ME 5374-ST

Machine Learning for Materials Science and Discovery

Fall 2025

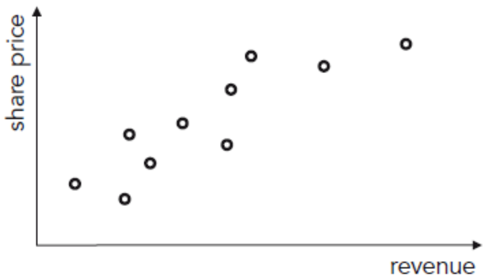
Asst. Prof. Peter Schindler

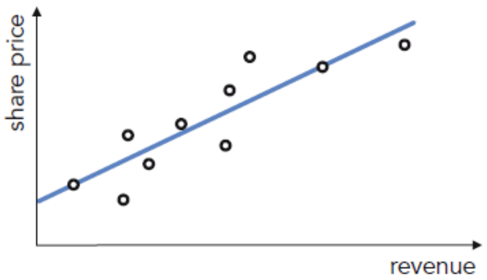
Lecture 6 – Machine Learning Basics 2

- Feature Filtering and Regularization Methods
- Distance in High-dimensional Space
- Logistic Regression, Classification, and its Performance Metrics
- Clustering: K-means
- Dimensionality Reduction, Principal Component Analysis
- Decision Tree, Random Forest, Ensemble and Bagging Methods
- k-fold Cross-validation

