

Categorical features:

- Country categories = {'CA', 'FRA', 'USA'}
- 'CA' = 1, 'FRA' = 2, 'USA' = 3

Implies France is between Canada and USA!

ONE HOT ENCODING

Create a ‘dummy’ feature for each category

'CA' => [1, 0, 0], 'FRA' => [0, 1, 0], 'USA' => [0, 0, 1]

No spurious relationships!

STORAGE SPACE

Ok, so that works if we only have a few columns, but what if we had many columns?

SPARSE VECTORS

Size of vector, indices of non-zero elements, values

```
DenseVector(0, 0, 0, 7, 0, 2, 0, 0, 0, 0)
SparseVector(10, [3, 5], [7, 2])
```

ESTIMATOR/TRANSFORMER/PIPELINE

ESTIMATOR

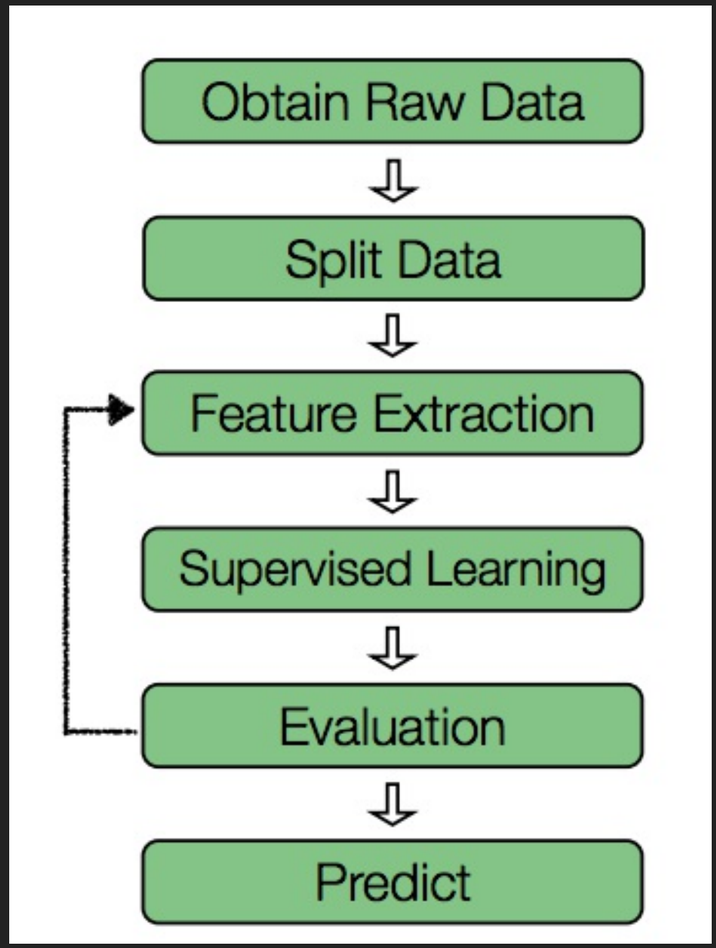
- An algorithm that trains on a dataset and produces a model
- Implements a fit() method

TRANSFORMER

- Converts one DataFrame into another, generally by appending one or more columns
- Implements a transform() method

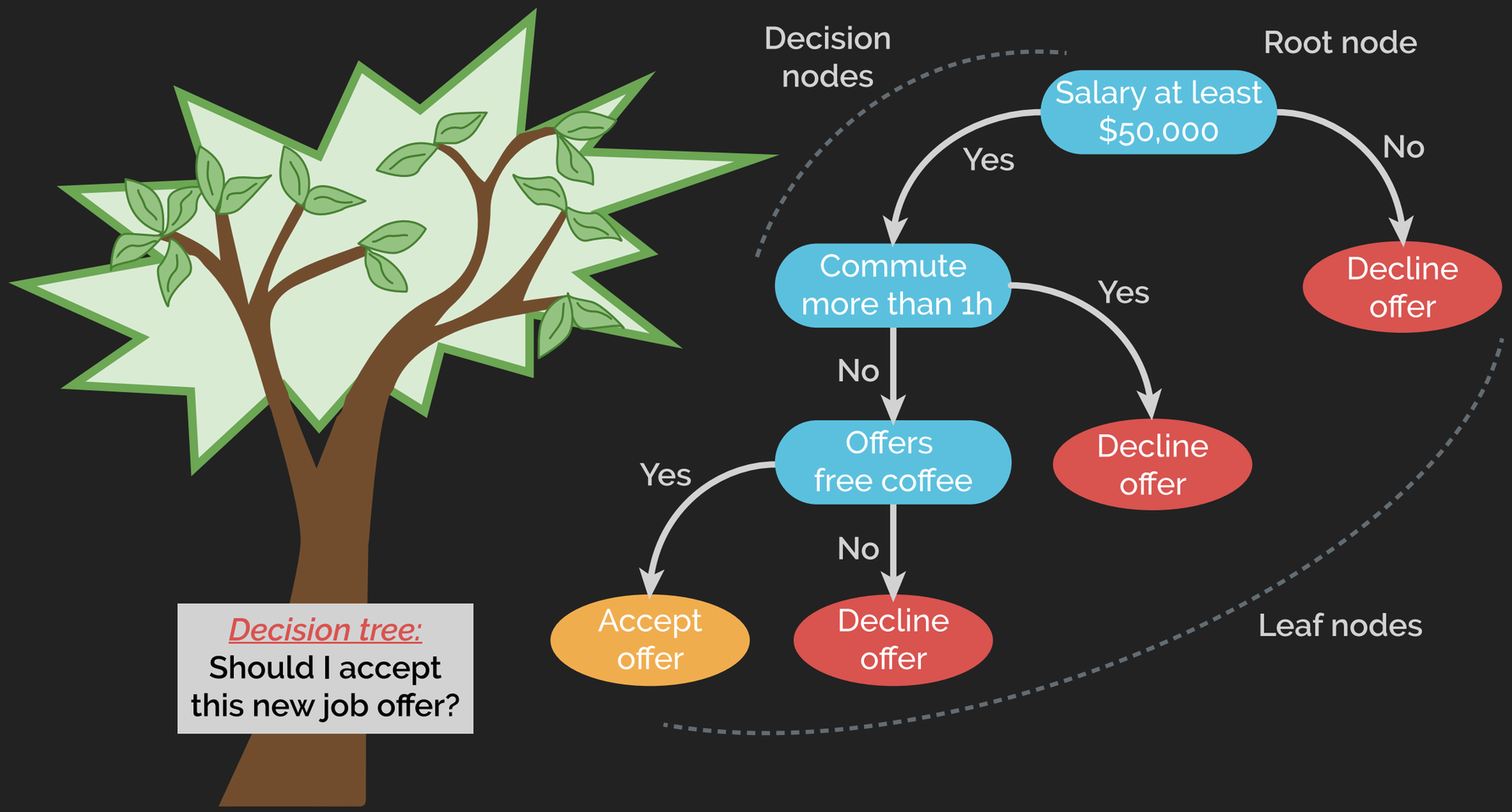
PIPELINE

- Sequence of stages (Transformers and Estimators)
- Great way to organize code!



