# Lecture 11: Basic Classification Techniques

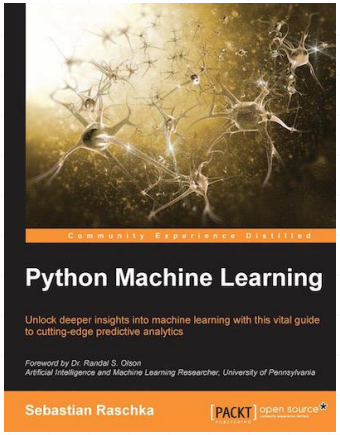
Course: Biomedical Data Science

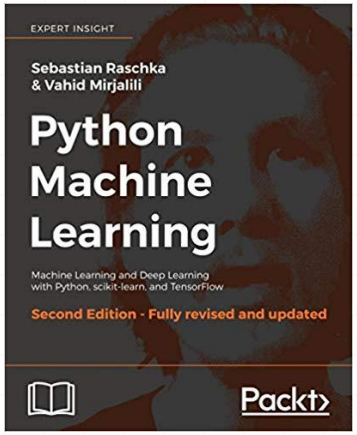
Parisa Rashidi

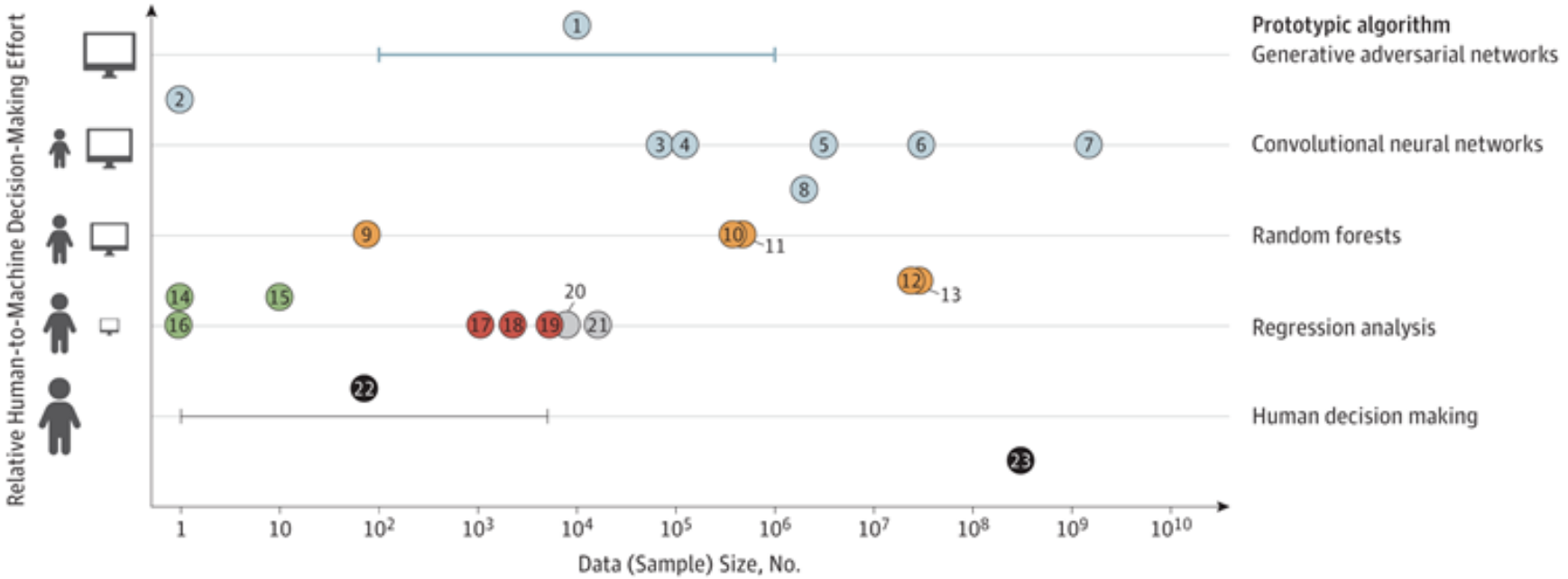
Fall 2018

# Disclaimer

The following slides are partially based on:






**Deep learning**

1. Generative adversarial networks (2014)
2. Google AlphaGo Zero (2017)
3. ATM check readers (1998)
4. Google diabetic retinopathy (2016)
5. ImageNet computer vision models (2012-2017)
6. Google AlphaGo (2015)
7. Facebook Photo Tagger (2015)
8. Prediction of 1-y all-cause mortality (2017)

**Classic machine learning**

9. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling (2002)
10. EHR-based CV risk prediction (2017)
11. Netflix Prize winner (2006)
12. Google Search (1998)
13. Amazon product recommendation (2003)

**Expert AI systems**

14. MYCIN (1975)
15. CASNET (1982)
16. DXplain (1986)

**Risk calculators**

17. $CHA_2DS_2$-VASc Score for atrial fibrillation stroke risk (2017)
18. MELD end-stage liver disease risk score (2001)
19. Framingham CV risk score (1998)

**Randomized Clinical Trials**

20. Celecoxib vs nonsteroidal anti-inflammatory drugs for osteoarthritis and rheumatoid arthritis (2002)
21. Use of estrogen plus progestin in healthy postmenopausal women (2002)

**Other**

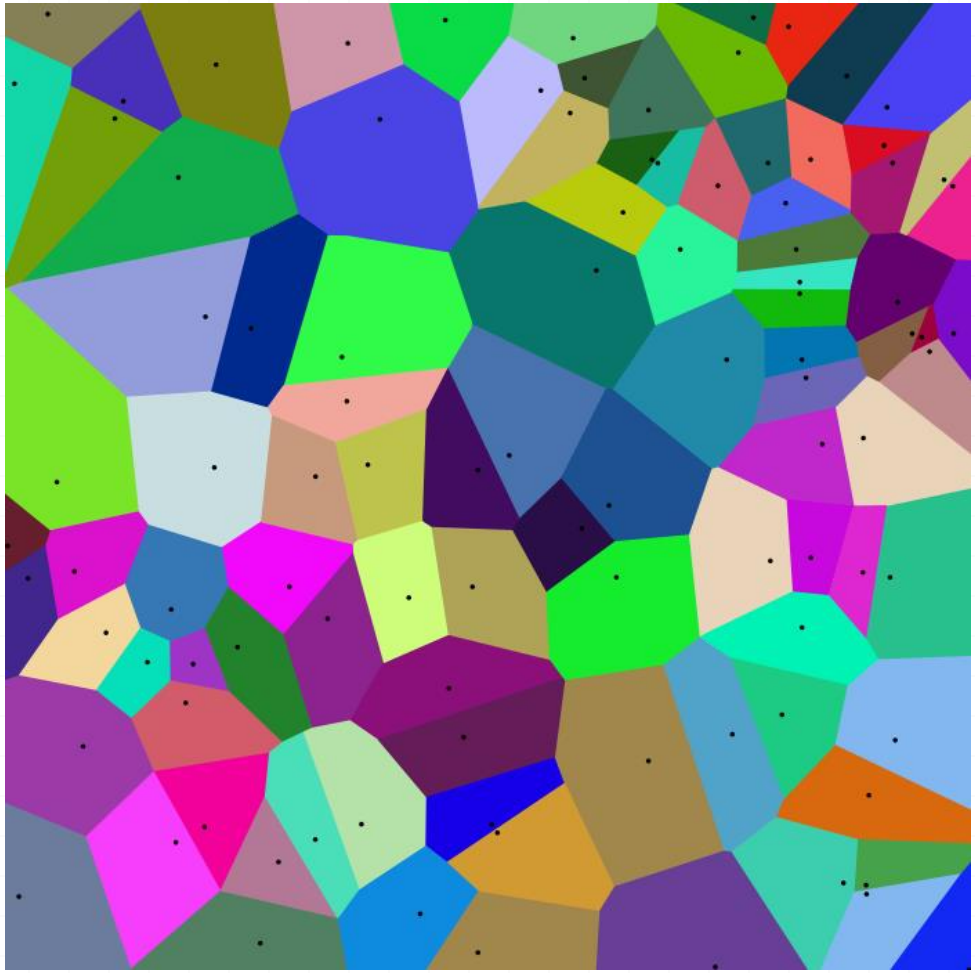
22. Clinical wisdom
23. Mortality rate estimates from US Census (2010)

Beam, A. L., & Kohane, I. S. (2018). Big data and machine learning in health care. *Jama*, *319*(13), 1317-1318.