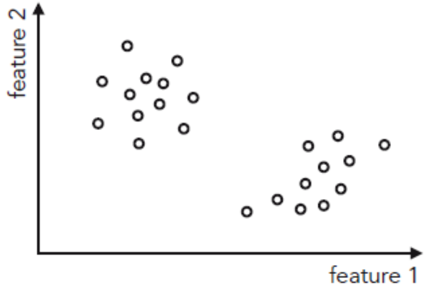
Supervised vs. Unsupervised Learning

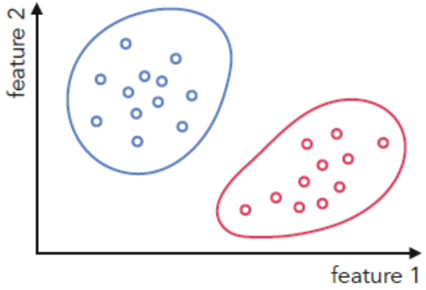
Supervised: Regression



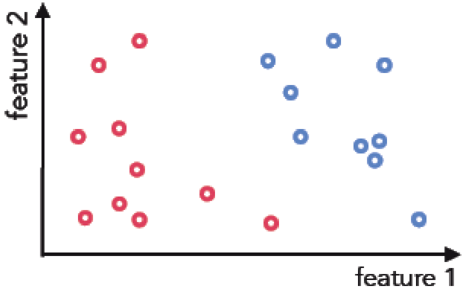


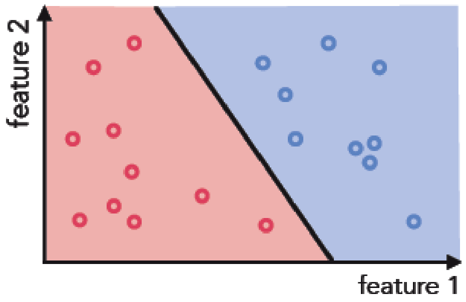
Unsupervised: Clustering



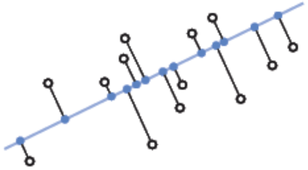


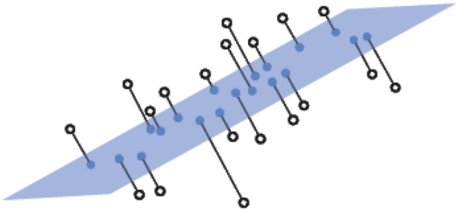
Supervised: Classification





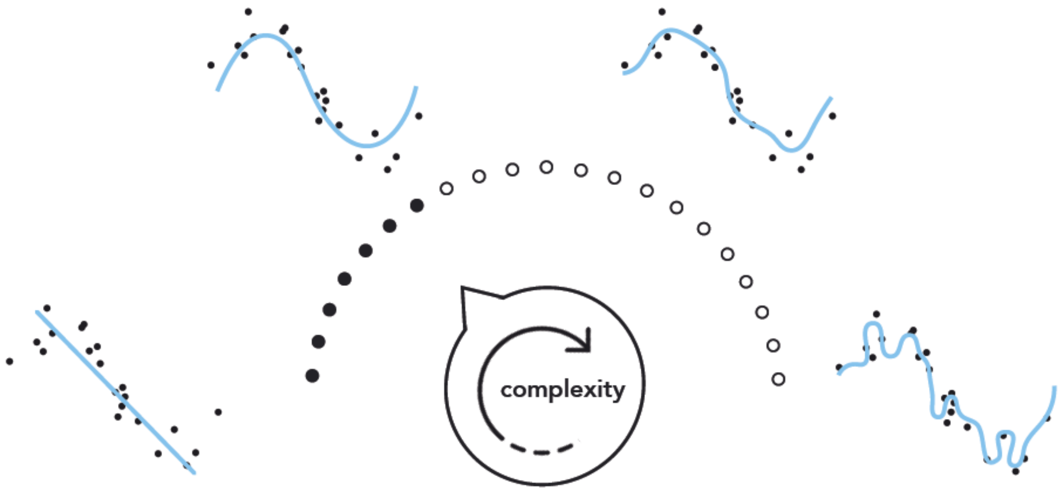
Unsupervised: Dim. Reduction

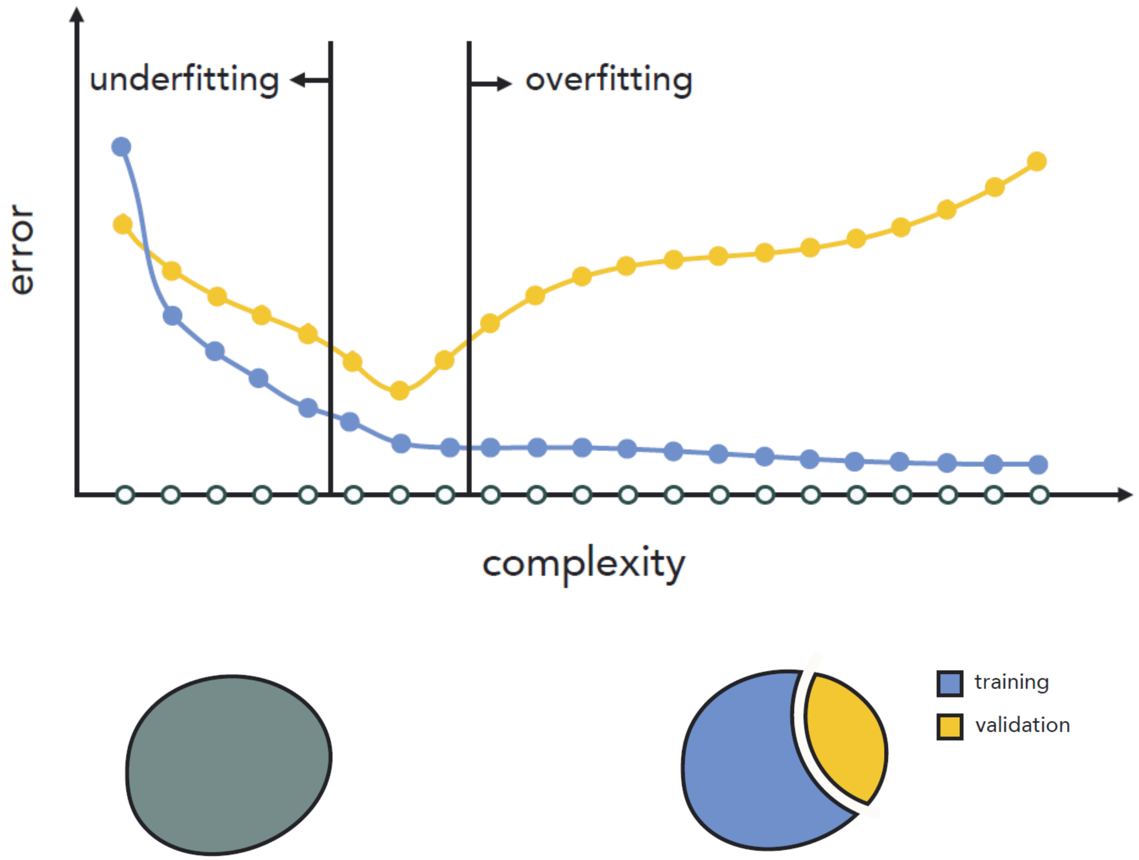




"*Machine Learning Refined*", 2nd Edition, J. Watt, R. Borhani, A. K. Katsaggelos

Overfitting and Underfitting





"*Machine Learning Refined*", 2nd Edition, J. Watt, R. Borhani, A. K. Katsaggelos

Approaches to Avoid Overfitting

0. Remove low variance features and highly correlated features
1. Filtering methods (pre-ML model)
   Correlation matrix (*e.g.* Pearson)
2. Iterative methods (using ML model performance)
3. Regularization methods (directly enforced during ML training)

Approach 1: Filtering Methods (Pre-ML)

Pearson Coefficient:

$$r_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

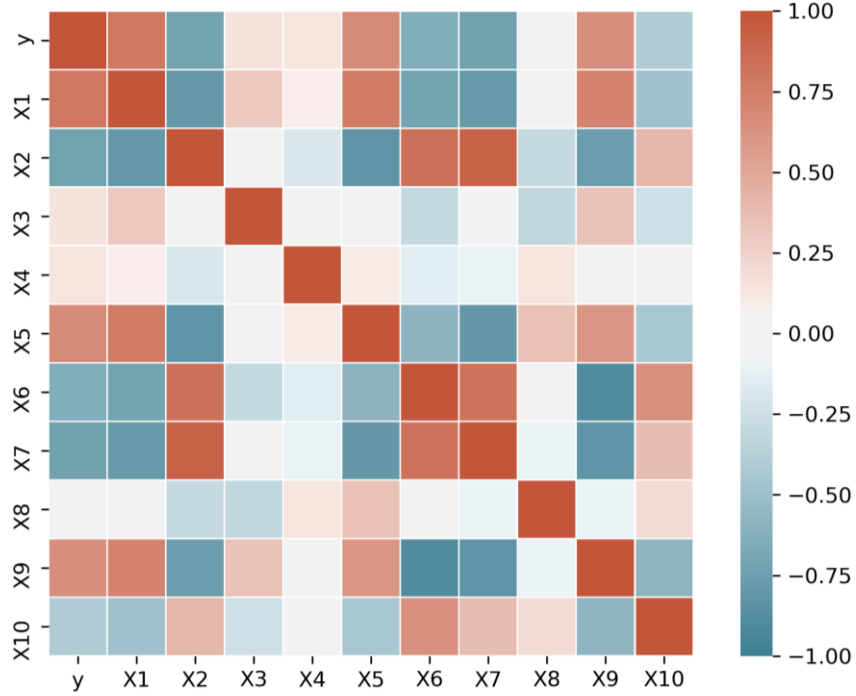
$$r_{X,Y} \in [-1, 1]$$

$r_{X,Y}$ positive: X↑ → Y↑, X↓ → Y↓

$r_{X,Y}$ negative: *X*↑ → Y↓, X↓ → Y↑

$r_{X,Y}$ zero: *X* and *Y* not correlated

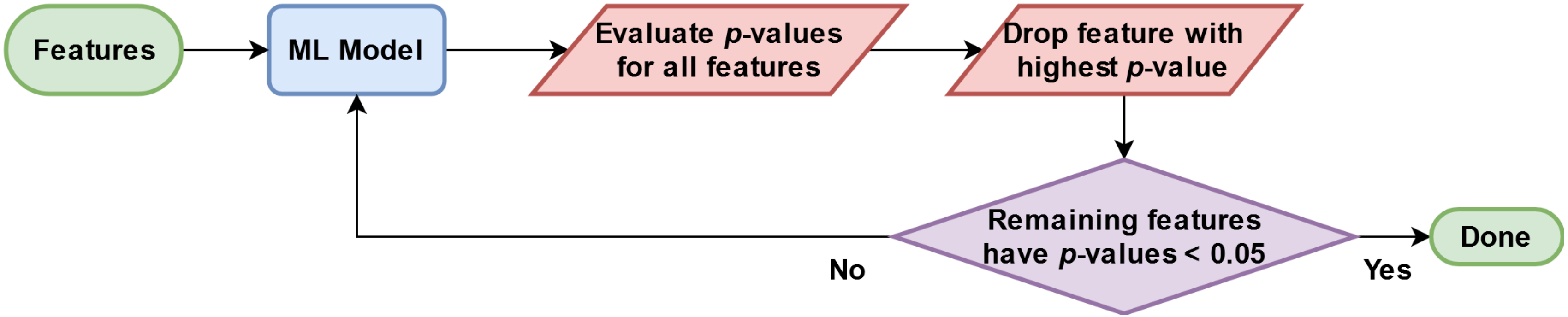
Correlation Matrix



Approach 2: Iterative Methods (Utilizes ML Model)