DECISION TREES

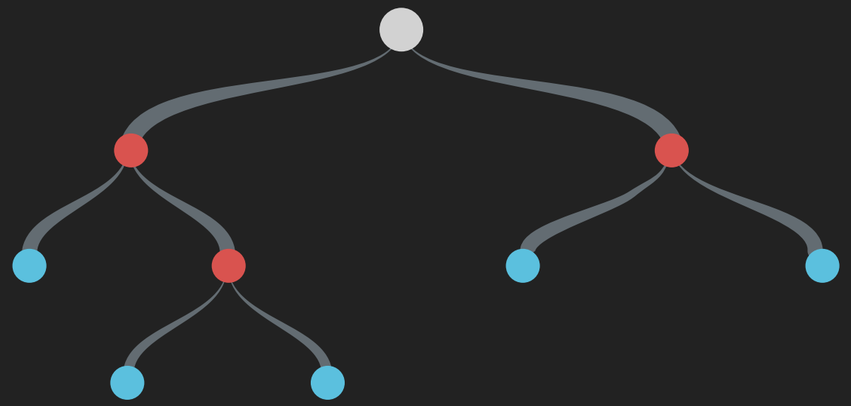
DECISION MAKING



Slide credit G. Calmattes

DECISION TREE

Really, it's an upside tree (leaves on bottom)



Leaf:

- Terminal node (no child)

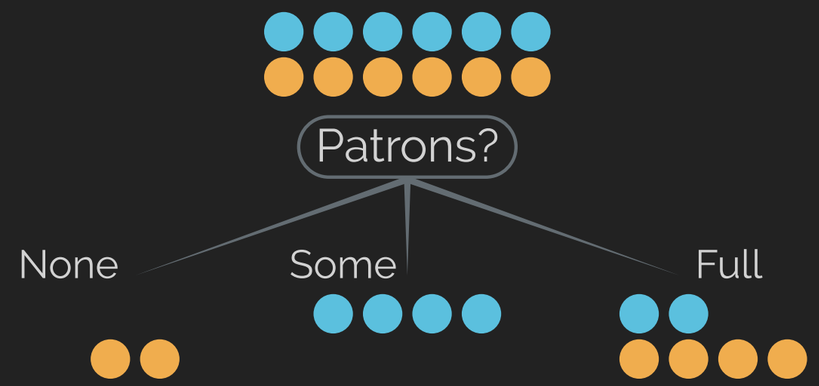
Internal Node:

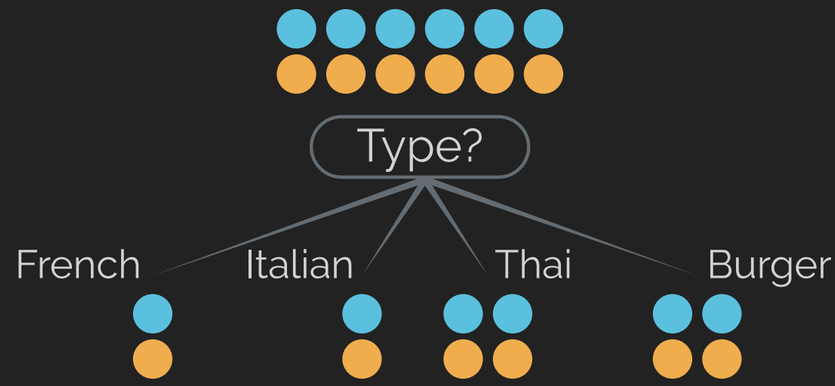
- Splits the predictor space

CHOOSING A RESTAURANT

Examples	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X_1	T	F	F	T	Some	$$$	F	T	French	0-10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30-60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0-10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10-30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0-10	T
X_7	F	T	F	F	None	$	T	F	Burger	0-10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0-10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10-30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0-10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30-60	T

WHICH ATTRIBUTE TO SPLIT?

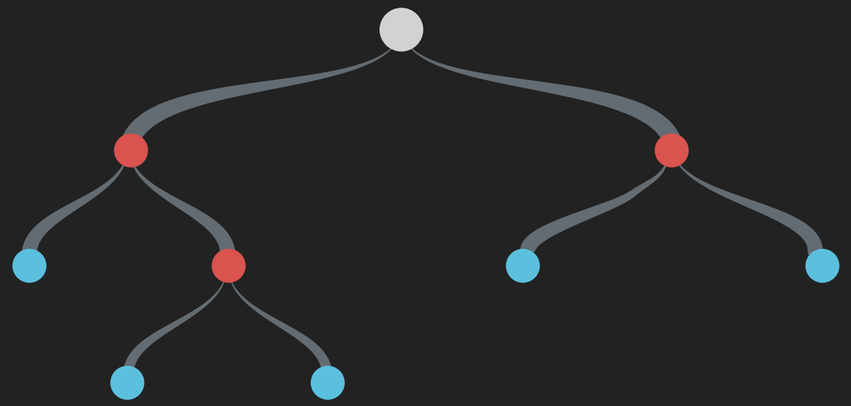




Patrons is a better choice because it gives more information about the classification