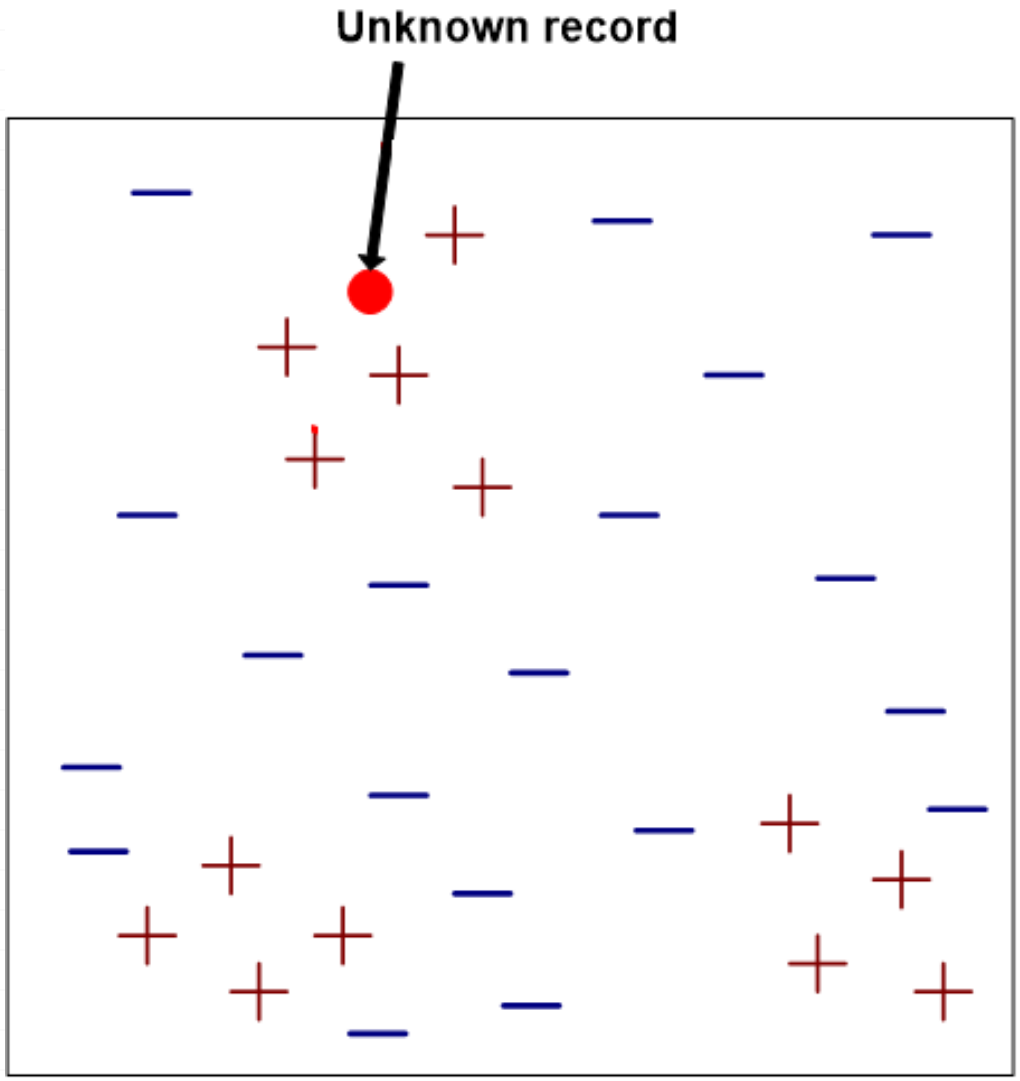
# Agenda


- Basic Classification techniques
  - K-Nearest neighbors
  - Decision Trees
  - Random Forest
  - XGBoost
- More Advanced
  - Support Vector Machines
  - Neural Network
  - Deep Learning

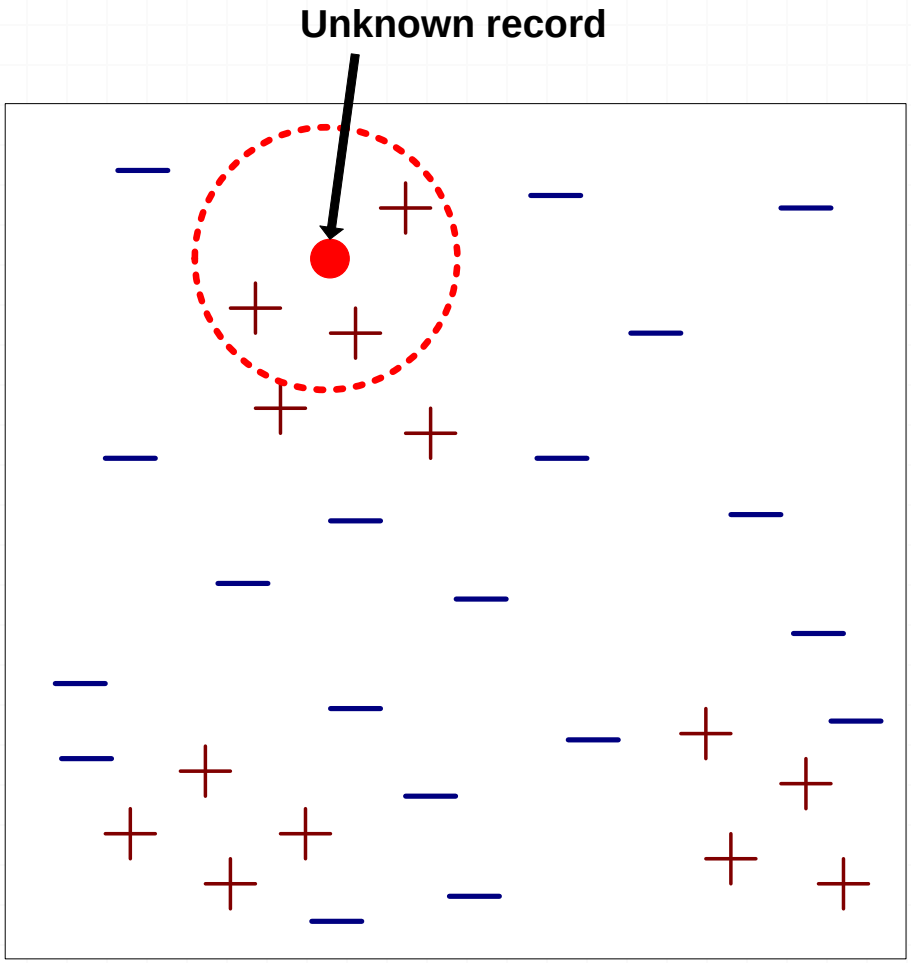
# k Nearest **Neighbors** Classifier (kNN)