Uses ML model to assess performance and then **iteratively** remove/add features

- Backward Elimination (*p*-value)
- Recursive Feature Elimination (accuracy/feature importance)
- Forward Selection, Bidirectional Elimination



Distances in High-Dimensional Space

$$\boxed{\ell_p \text{ norm: } \|\theta\|_p = \sqrt[p]{\sum_i |\theta_i|^p}}$$

ℓ_0 norm

$$\|\theta\|_0 = \#\left(i \mid \theta_i \neq 0\right)$$

Number of non-zero entries "norm"

ℓ_1 norm

$$\|\theta\|_1 = \sum_i |\theta_i|$$

Manhattan norm

ℓ_2 norm

$$\|\theta\|_2 = \sqrt[2]{\sum_i |\theta_i|^2}$$

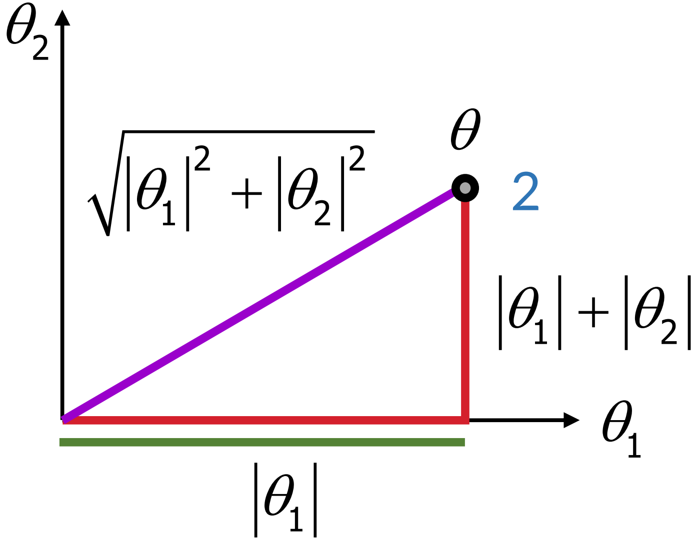
Euclidean norm

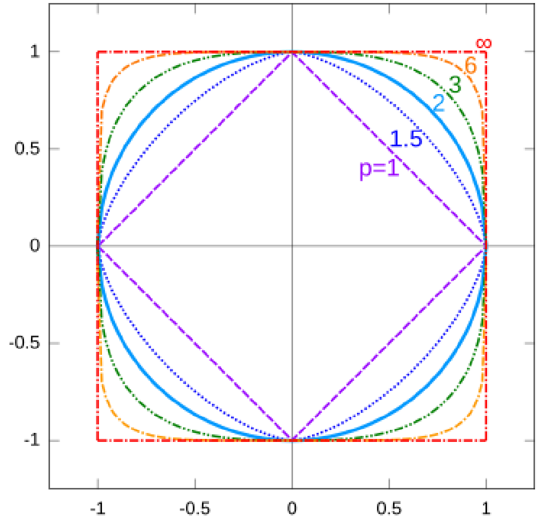
ℓ_∞ norm

$$\|\theta\|_\infty = \max |\theta_i|$$

Chebyshev norm

ℓ_0: Use for sparsity and feature selection.
ℓ_1: Use when data has different scales or follows a grid-like structure.
ℓ_2: Use when data is dense and continuous, and features have similar scales.
ℓ_∞: Use to emphasize the largest single difference; highlights outliers.





wikipedia.org

Approach 3: Regularization

Linear regression (least squares) $\{x^1, y^1\}, \ldots, \{x^m, y^m\}$; $x^i \in \mathbb{R}^{n+1}, y \in \mathbb{R}$

$X \in \mathbb{R}^{m\times(n+1)}$

$$y = X\theta \qquad \underset{\theta\in\mathbb{R}^{n+1}}{\arg\min} \|y - X\theta\|_2$$

Exact solution: $\theta = (X^T X)^{-1} X^T y$

Finding **sparse solution**: Regularization/compressed sensing

$$\underset{\theta\in\mathbb{R}^{n+1}}{\arg\min}\left(\|y - X\theta\|_2 + \lambda\|\theta\|_0\right)$$

Exact solution but NP hard (not convex)

$$\underset{\theta\in\mathbb{R}^{n+1}}{\arg\min}\left(\|y - X\theta\|_2 + \lambda\|\theta\|_1\right)$$

LASSO regression

$$\underset{\theta\in\mathbb{R}^{n+1}}{\arg\min}\left(\|y - X\theta\|_2 + \lambda\|\theta\|_2\right)$$

Ridge regression

$$\underbrace{\text{LASSO regression} \qquad \text{Ridge regression}}_{\text{Elastic Net}}$$

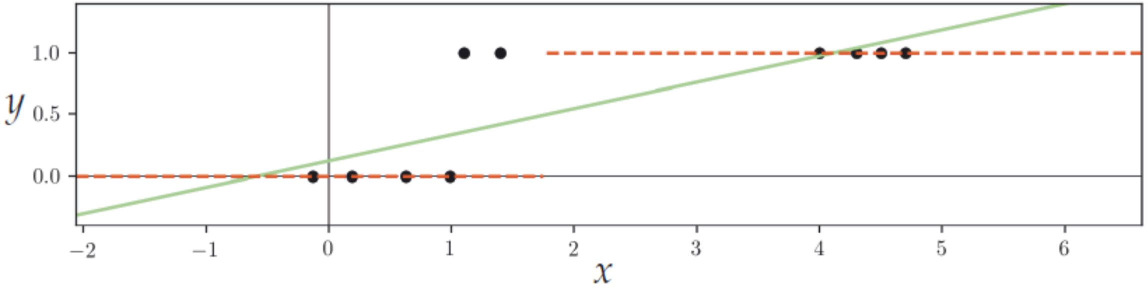
$$\theta = \left(X^T X + \lambda \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right)^{-1} X^T y$$

NP = nondeterministic polynomial time,

LASSO = Least Absolute Shrinkage and Selection Operator

Logistic Regression and Classification

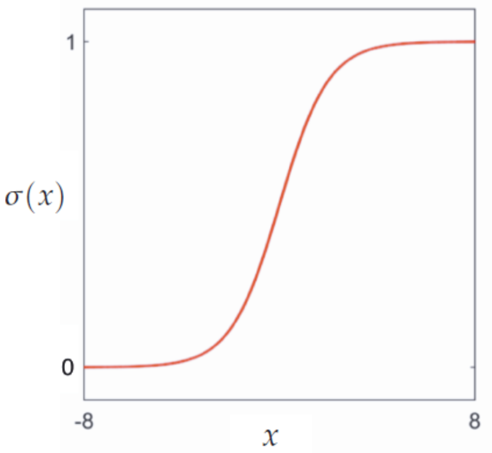
Data in classification task: (x^i, y^i), where $x^i \in \mathbb{R}^{n+1}$, $y^i \in \{0, 1\}$ for $i \in [1, m]$