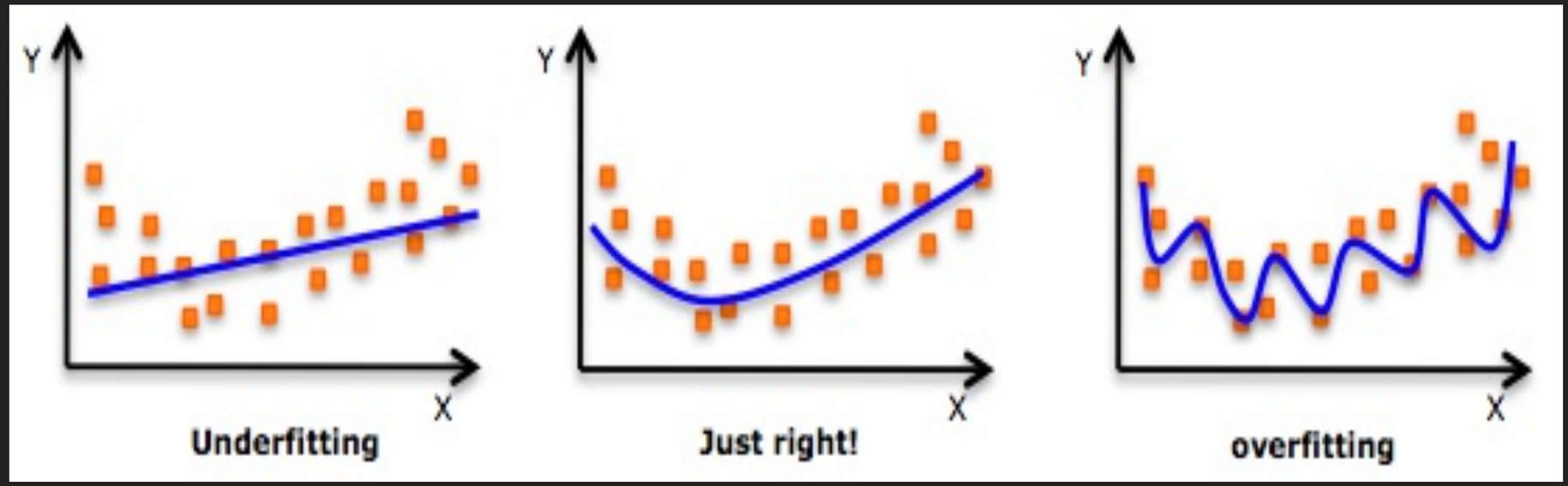
DEPTH

Depth: Longest path from root to a leaf node



- If too deep, can overfit
- If too shallow, can underfit

UNDERFITTING VS. OVERFITTING



EVALUATE PREDICTIONS

Measure "closeness" between label and prediction

- e.g. When predicting someone's weight, better to be off by 2 lbs instead of 20 lbs

Evaluation metrics:

- Loss: $(y - \hat{y})$
- Absolute loss: $|y - \hat{y}|$
- Squared loss: $(y - \hat{y})^2$

EVALUATION METRIC: RMSE

$$Error = (y_i - \hat{y}_i)$$

$$SE = (y_i - \hat{y}_i)^2$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

TRAIN VS. TEST RMSE

- Which is more important? Why?

DECISION TREE #1

COMPARISON

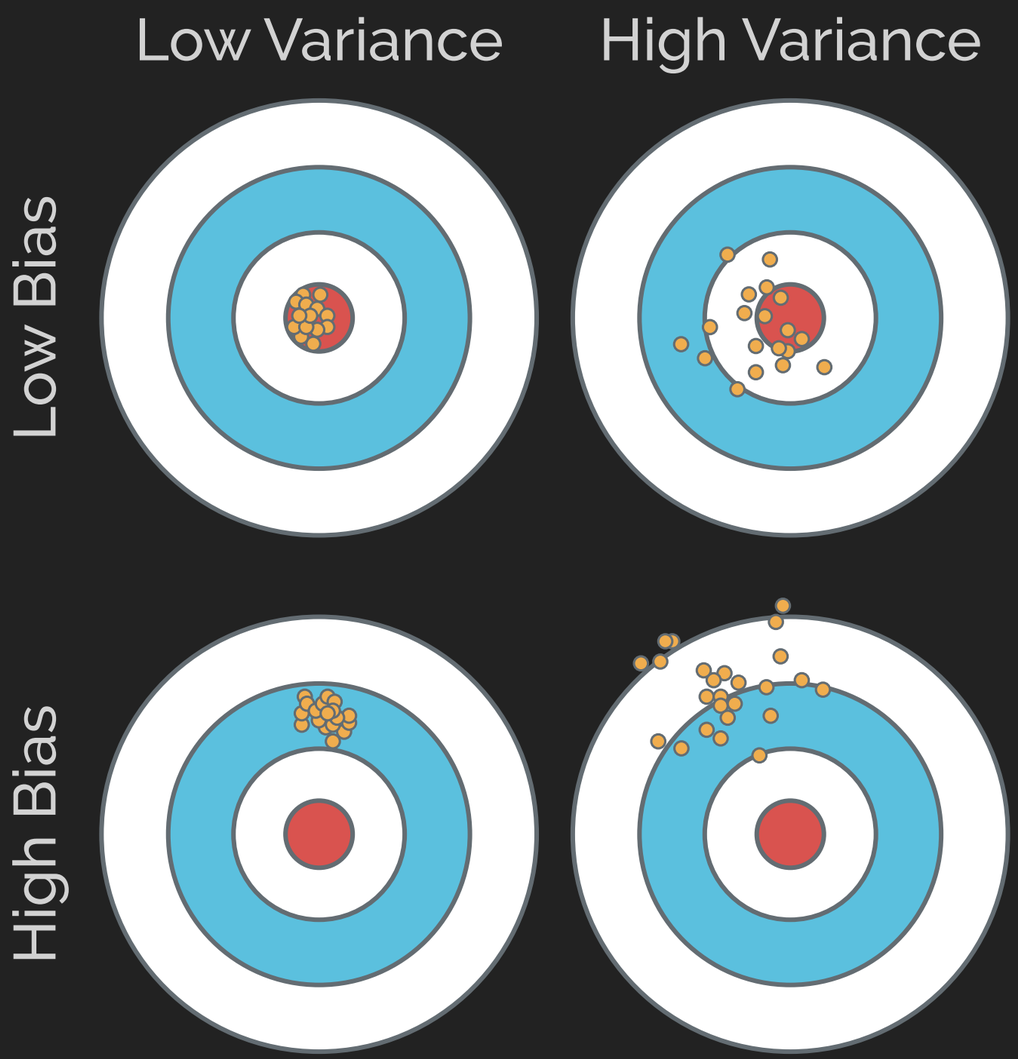
Pros

- Interpretable
- Simple
- Flexible (different data types)
- Classification or Regression

Cons

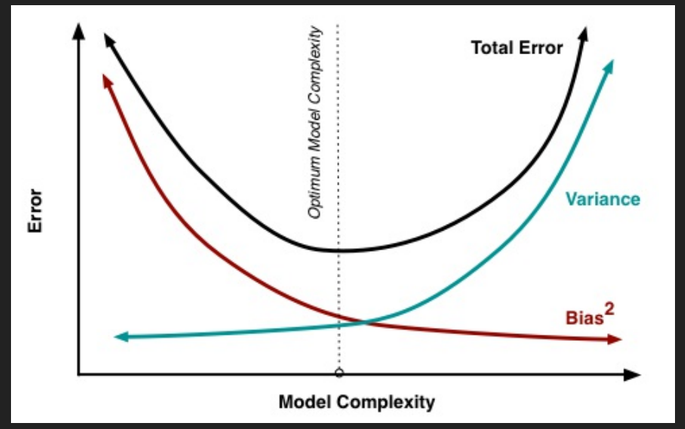
- Poor accuracy
- High variance

BIAS VS VARIANCE



BIAS-VARIANCE TRADE-OFF

$$Error = Variance + Bias^2 + noise$$

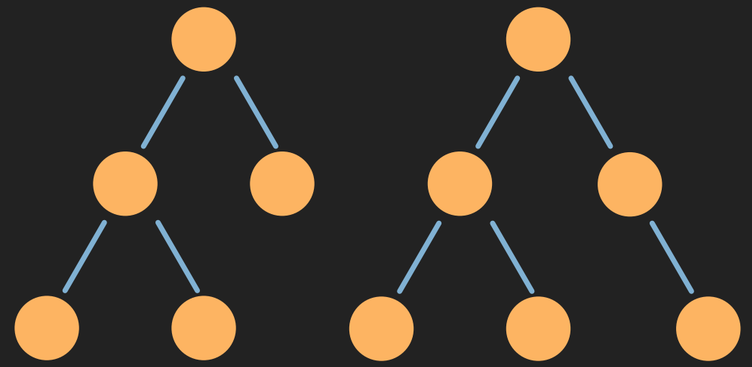


- Reduce bias: Build more complex models
- Reduce variance: Use a lot of data or simple model
- How to reduce the error caused by noise?