# Ideas?



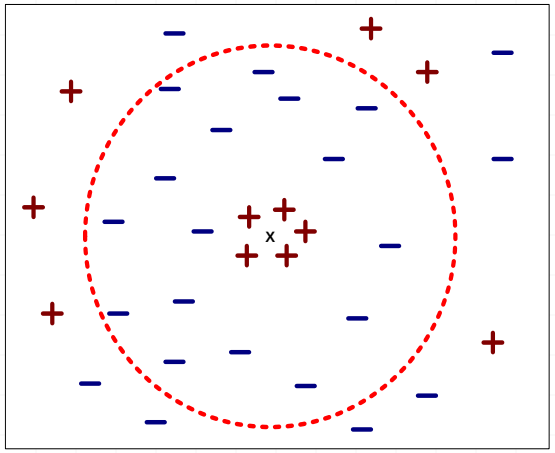
# Nearest-Neighbor Classifier




- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

# Choosing k

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification

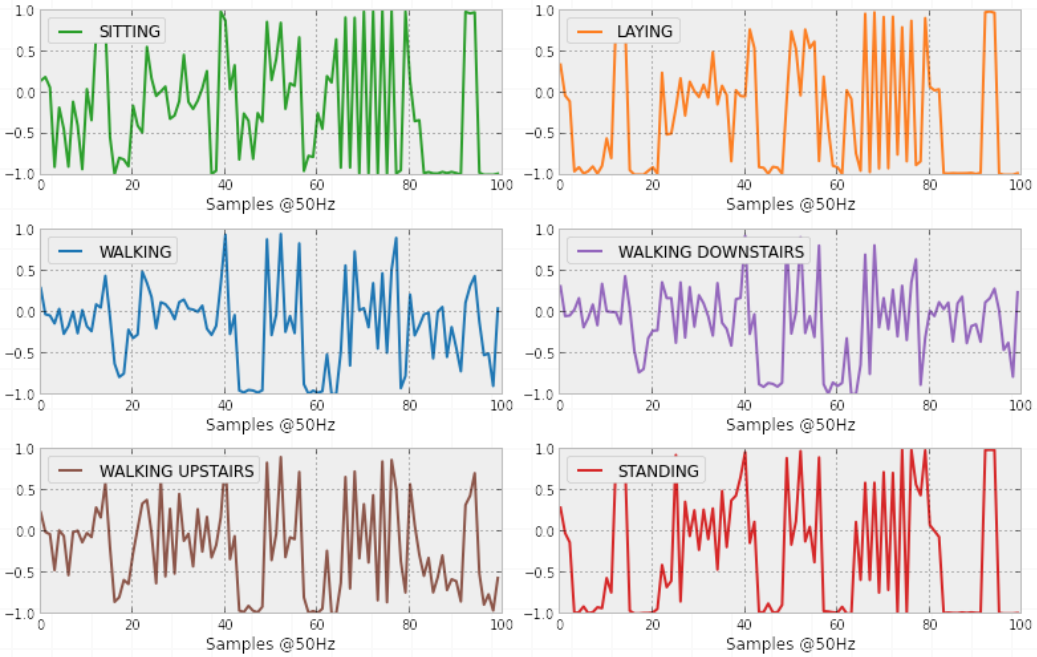
- Compute distance between two points:
  - Euclidean distance (L2 distance)

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

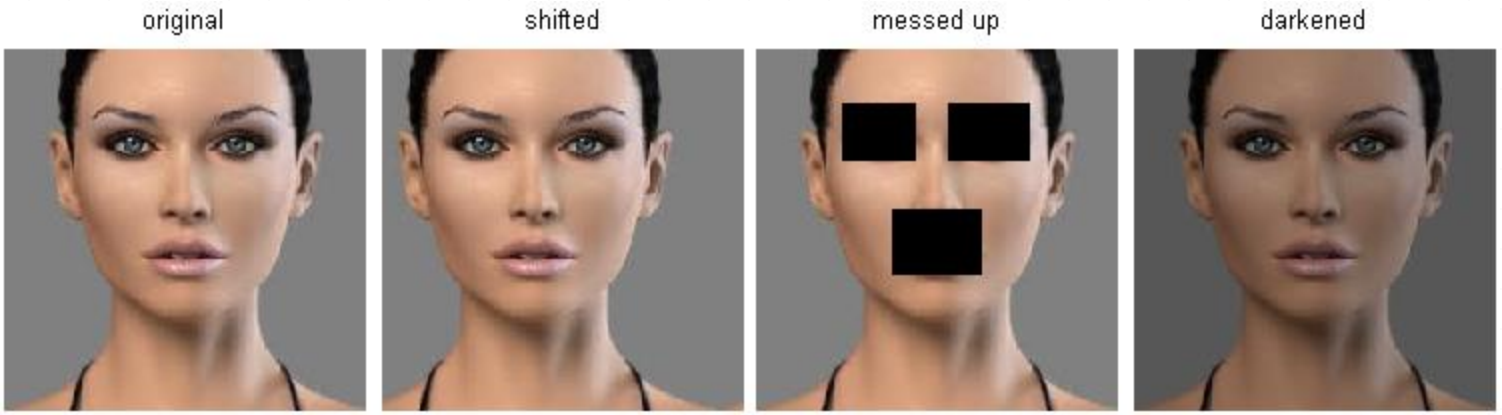
# L2 Distance with Time Series?

- Activity Recognition Example here




https://github.com/markdregan/K-Nearest-Neighbors-with-Dynamic-Time-Warping

# L2 Distance with Images?




Pixel-based distances on high-dimensional data (and images especially) can be very unintuitive. An original image (left) and three other images next to it that are all equally far away from it based on L2 pixel distance. Clearly, the pixel-wise distance does not correspond at all to perceptual or semantic similarity.

http://cs231n.github.io/classification/

# Nearest Neighbor Classification Issues

- Scaling issues
  - Attributes have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb

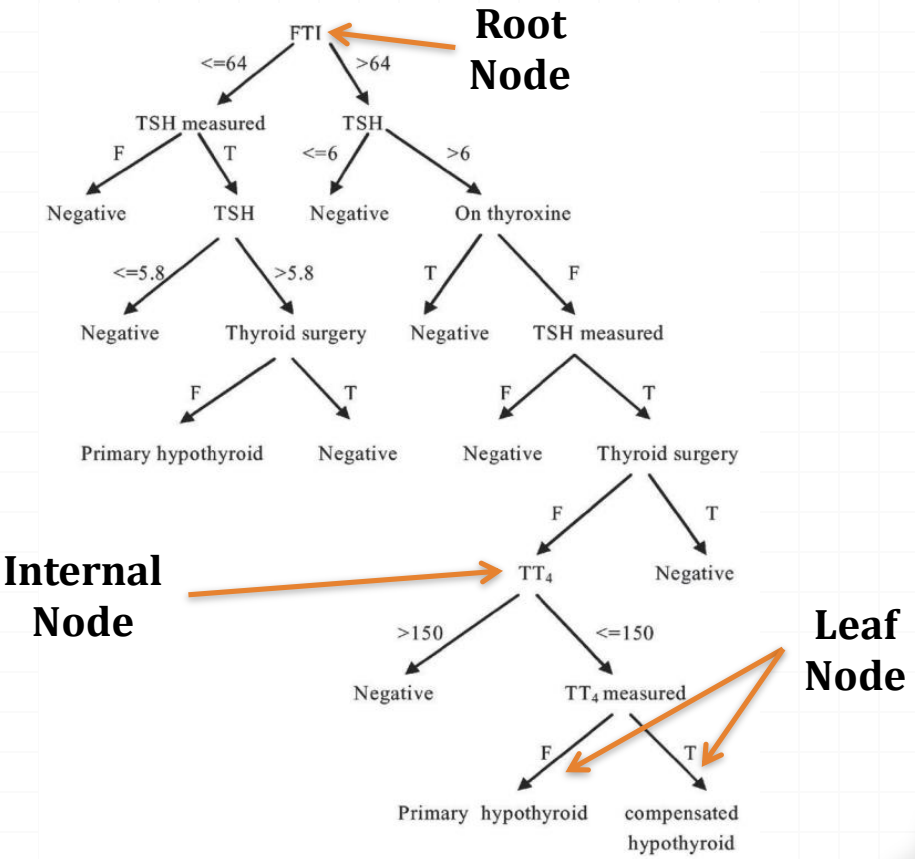
# Summary: Nearest neighbor Classification