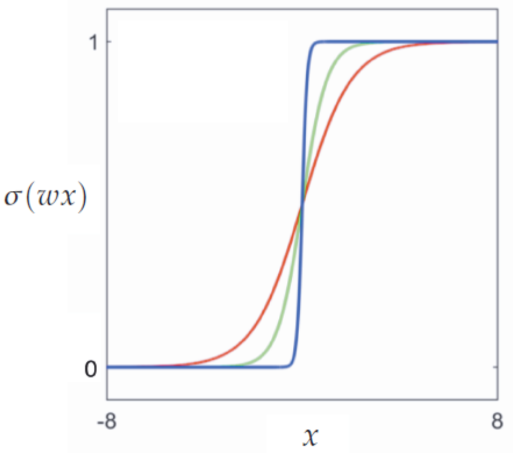


$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\delta(\theta^T x^i) - y^i)^2$$

Gradient Descent doesn't work on step function!

Instead: Sigmoid Function





$$\boxed{h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}}$$

$$\text{predict } y = 1 \quad \text{if} \quad h_\theta(x) \geq 0.5 \quad \rightarrow \quad \theta^T x \geq 0$$

$$\text{predict } y = 0 \quad \text{if} \quad h_\theta(x) < 0.5 \quad \rightarrow \quad \theta^T x < 0$$

"*Machine Learning Refined*", 2nd Edition, J. Watt, R. Borhani, A. K. Katsaggelos

Logistic Cost Function (Cross Entropy)

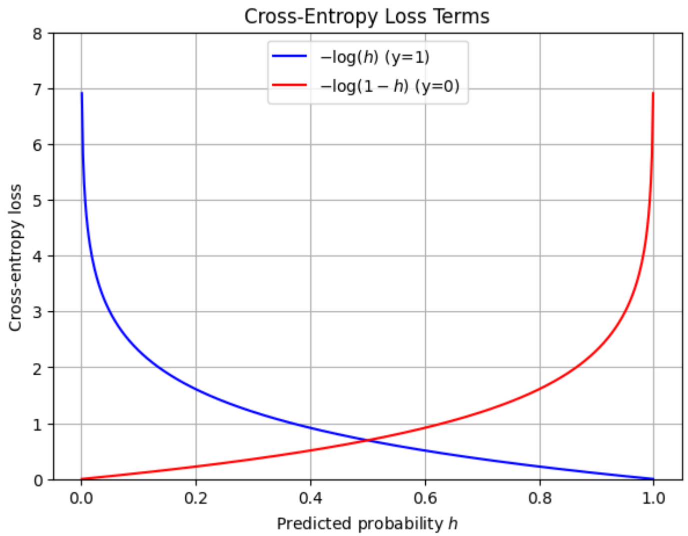
Regular MSE cost function would work but is generally non-convex

Convex alternative that works only for y ∈ {0, 1}:

$$\boxed{J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} \underbrace{y^i \log h_\theta(x^i)}_{\text{for } y=1} + \underbrace{(1-y^i)\log(1-h_\theta(x^i))}_{\text{for } y=0}\right]}$$

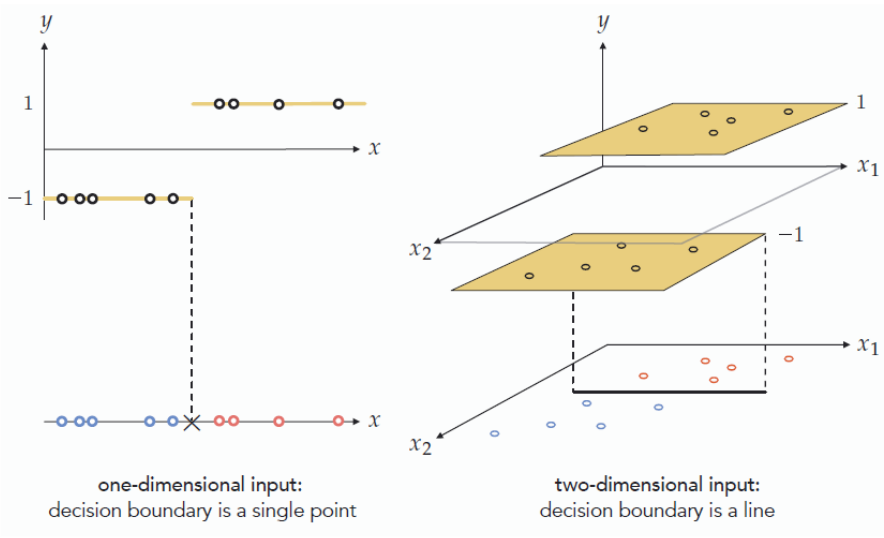
One can analytically derive its gradient:

$$\boxed{\nabla J(\theta) = \frac{1}{m} x^T (h_\theta(x) - y)}$$



Softmax and Perceptron Cost Functions

For class labels y ∈ {0, 1} one can instead use the following cost functions:

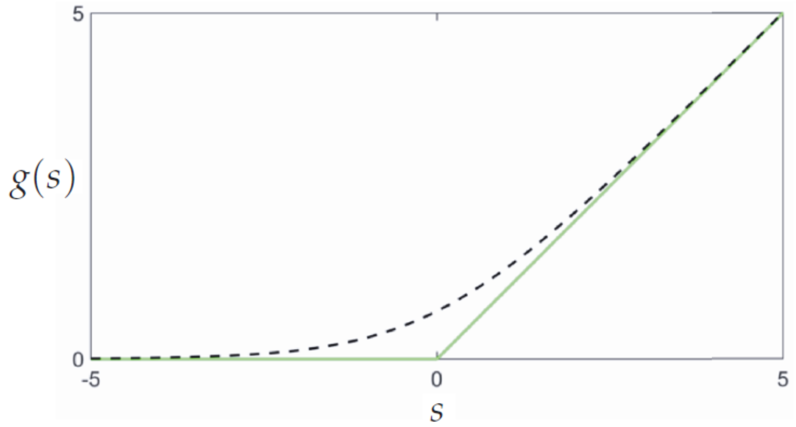


Softmax Cost

$$\boxed{J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} \log(1 + e^{-y^i \theta^T x^i})\right]}$$