BAGGING

- Averaging a set of observations reduces variance.

- But we only have one training set ... or do we?



BOOTSTRAP

Simulate new datasets:

- Take samples (with replacement) from original training set
- Repeat n times

BOOTSTRAP VISUALIZATION

Sample has n elements.

Probability of getting picked: $\frac{1}{n}$

Probability of not getting picked: $1 - \frac{1}{n}$

- If you sample n elements with replacement, the probability for each element of not getting picked in the sample is: $(1 - \frac{1}{n})^n$
- As $n \rightarrow \infty$, this probability approaches $\frac{1}{e} \approx .368$

Thus, 0.632 of the data points in your original sample show up in the Bootstrap sample (the other 0.368 won't be present in it)

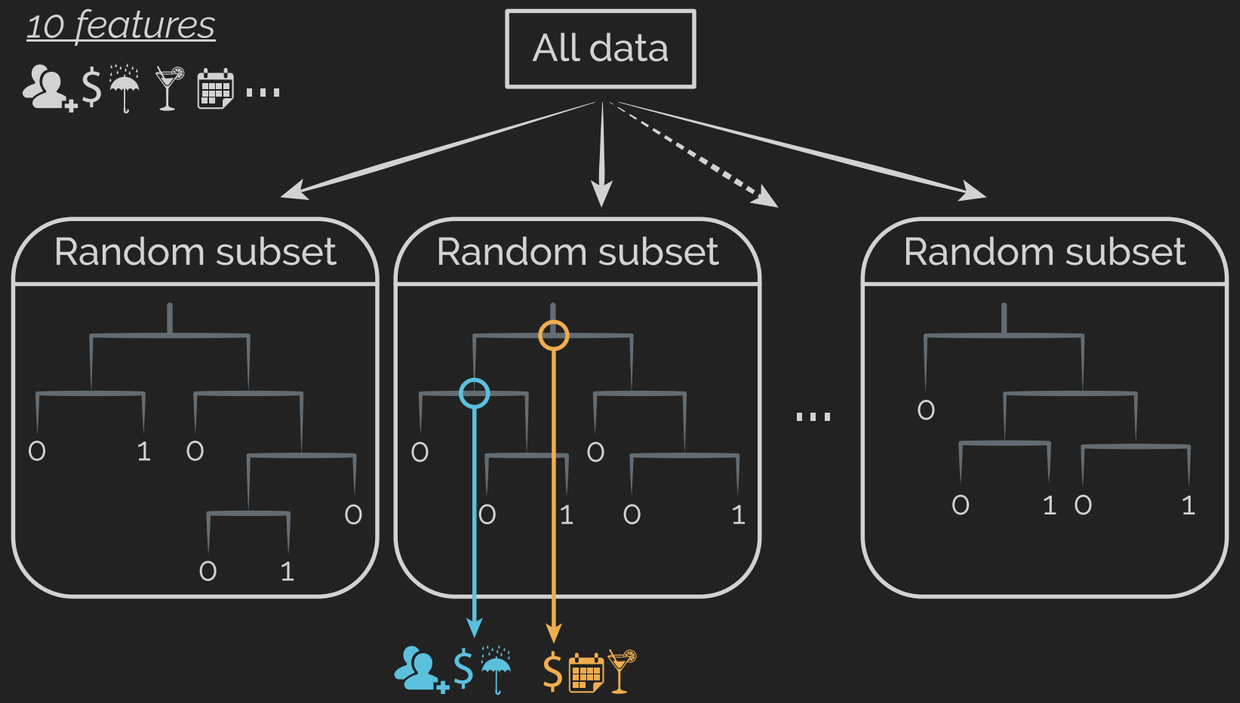
BAGGING

- Train a tree on each bootstrap sample, and average their predictions (Bootstrap Aggregating)

RANDOM FORESTS

Like bagging, but removes correlation among trees.

- At each split, considers only a subset of predictors.