- k-NN classifiers are lazy learners
  - It does not build models explicitly
    - Unlike eager learners such as decision tree induction
    - Classifying unknown records are relatively expensive
    - Requires the entire dataset to be stored

# Decision Tree

# Decision Tree: Motivation

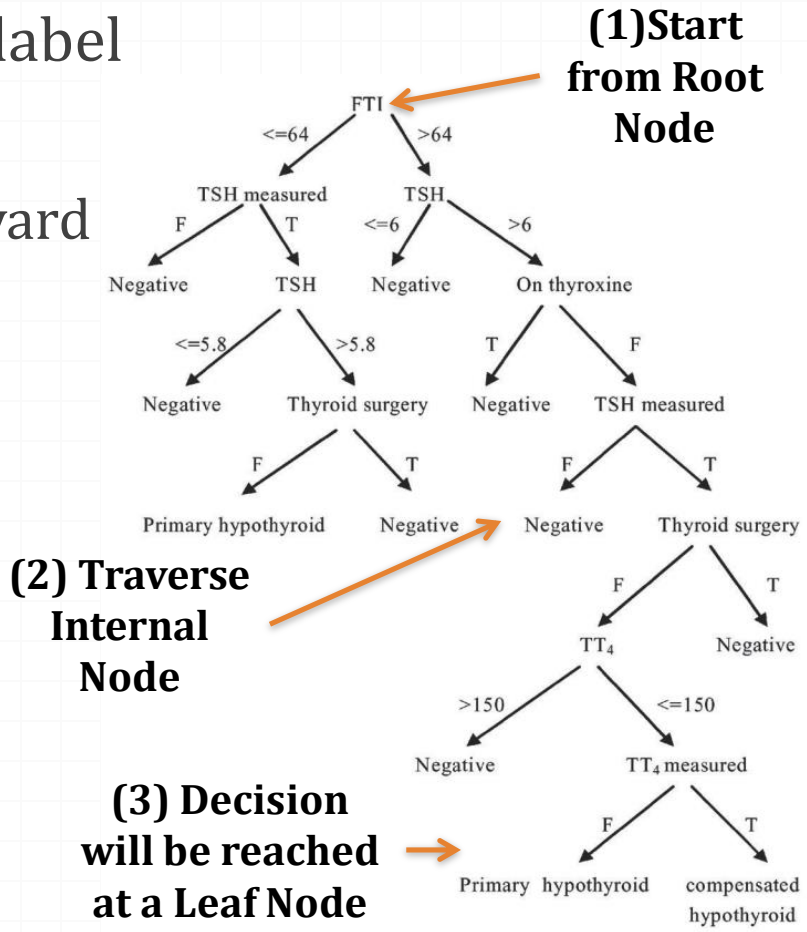
- Hypothyroid diagnosis




*Chou, Kuang-Yi, et al. "The Irrelevant Values Problem of Decision Tree in Medical Examination." Journal of Applied Science and Engineering 15.1 (2012): 89r96.

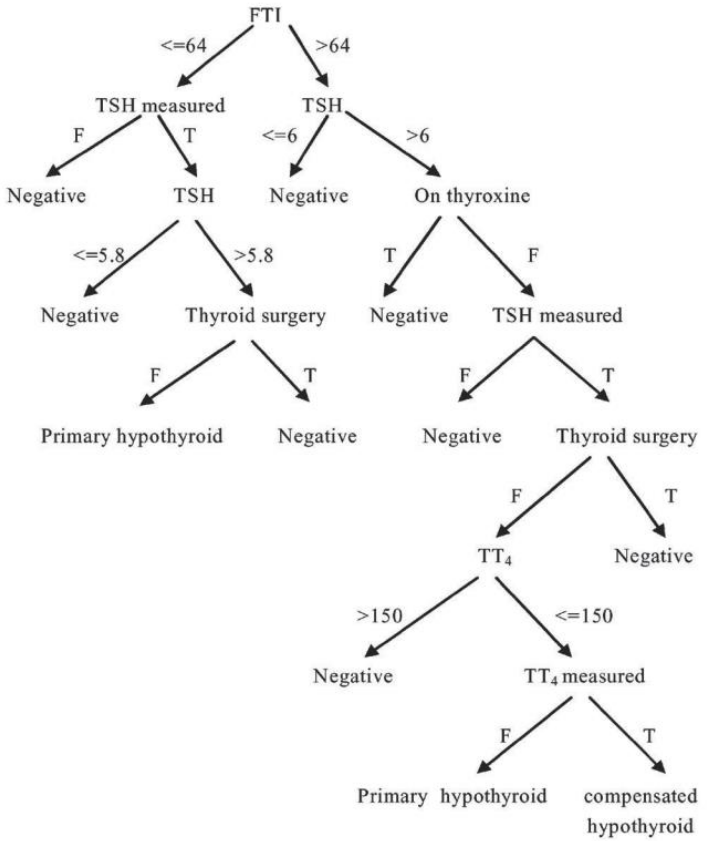
# Decision Tree: Using

- Each leaf node is assigned a label
- Once we have the tree
  - Classification is straightforward



# Decision Tree: Building

1. How to build such a tree?
2. Is the solution unique?

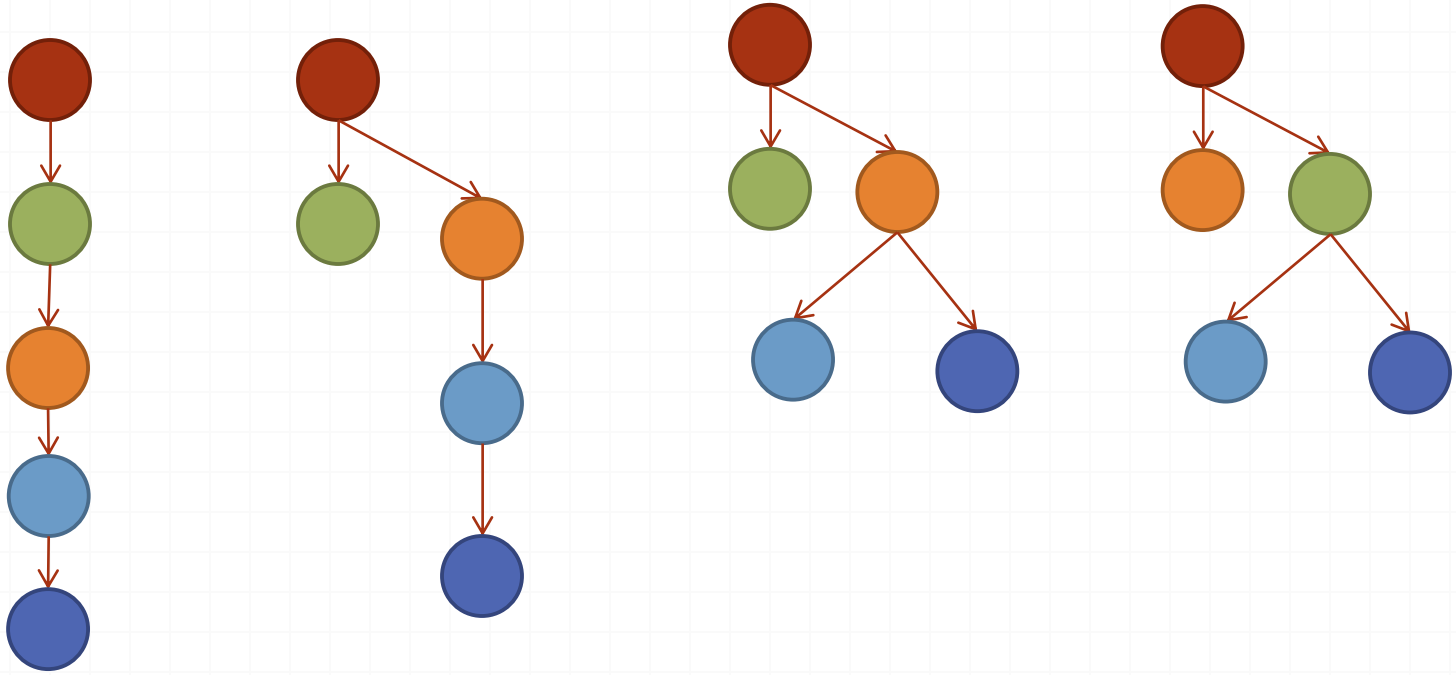


# Easy, but Effective!

- Collection of decision trees (called random forest and recently XGBoost) are winners of most Kaggle competitions
  - On structured data
- On unstructured data (e.g. images,)
  - Deep learning