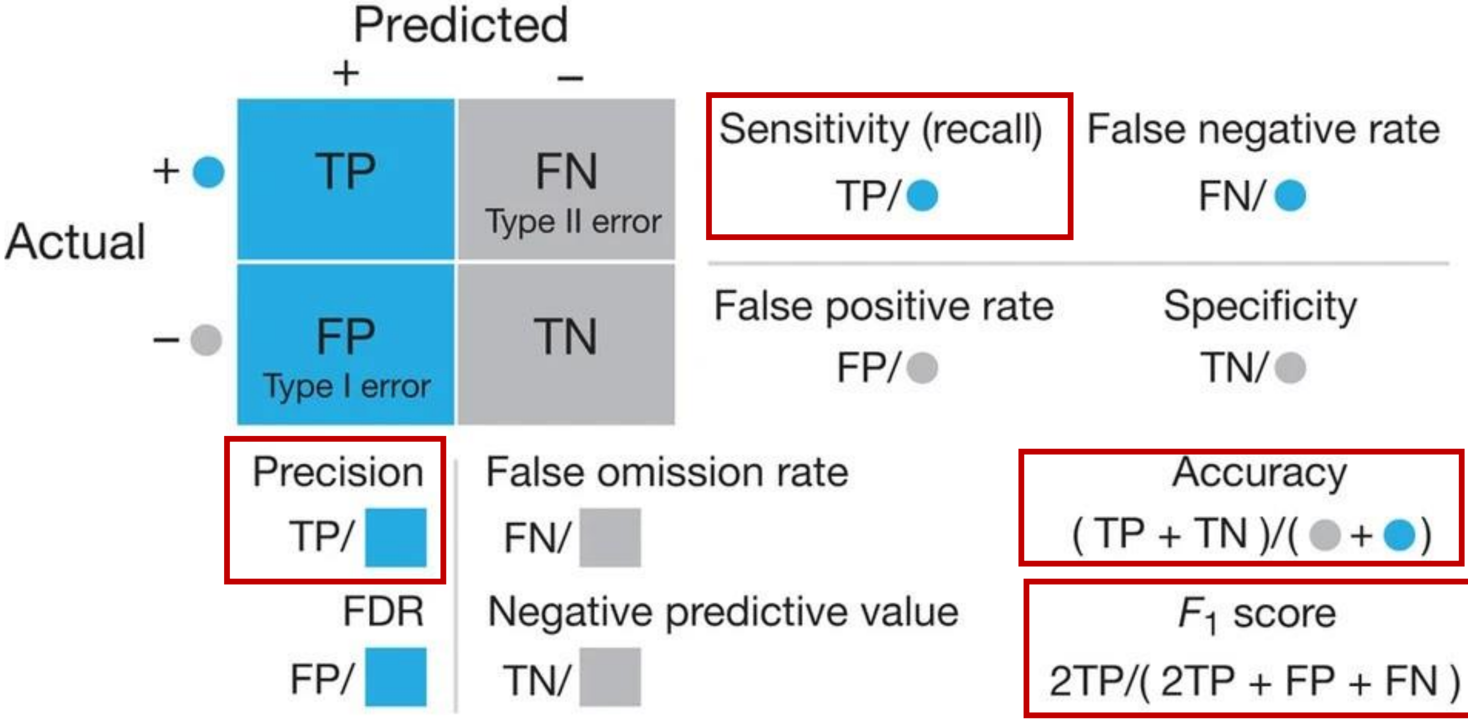
Perceptron Cost

$$\boxed{J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} \max(0, -y^i \theta^T x^i)\right]}$$



All perform similarly for datasets with noise (i.e., not linearly separable). Also very similar to Support Vector Machines (SVMs)