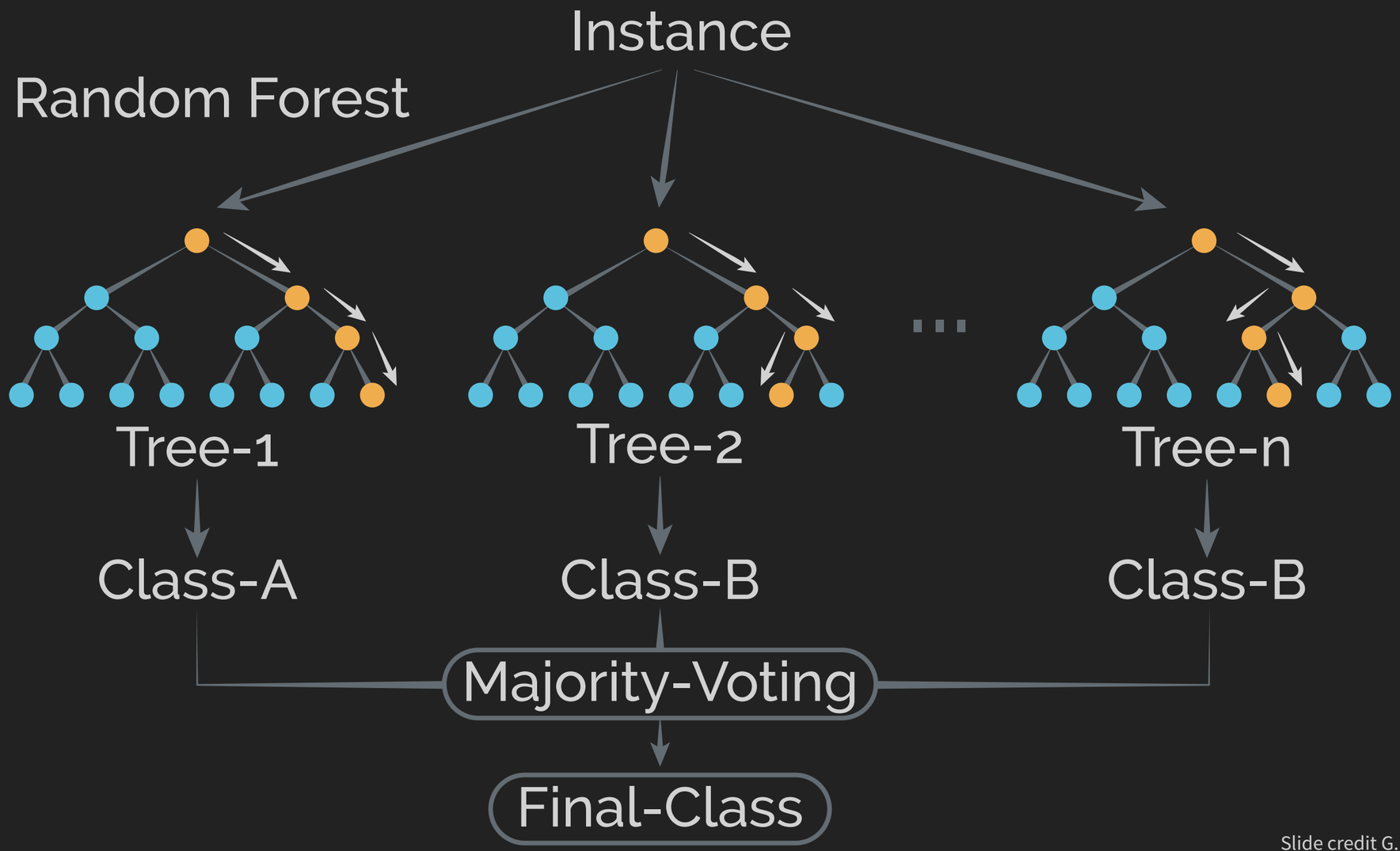


Notes:

Random forest typically considers $\sqrt{\text{number of features}}$ at each split (if 10 features, then it considers $\sqrt{10} \approx 3$ features), and picks the best one.

Slide credit G. Calmattes

RANDOM FORESTS



Slide credit G. Calmattes

HYPERPARAMETER SELECTION

- Which dataset should we use to select hyperparameters? Train? Test?

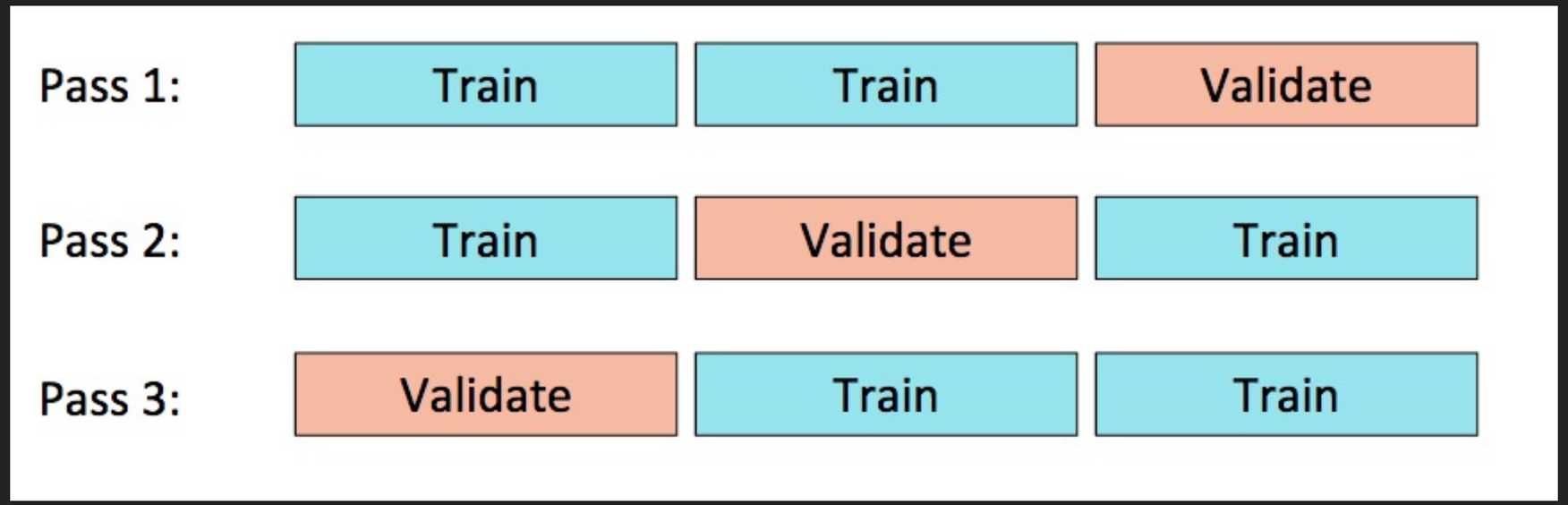
VALIDATION DATASET

Split the dataset into three!

- Train on the *training set*
- Select hyperparameters based on performance of the *validation set*
- Test on *test set*

What if you don't have enough data to split in 3 separate sets?

K-FOLD CROSS VALIDATION



GRID SEARCH

- Try various hyperparameter settings to find the optimal combination

Tree Depth	Number of Trees
5	2
8	4

- How many models will this build?