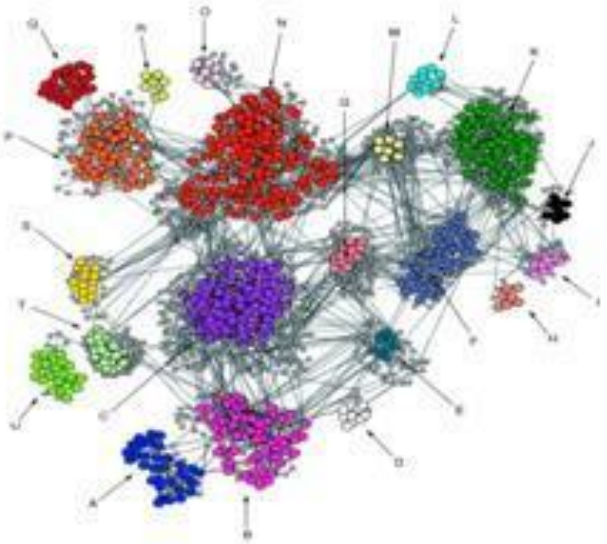
# Number of Trees

- Exponential



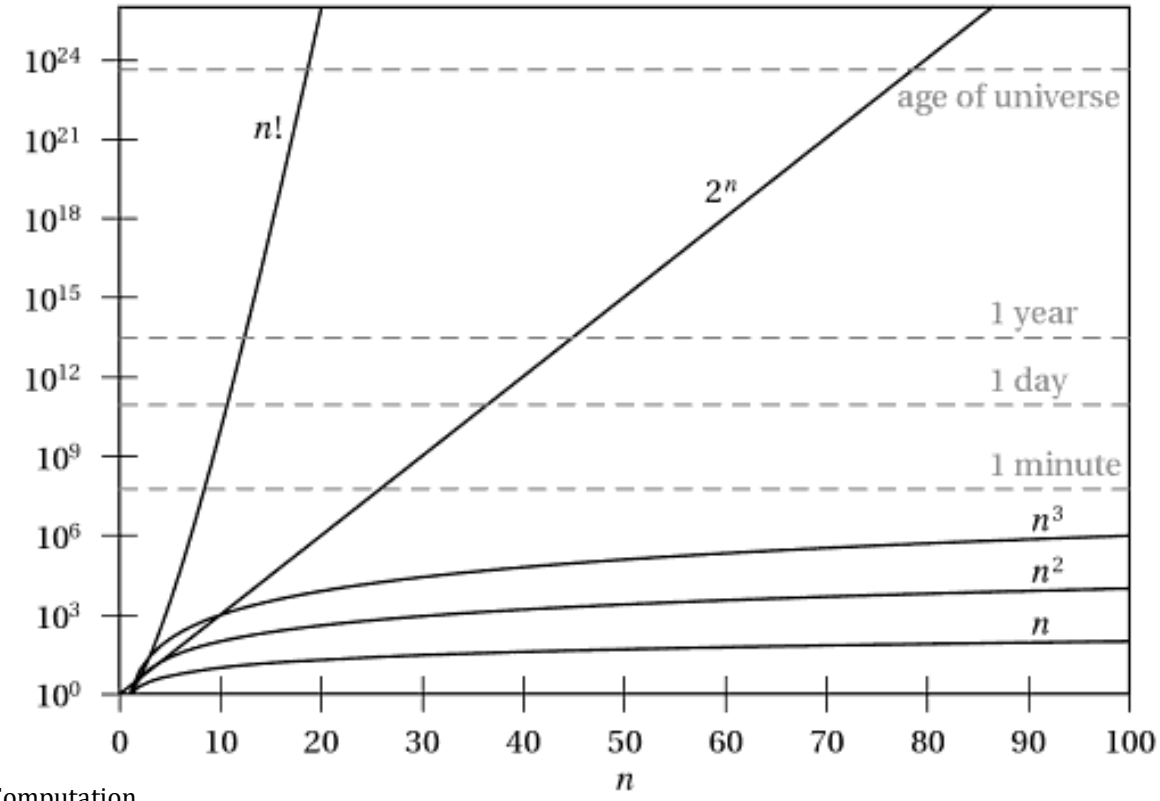
# A Little Detour: Computational Complexity

- Computational complexity classes
  - Is expressed in terms of input size *n*
    - *Polynomial: $O(n)$, $O(n^2)$, $O(n^3)$, ..*
    - *Exponential: $O(2^n)$, $O(3^n)$, ...*



# A Little Detour: Computational Complexity

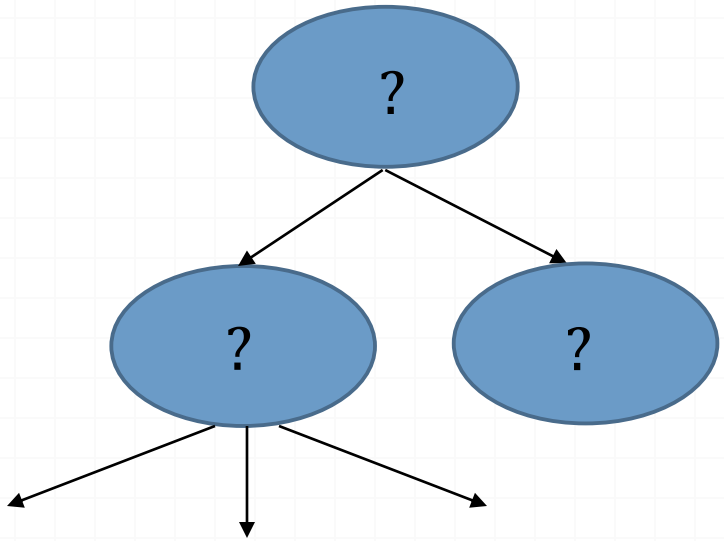
- Suppose we can solve a problem of size n = 1 in 1 μ second.




* Moore, Cristopher; Mertens, Stephan: The Nature of Computation.

# Decision Tree: Build

- How to build such tree?
  - Exponential number of trees!
  - Greedy approach:
    - Suboptimal
    - Many algorithms
      - ID3, C4.5, CART, …



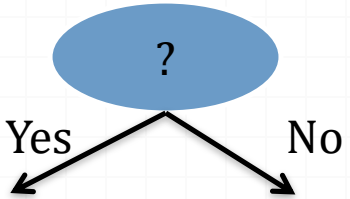
# Decision Tree: Example Data

- Training set for predicting if someone will be cheating on their tax return:

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

# Decision Tree Algorithm

- Choose a feature (which one? Later on this …)
- If all the records associated with a node *t* have the same label
  - *t* is a leaf node
- Otherwise
  - Partition records into smaller subsets