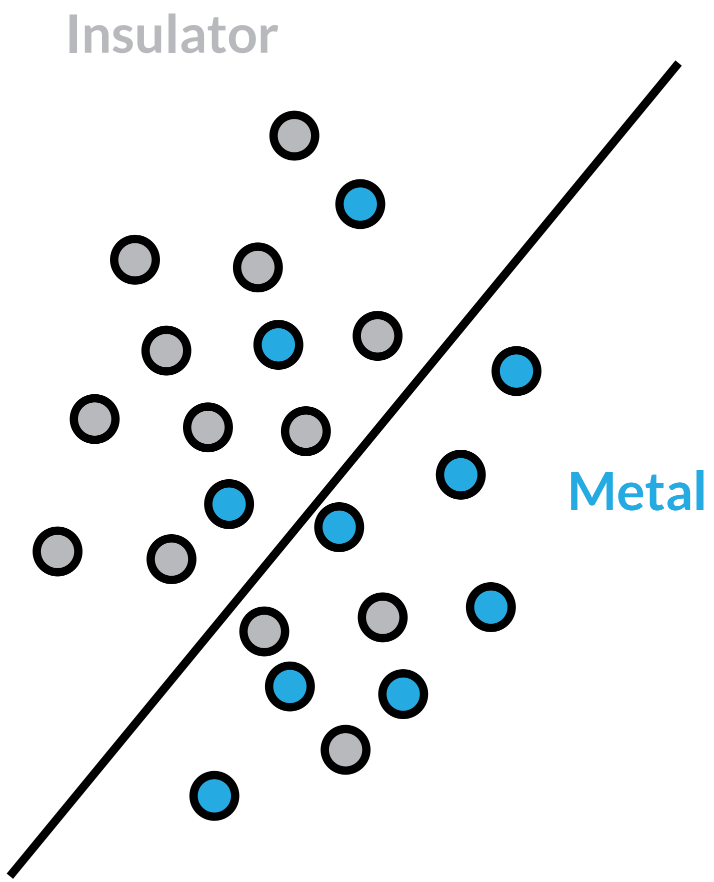
Classification Metrics

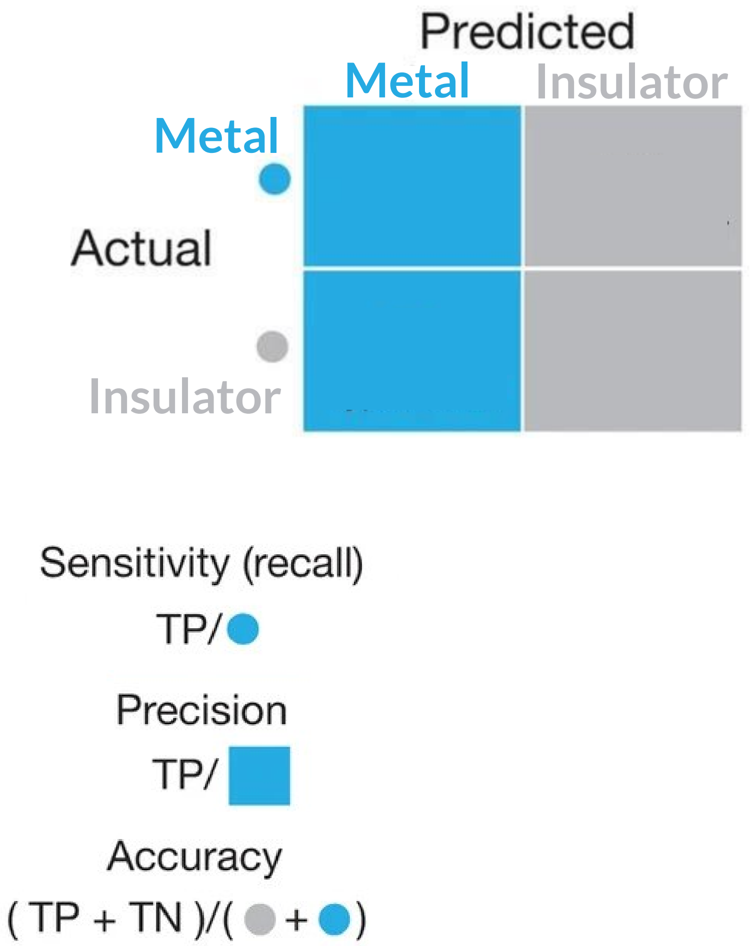


harmonic mean of precision and recall

https://www.nature.com/articles/nmeth.3945/figures/1

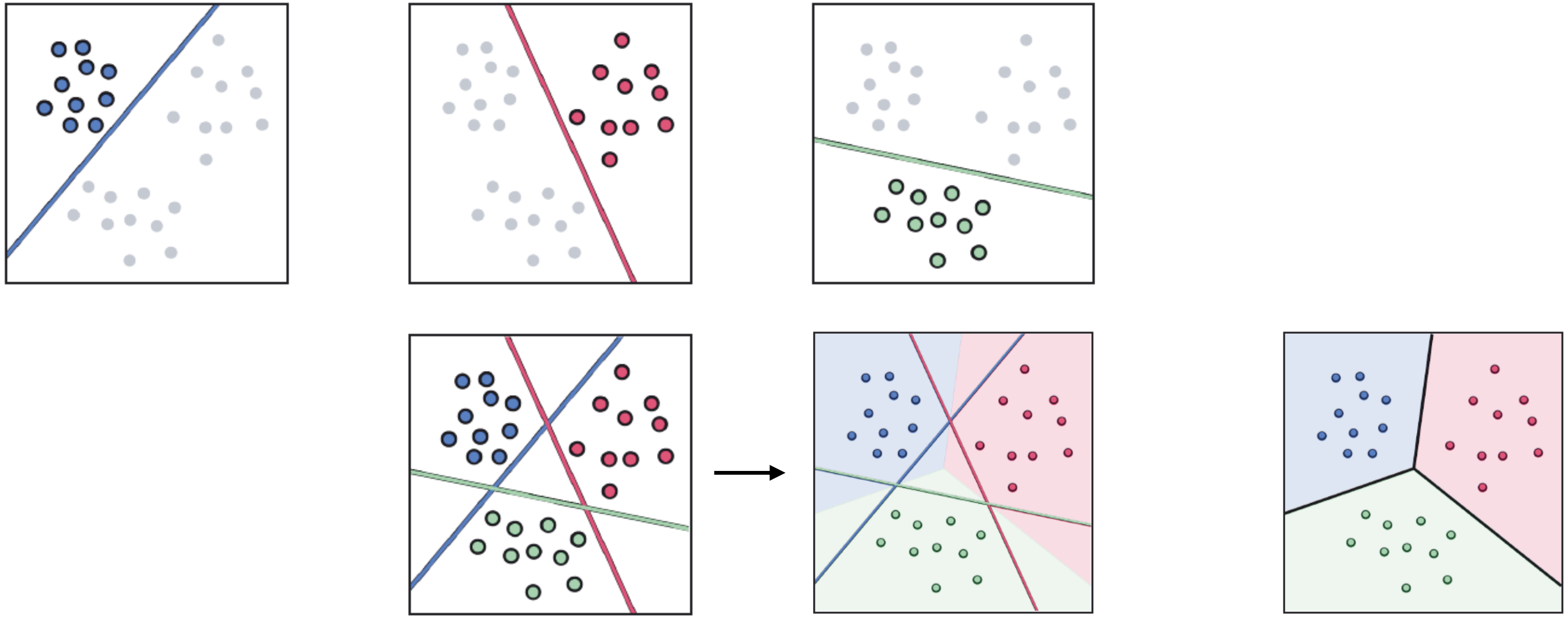
Classification Metrics Example





Multi-Class Classification

Multiple One-versus-Rest or One-versus-All classifiers



Alternative:
Simultaneous minimization of multi-perceptron or multi-softmax cost

“*Machine Learning Refined*”, 2nd Edition, J. Watt, R. Borhani, A. K. Katsaggelos

K-Means Clustering

1. Pick number K of clustering centroids
2. Cluster Assignment: Group points by closest centroid (2-norm)
3. Move centroid to average position of grouped points
4. Check if centroid position converged,
   if not, go to step 2

How to pick K:
Plot performance vs. K and identify "elbow" in the curve

Warning:
Perform multiple times with different initialization

"*Machine Learning Refined*", 2nd Edition, J. Watt, R. Borhani, A. K. Katsaggelos

Dimensionality Reduction: Principal Component Analysis (PCA)

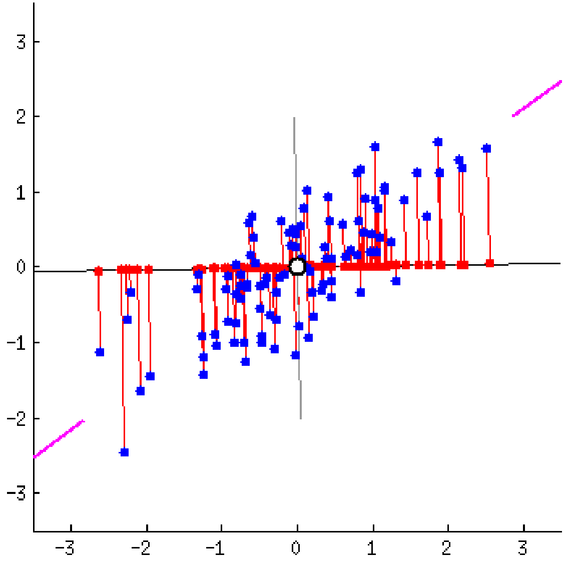
$$C = \frac{1}{m}\sum_{i=1}^{m}(x^i)(x^i)^T$$