SUMMARY

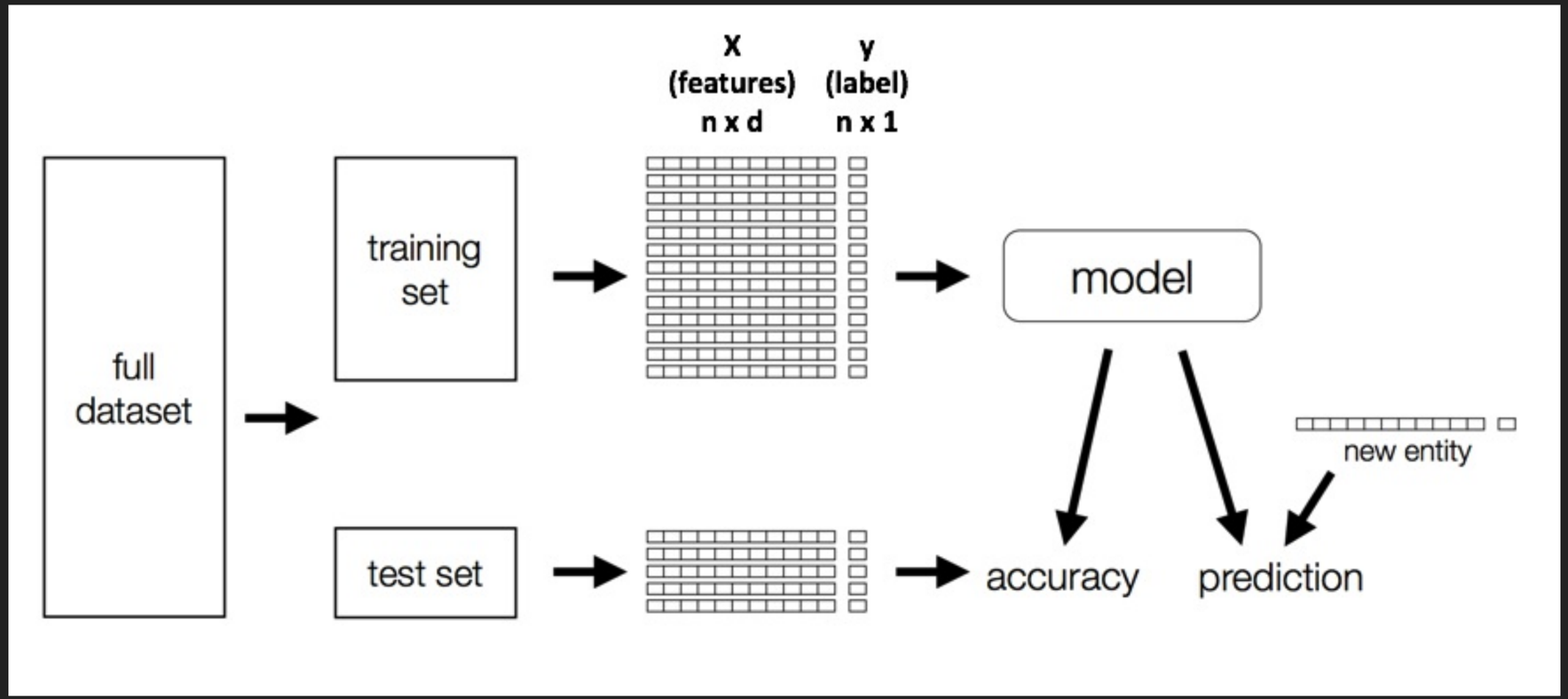
SPARK

- RDD, DataFrame, Dataset
  - Immutable once constructed
  - Track lineage information
  - Operations on collection of elements in parallel
- Transformations vs. Actions

WHY LAZY?