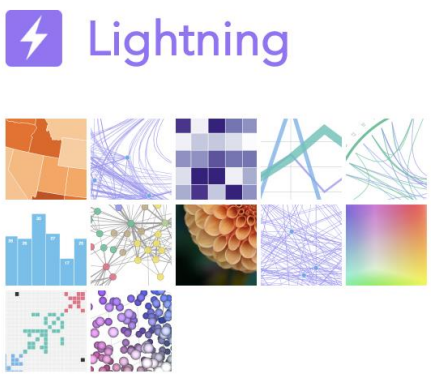
# Decision Tree Visualized

- Awesome decision tree visualization

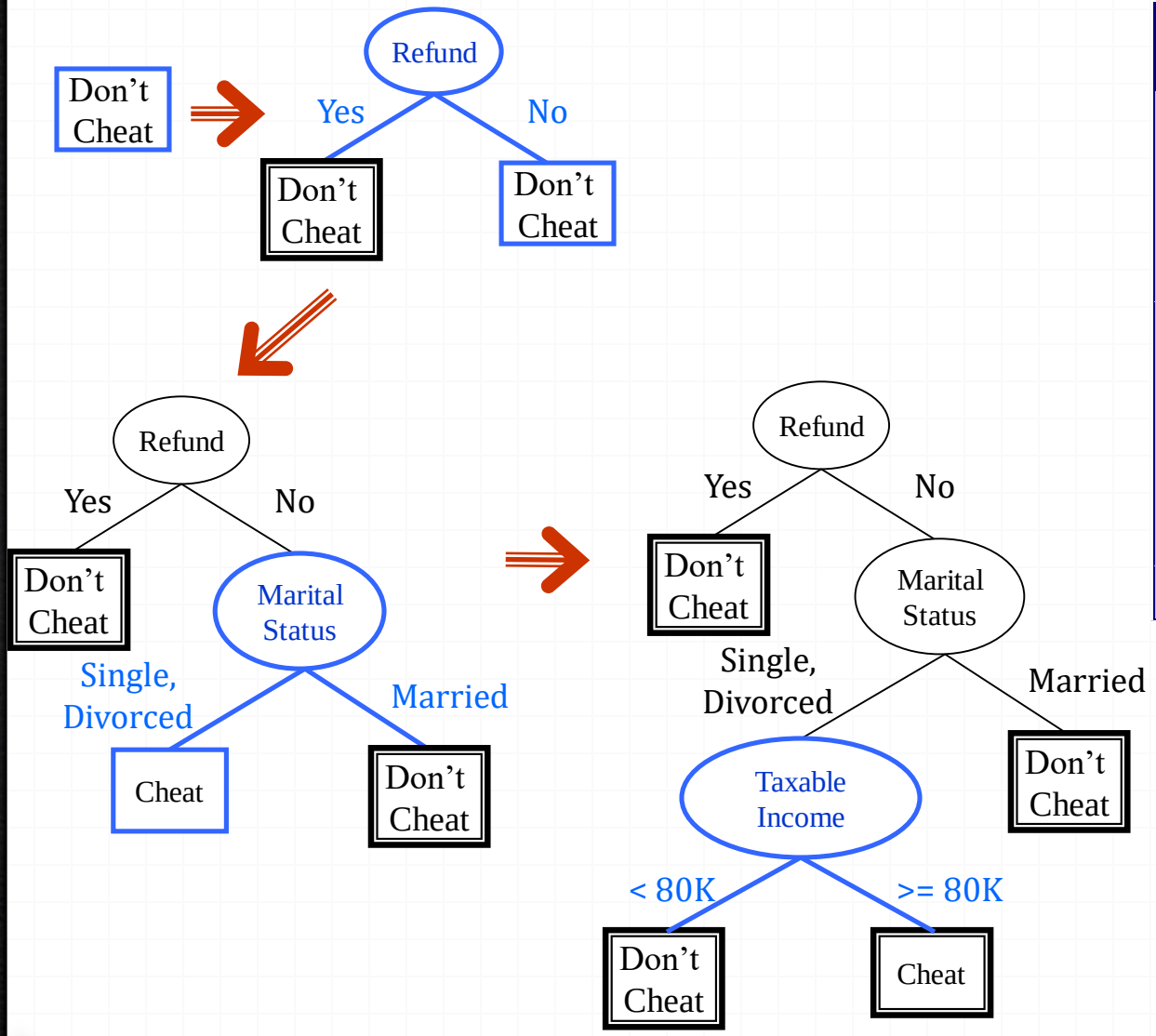
  http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

- A good visualization library

http://lightning-viz.org/



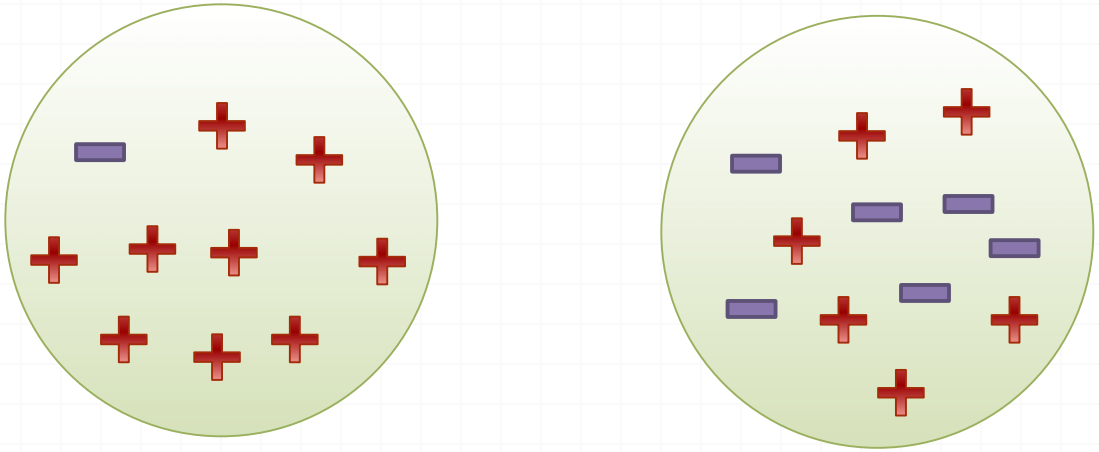
# Algorithm




| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

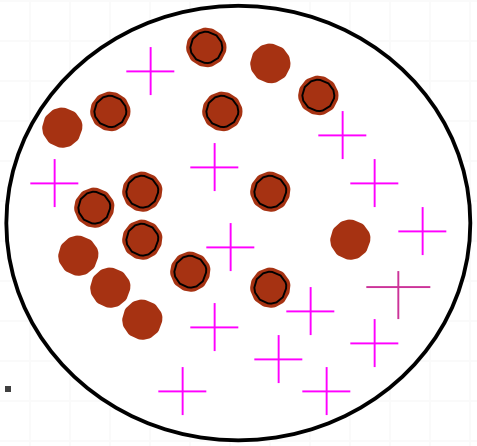
# Which Feature?

- Which feature should be chosen at each step?
  - Based on what measure?
    - Most measures are based on node impurity



# Entropy: a common way to measure impurity

$$\text{Entropy} = \sum_i - p_i \log_2 p_i$$



$p_i$ is the relative frequency of class i

Compute it as the proportion of class i in the set.

16/30 circles; 14/30 crosses,

$\log_2(16/30) = -0.9$;
$\log_2(14/30) = -1.1$

Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

*Entropy comes from information theory*.

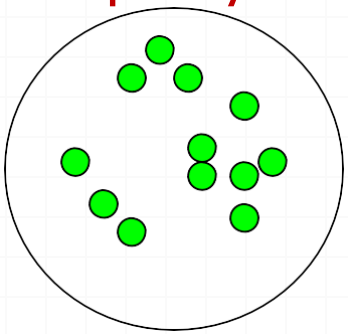
http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

# More Examples

What is the entropy of a group in which all examples belong to the same class?

- entropy = $- 1 \log_2 1 = 0$

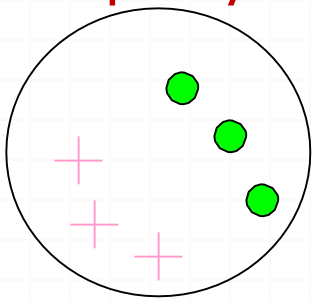
Minimum impurity



What is the entropy of a group with 50% in either class?

- entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Maximum impurity



# From Entropy to Information Gain

- We want to determine which feature in a given set of training features vector is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given feature of the features vector is.