$$C \cdot v = \lambda \cdot v$$

1. Feature scaling/mean normalization
2. Compute covariance matrix *C*
3. Eigen-decomposition of *C*
4. Pick the first *k* eigenvectors $v_1, v_2, \ldots, v_k$ for $\lambda_1 > \lambda_2 > \ldots > \lambda_k$
5. Project data onto these principal axes
   (Eigenvalue corresponds to capture variance)

$$z^i = V_{red}^T x^i \quad \rightarrow \quad x_{approx}^i = V_{red} \cdot z^i$$

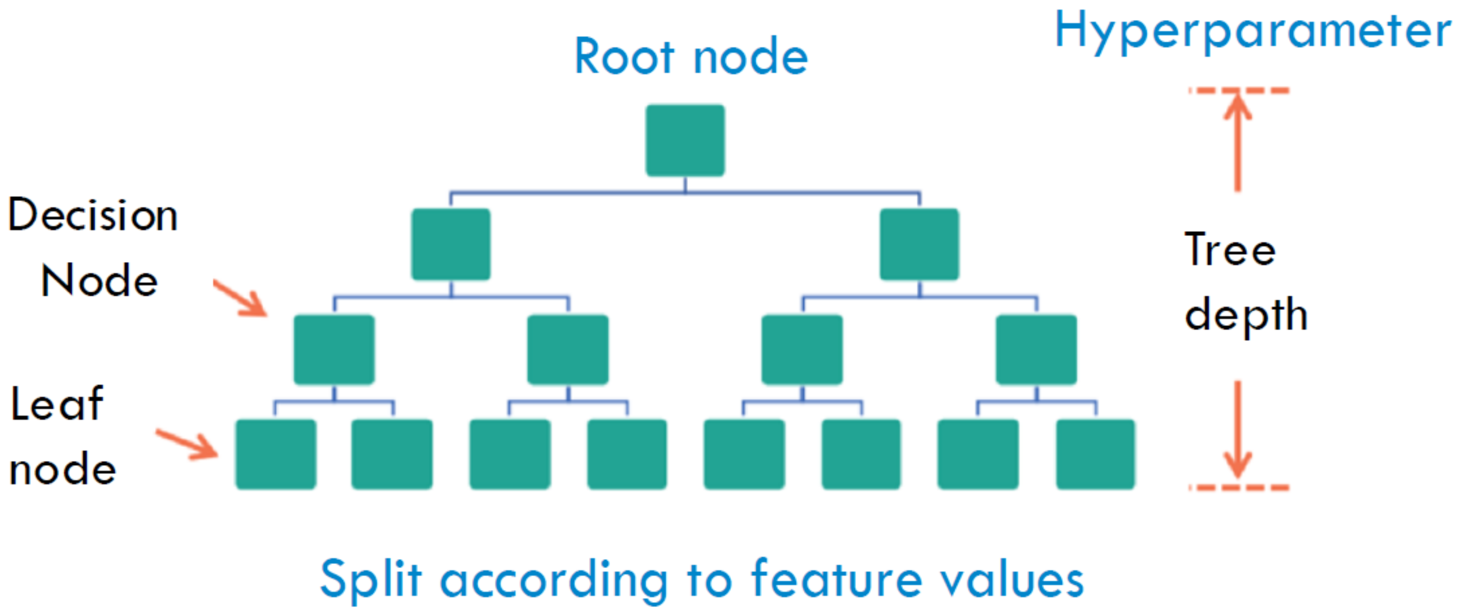
- Can choose *k* based on capture variance target
- For compression to speed up ML algorithms
- Visualization (*k*=2, 3)
- Don't use for reduction of features (regularization).
  PCA doesn't know anything about target label!



stackexchange.com

Decision Tree

Tree-like model splits data multiple times according to feature values (decision rules)



Tree to assign class *N*

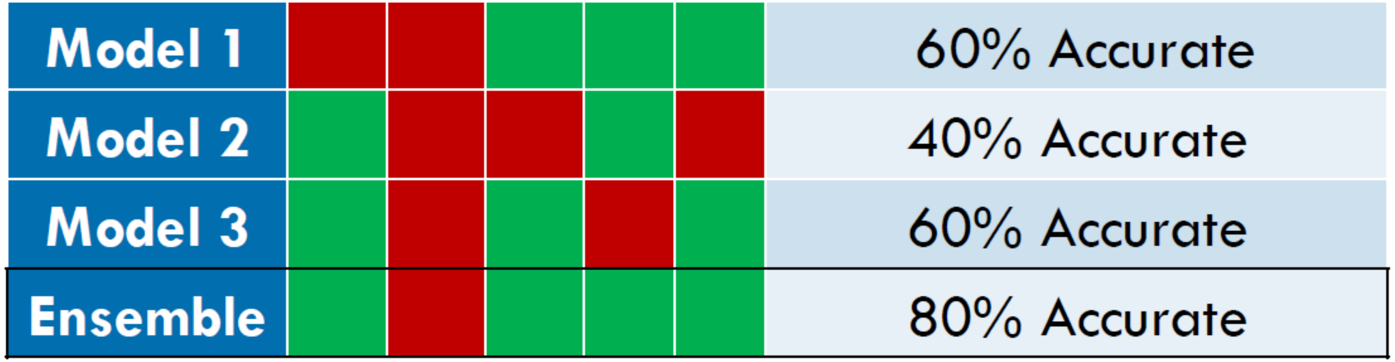
```
              [ Start ]
              Is x2 < 3?
              /        \
           Yes          No
           /              \
   [ Node 1 ]          [ Node 2 ]
   Is x5 < 7?          Is x5 ≥ 7?
    /       \           /       \
  Yes        No       Yes        No
  /           \         \         \
Class A    Class B   Class C    Class D
```

- *Hyperparameters:* no. of trees, max depth, min. samples…
- Powerful, but prone to overfitting
- *Greedy search* (local; not gradient descent).
  Tries all features/thresholds at head node, then continue with each child node,…

By Prof. Aaron Walsh

Ensemble Methods: Random Forest

- Combine predictions from multiple models: Majority voting or Averaging
- Generally leads to higher accuracy (at loss of interpretability)