- Not forced to load all data at step #1
- Apply various optimizations

BUILD A MODEL



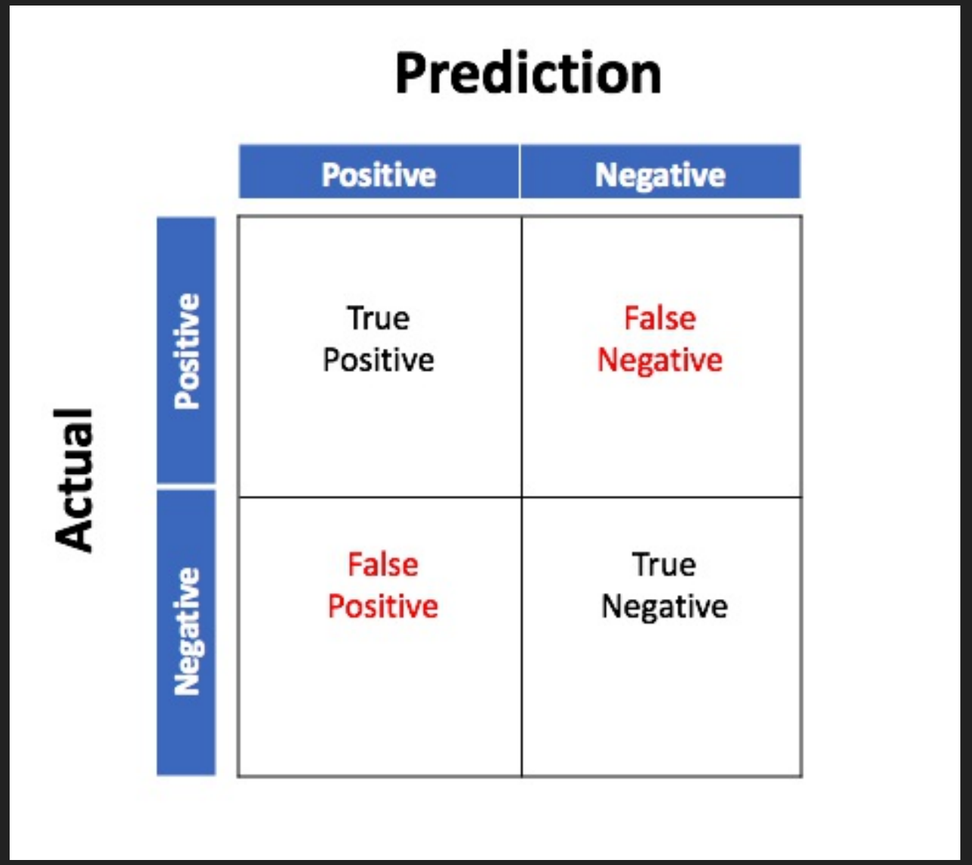
LOGISTIC REGRESSION

- Interpret as probability

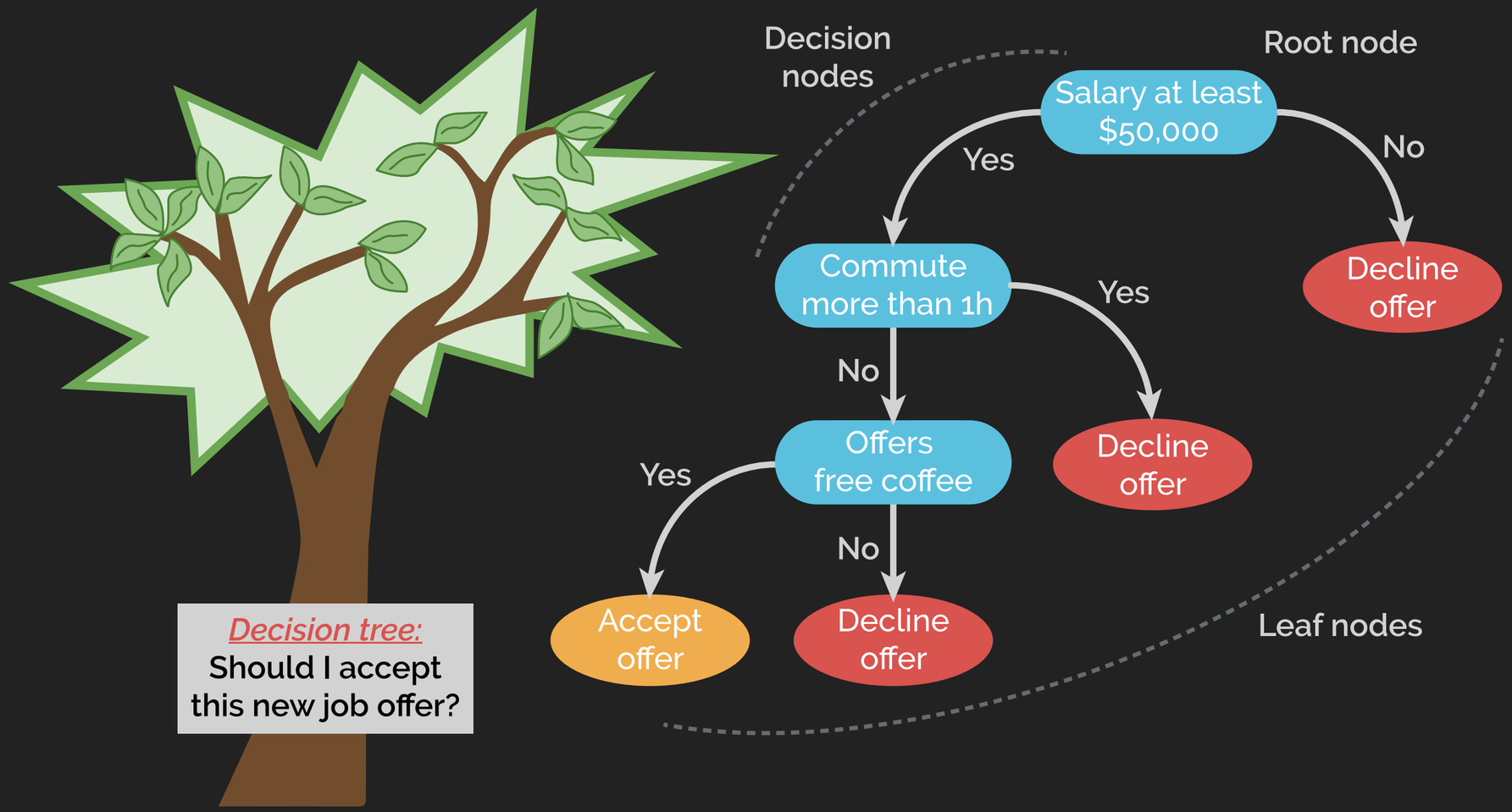
$P[y = 1 \mid x]$

- NLP + Pipelines

CONFUSION MATRIX

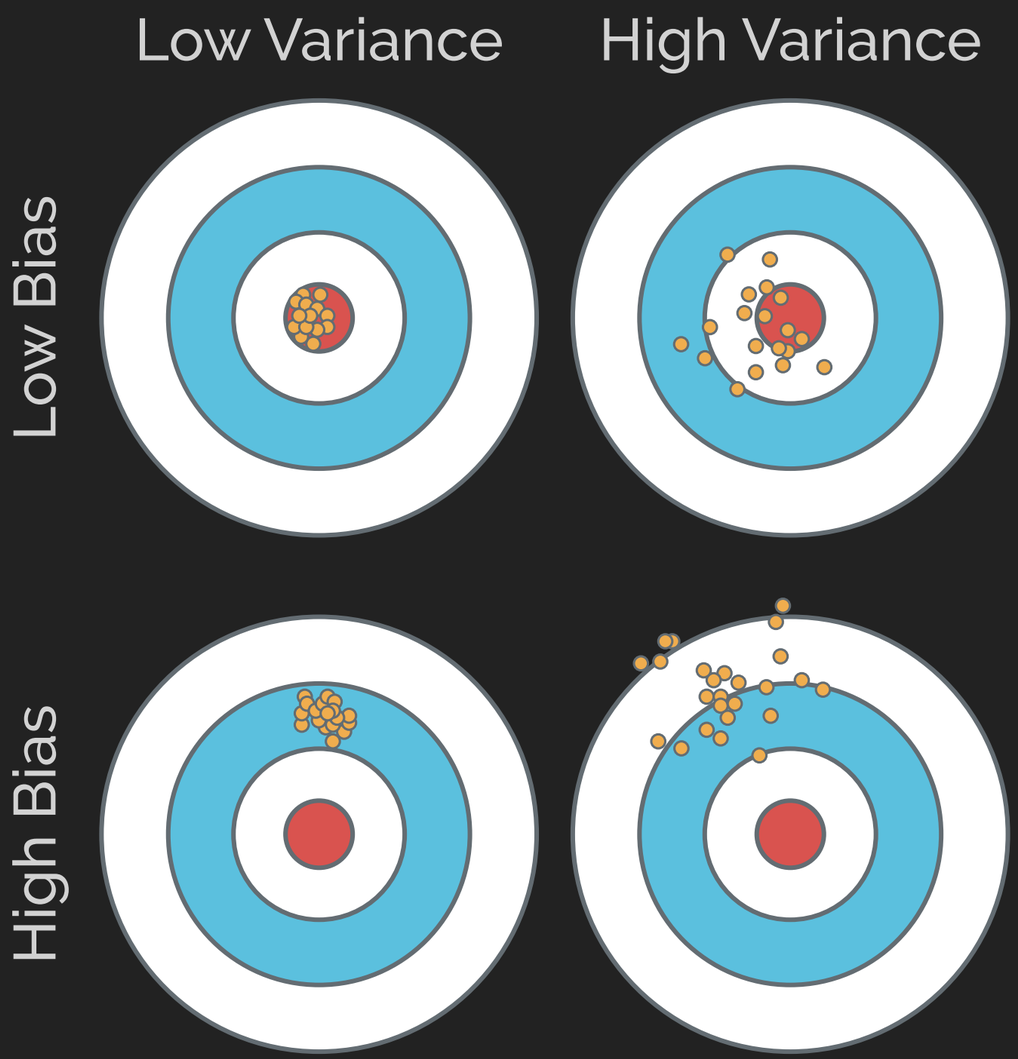


DECISION TREES



Slide credit G. Calmattes

BIAS VS VARIANCE



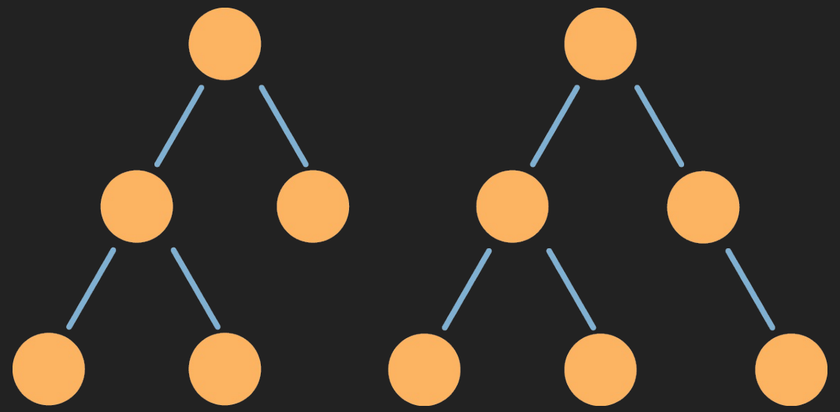
BOOTSTRAP

Simulate new datasets:

- Take samples (with replacement) from original training set
- Repeat n times

BAGGING

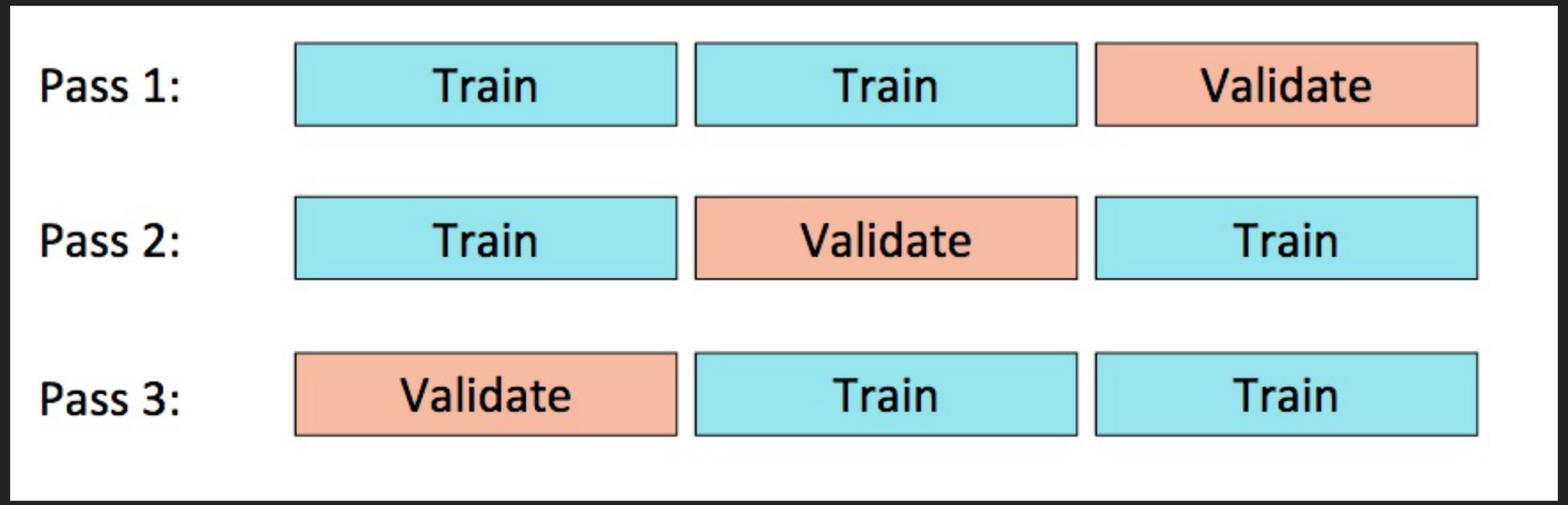
Purpose of bagging?



RANDOM FORESTS

- What is random about the trees?
- Parallelizable?

K-FOLD CROSS VALIDATION



GRID SEARCH

- Try various hyperparameter settings to find the optimal combination

Tree Depth	Number of Trees
5	2
8	4

PIPELINE

- Put all of these feature transformation and model building steps into a single line of code!

WHICH MODEL TO CHOOSE?

- Some models are more costly to train
- Others are more costly to test
- Need for interpretability?

WHY NOT NEURAL NETWORKS?

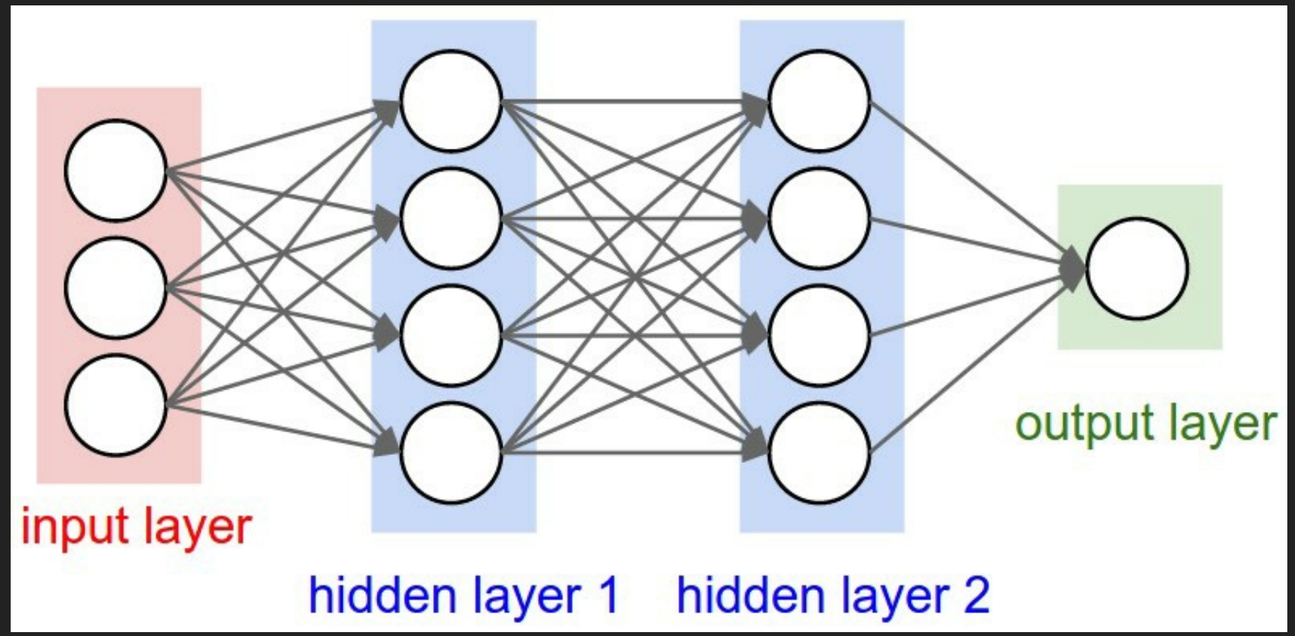
LET'S BUILD A CLASSIFIER

WHAT DID THE NEURAL NETWORK LEARN?

A GRASS CLASSIFIER!

NETWORK STRUCTURE

- Input layer (fixed)
- Zero or more hidden layers
- Output layer (fixed)



PERFORMANCE

They perform very well when unseen data similar to training data

Main drawbacks:

- Lack of interpetability
- Time to train
- Overfit

THANK YOU

EXTENSION: GRADIENT BOOSTED DECISION TREES

- Sequential method
  - Each tree added to improve upon current forest (shrunken version)
- Fit trees to residuals (on first iteration, residuals are the true predictions)
- Idea: build tree more slowly
- These trees are NOT independent of each other

GBDT

Y	Prediction	Residual
40	35	5
60	67	−7
30	28	2
33	32	1
25	27	−2

Y	Prediction	Residual
5	3	2
−7	−4	−3
2	3	−1
1	0	1
−2	−2	0

Y	Prediction
40	38
60	63
30	31
33	32
25	25

KAGGLE

- Gradient boosted trees (xgboost in particular) are the most winning Kaggle method