  We will use it to decide the ordering of attributes in the nodes of a decision tree.

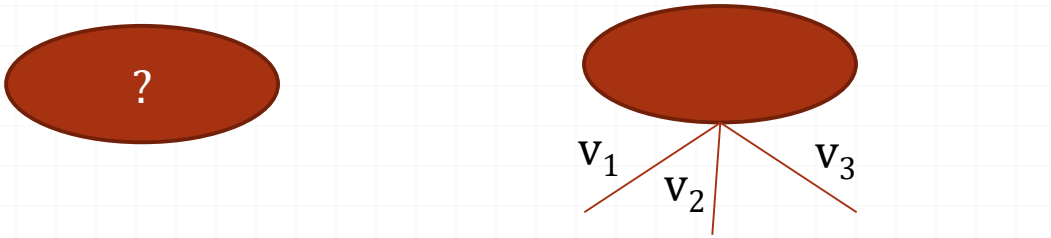
http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

# Information Gain

- We will compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the children (after splitting).
  - The larger their difference, the better the test condition.

$$\Delta = I(before\ splitting) - I(after\ spiltting)$$

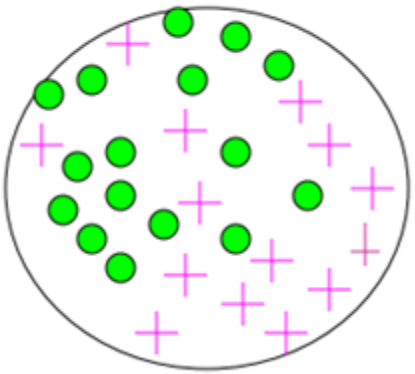
$$\Delta = I(parent) - weighted\_average\ (I(children))$$



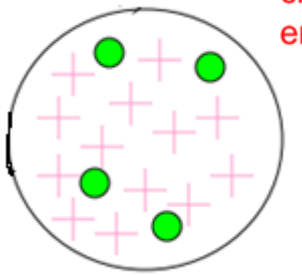

$$\Delta = I(parent) - \frac{N_1}{N} I(v_1) - \frac{N_2}{N} I(v_2) - \frac{N_3}{N} I(v_3)$$

# Computing Information Gain

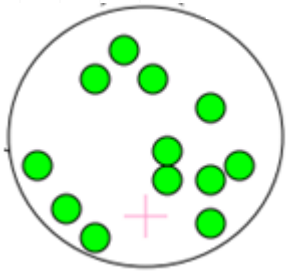
$$\Delta = I(parent) - weighted\_average\ (I(children))$$




parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.99$




child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$




child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

**Information Gain= 0.996 - 0.615 = 0.38 for this split**

# Which Feature to choose?

- Choose the feature with highest Δ
- (Note: in case of the entropy measure, Δ is known as "information gain")

# Example Dataset

$\mathbf{D} =$

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |

$\mathbf{X} = \{X_1, X_2, X_3, X_4\}$

CS181 Lecture 7, Harvard, link

# Example Dataset

**D** =

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |

$$\text{Entropy}(\mathbf{D}) = \text{Entropy}(4:3)$$
$$= -\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7}$$
$$= 0.98$$

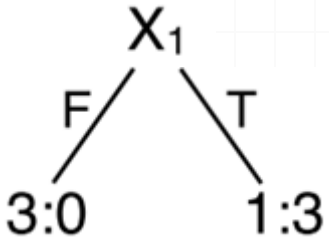
$$\mathbf{X} = \{X_1, X_2, X_3, X_4\}$$

$$Entropy(t) = -\sum_j p(j \mid t)\log_2 p(j \mid t)$$

# Example Dataset

D =

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |




$$\text{Entropy}(3:0) = -\tfrac{3}{3}\log_2\tfrac{3}{3} - \tfrac{0}{3}\log_2\tfrac{0}{3} = 0.0$$

$$\text{Entropy}(1:3) = -\tfrac{1}{4}\log_2\tfrac{1}{4} - \tfrac{3}{4}\log_2\tfrac{3}{4} = 0.81$$

$$Entropy(t) = -\sum_j p(j\,|\,t)\log_2 p(j\,|\,t)$$

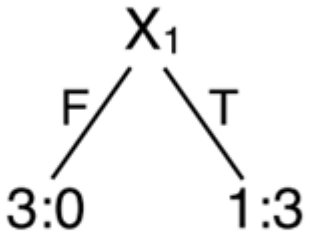
CS181 Lecture 7, Harvard, link

# Example Dataset

**D =**

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |

$$\text{Entropy}(\mathbf{D}) = 0.98$$




$$\text{Entropy}(3:0) = 0.0$$

$$\text{Entropy}(1:3) = 0.81$$

$$\textbf{weighted\_avg}\ (X_1, D) = \frac{3}{7} * 0 + \frac{4}{7} * 0.81 = 0.46$$

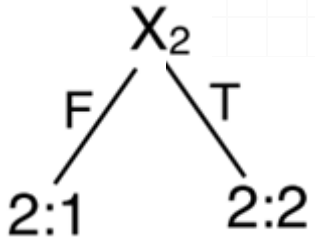
$$\text{Gain}(X_1) = 0.98 - 0.46 = 0.52$$

$$\Delta = I(parent) - weighted\_average\ (I(children))$$

# Example Dataset

D =

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |




$$\text{Entropy}(2:1) = 0.92$$

$$\text{Entropy}(2:2) = 1.0$$

$$\textbf{weighted\_avg}\ (X_2, D) = \frac{3}{7} * 0.92 + \frac{4}{7} * 1.0 = 0.97$$

$$\text{Gain}(X_2) = 0.98 - 0.97 = 0.01$$

$$\Delta = I(parent) - weighted\_average\ (I(children))$$

# Example Dataset

**D =**

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |

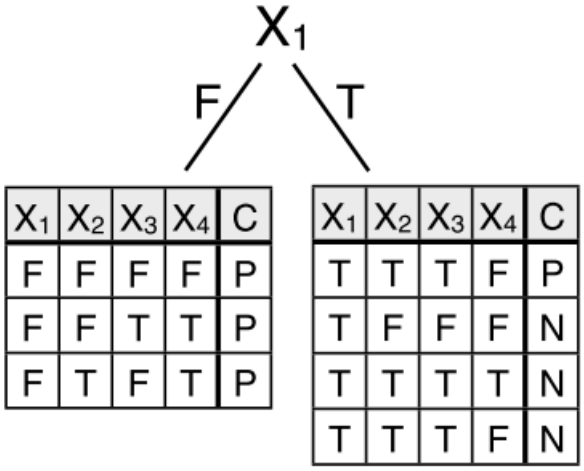
$\mathbf{X} = \{X_1, X_2, X_3, X_4\}$

$\text{Gain}(X_1) = 0.52$ ✓

$\text{Gain}(X_2) = 0.01$

$\text{Gain}(X_3) = 0.01$

$\text{Gain}(X_4) = 0.01$




| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| F | F | F | F | P |
| F | F | T | T | P |
| F | T | F | T | P |

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | C |
|---|---|---|---|---|
| T | T | T | F | P |
| T | F | F | F | N |
| T | T | T | T | N |
| T | T | T | F | N |

# Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar feature values

# Using Decision Trees: Advantages

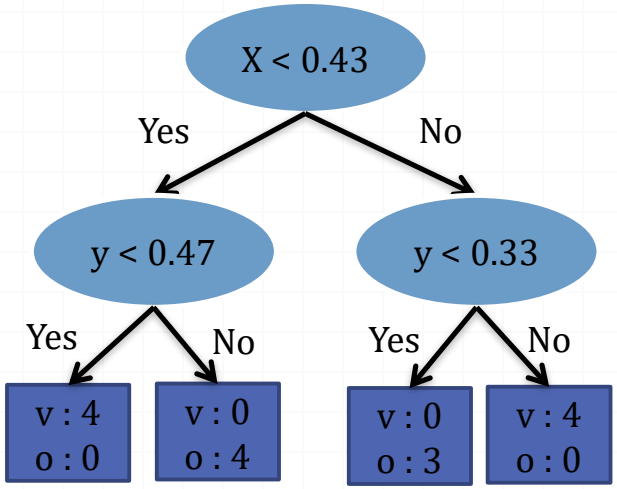
- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