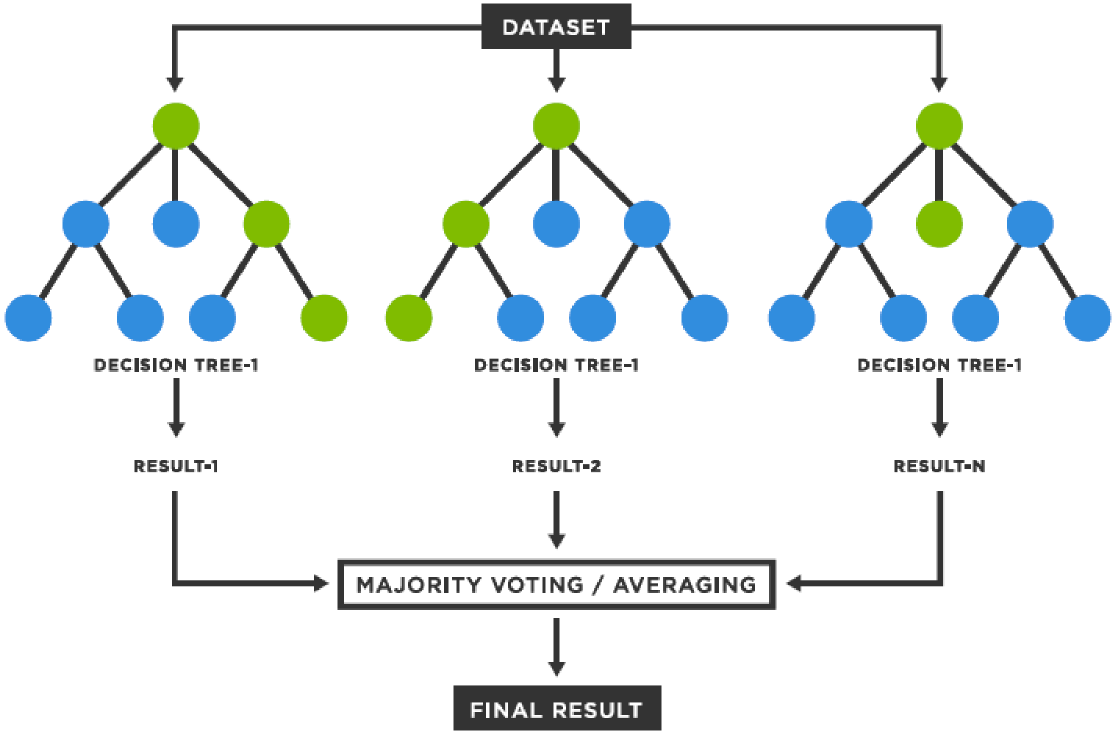
- **Random Forests**: Ensemble of independent decision trees
- **Gradient Boosted Regression:** Ensemble of coupled decision trees

$$\boxed{y^j = \sum_{i=1}^{m} \gamma_i \text{tree}_i(x^j)}$$

By Prof. Aaron Walsh

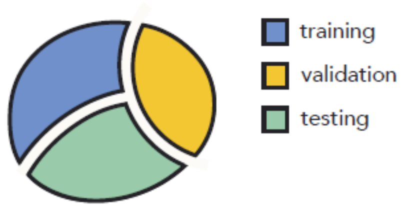
Bagging Method

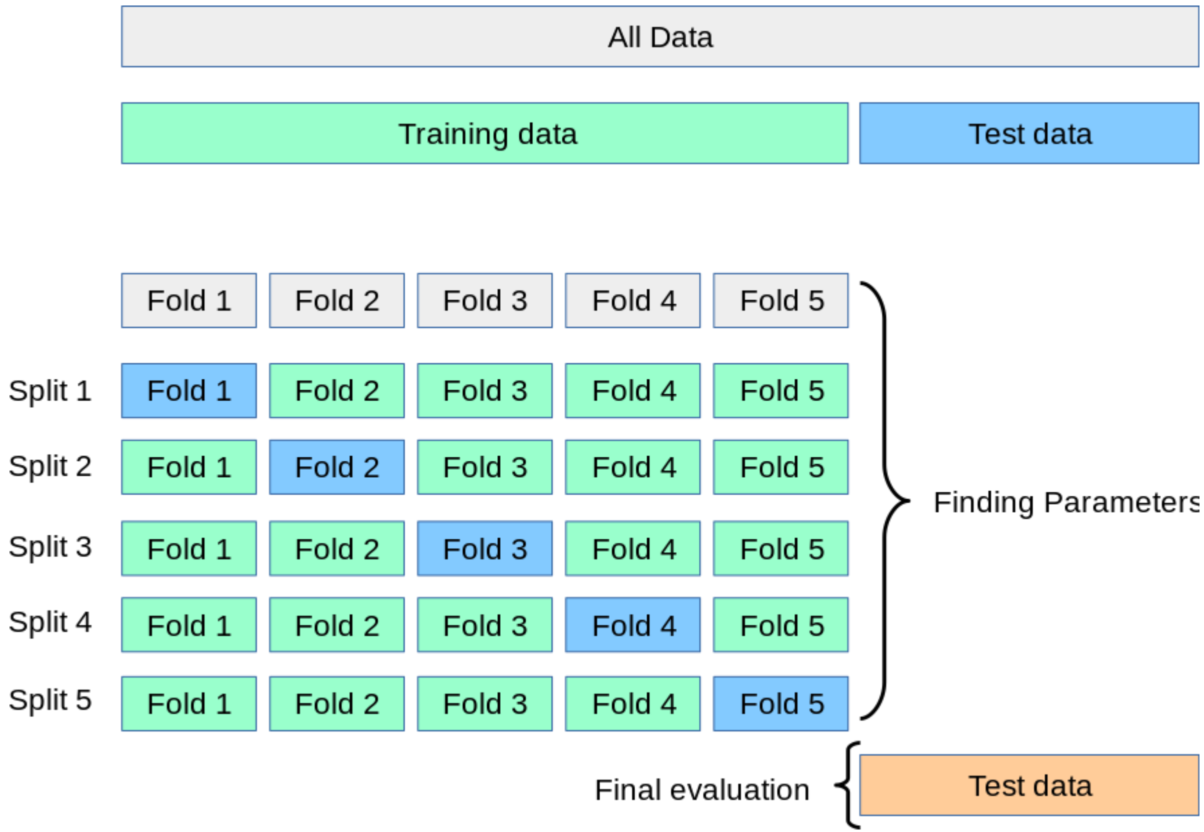
Each tree is generated from a random subset of training data and a random subset of features (bootstrap aggregation)



T. K. Ho, IEEE Trans. Pattern Anal. Mach. Intell. 20, 832 (1995)

Hyperparameter Tuning: *k*-fold Cross-Validation





https://scikit-learn.org/stable/modules/cross_validation.html

Things we Didn't Cover

There are a few more classic model architectures that we left out, including

- Support Vector Machine (SVM) – maybe on homework
- Gaussian Processes – maybe a guest lecture
- The Kernel approach
- Neural Networks (the Perceptron) – later lecture on deep learning

Lecture Feedback

Please, scan the QR code and take a minute to let me know how the lecture was and mention any **feedback/questions**

This form is **anonymous!**