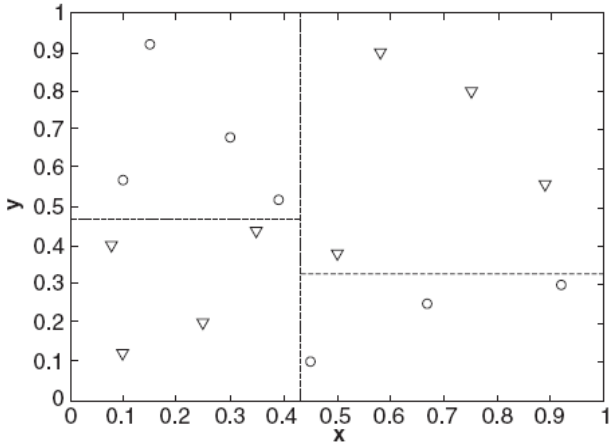
# Expressiveness of Decision Trees

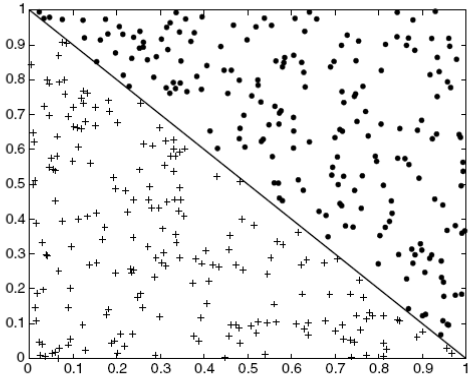
- Decision tree provides expressive representation for learning discrete-valued function
- Not expressive enough for modeling continuous variables

# Using Decision Trees: Problems

- Rectilinear decision boundary



# Random Forest (RF)

- An ensemble of decision trees
- The idea is to average multiple trees that individually suffer from high variance, to get a more robust model
- This will reduce overfitting

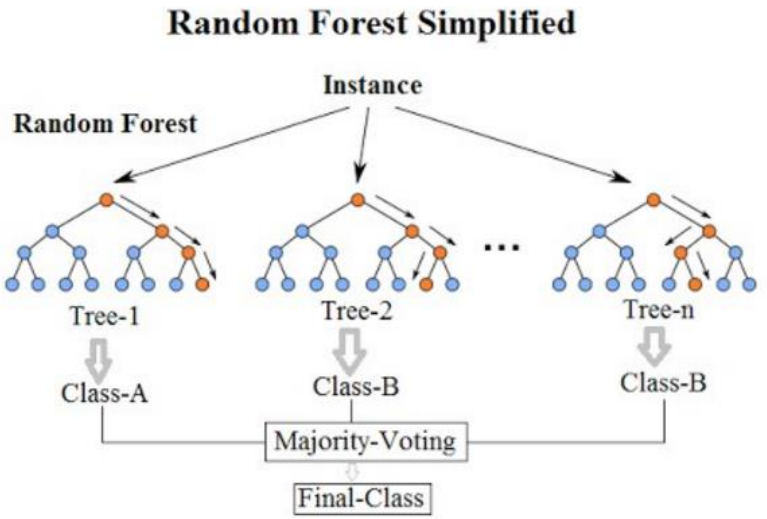



Image from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

# RF Algorithm

1. Draw a random **bootstrap** sample of size $n$ (randomly choose $n$ samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:

   a. Randomly select $d$ features without replacement.

   b. Split the node using the feature that provides the best split

3. Repeat the steps 1-2 $k$ times.
4. Aggregate the prediction by each tree to assign the class label by **majority vote**.

# RF Hyper-parameters

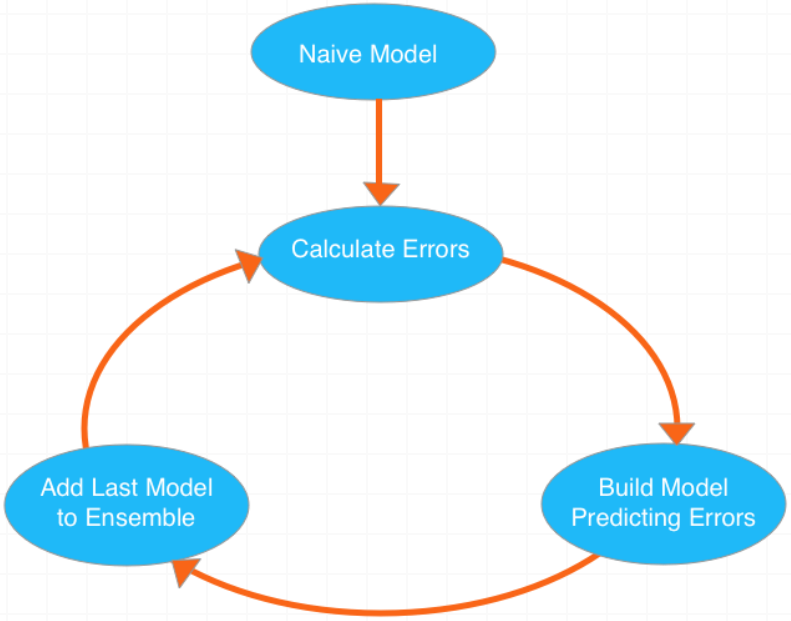
- Number of decision tress, $k$
- Size of bootstrap sample, $n$
  - Small size: more diversity among trees to avoid overfitting, however lower overall performance
  - Larger: more overfitting (more similar trees fit the training set more closely)
  - Default = same as dataset size
- Number of features, $d$
  - Default = $\sqrt{m}$ where $m$ is total number of features

# RF Summary

- More robust to overfitting
- Typically, the larger the number of trees, the better the performance
  - at the expense of increased computational cost

# XGBoost (eXtreme Gradient Boosting)

- Similar to Random Forest, but based on gradient boosting
- XGBoost builds short and simple decision trees iteratively.




https://www.kaggle.com/dansbecker/xgboost

# XGBoost

- Each tree is called a weak learner for their high bias. XGBoost starts by creating a first simple tree which has poor performance by itself.
- It then builds another tree which is trained to predict what the first tree was not able to.
- The algorithm goes on by sequentially building more weak learners, each one correcting the previous tree until a stopping condition is reached, such as the number of trees.
  - This is called Boosting

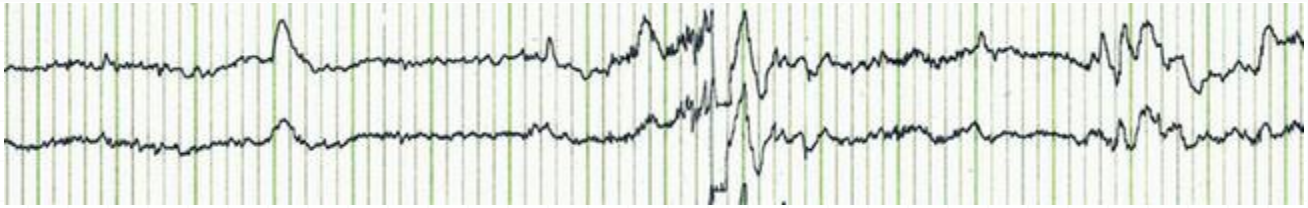
https://www.kaggle.com/dansbecker/xgboost

# Applications

- XGBoost is one of the top choices for structured data

```
from xgboost import XGBRegressor

my_model = XGBRegressor()
# Add silent=True to avoid printing out updates with each cycle
my_model.fit(train_X, train_y, verbose=False)
```



1st place of the Grasp-and-Lift EEG Detection. Link to the Kaggle interview.