The Machine Learning Landscape

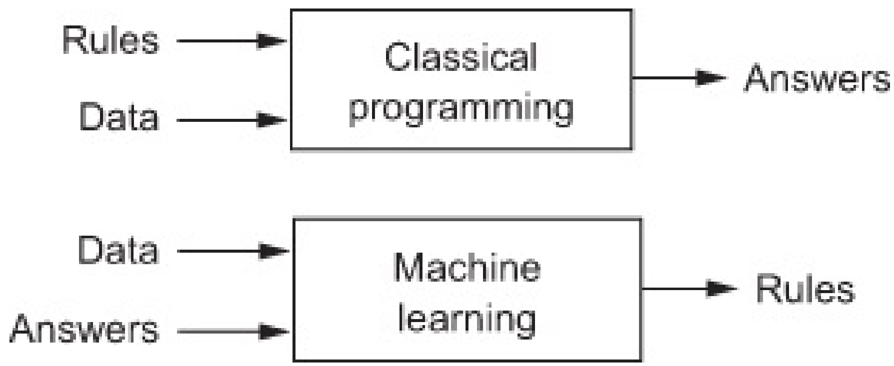
Vineet Bansal
Research Software Engineer, Center for Statistics & Machine Learning

vineetb@princeton.edu
Oct 31, 2018

CENTER FOR STATISTICS AND MACHINE LEARNING

What is ML?

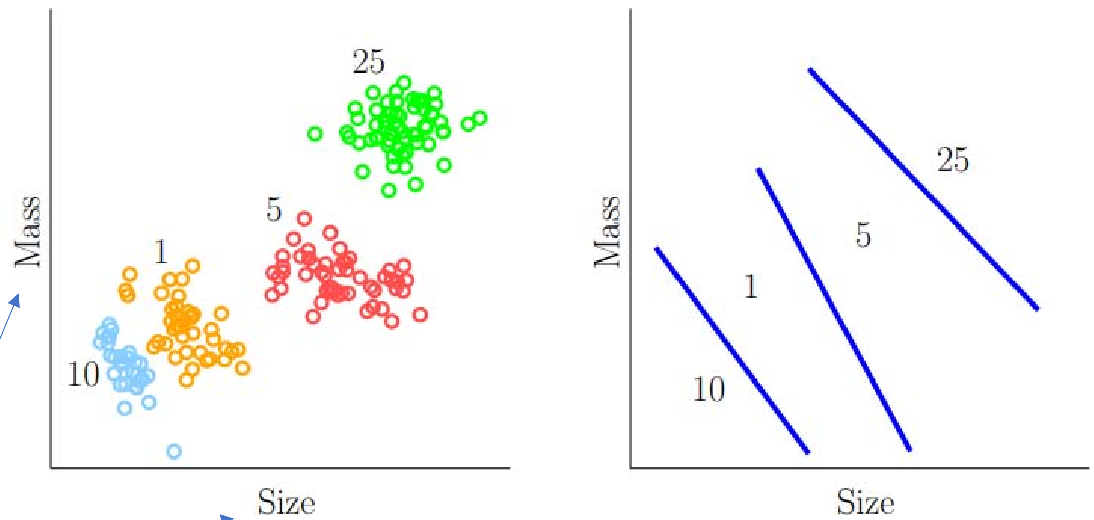
“A field of study that gives computers the ability to learn without being explicitly programmed.”



A machine-learning system is *trained* rather than explicitly programmed.

Types of ML Systems

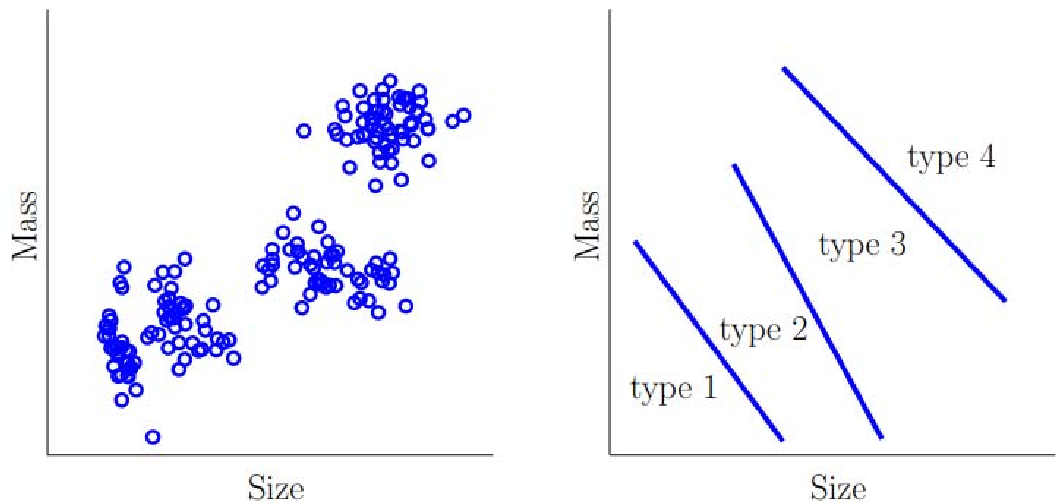
Supervised Learning - Training Data contains desired solutions, or *labels*



Features

Types of ML Systems

Unsupervised Learning - Training Data is *unlabeled*



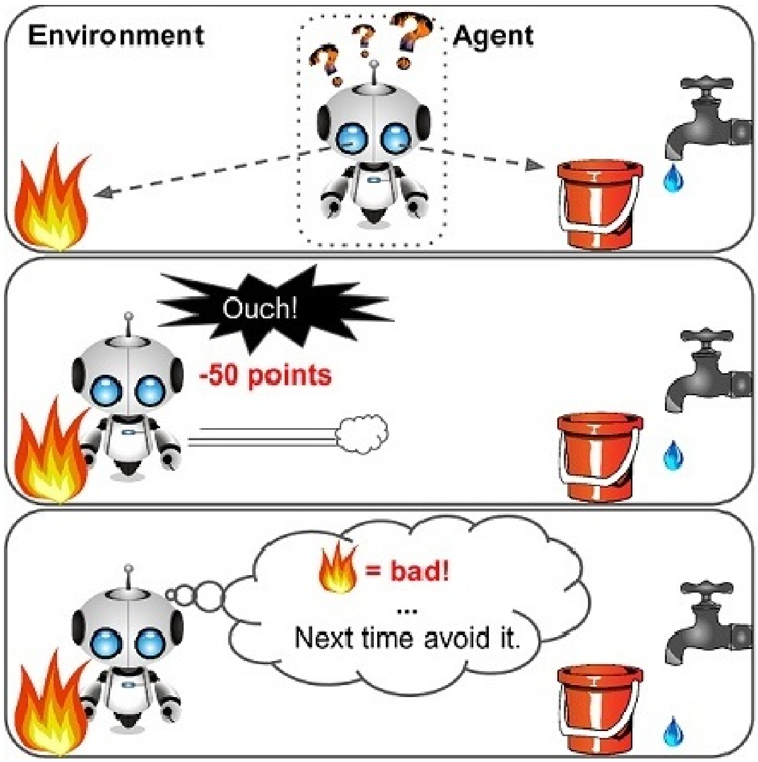
Types of ML Systems

Reinforcement Learning - Training Data does not contain target output, but instead contains some possible output together with a measure of how good that output is.

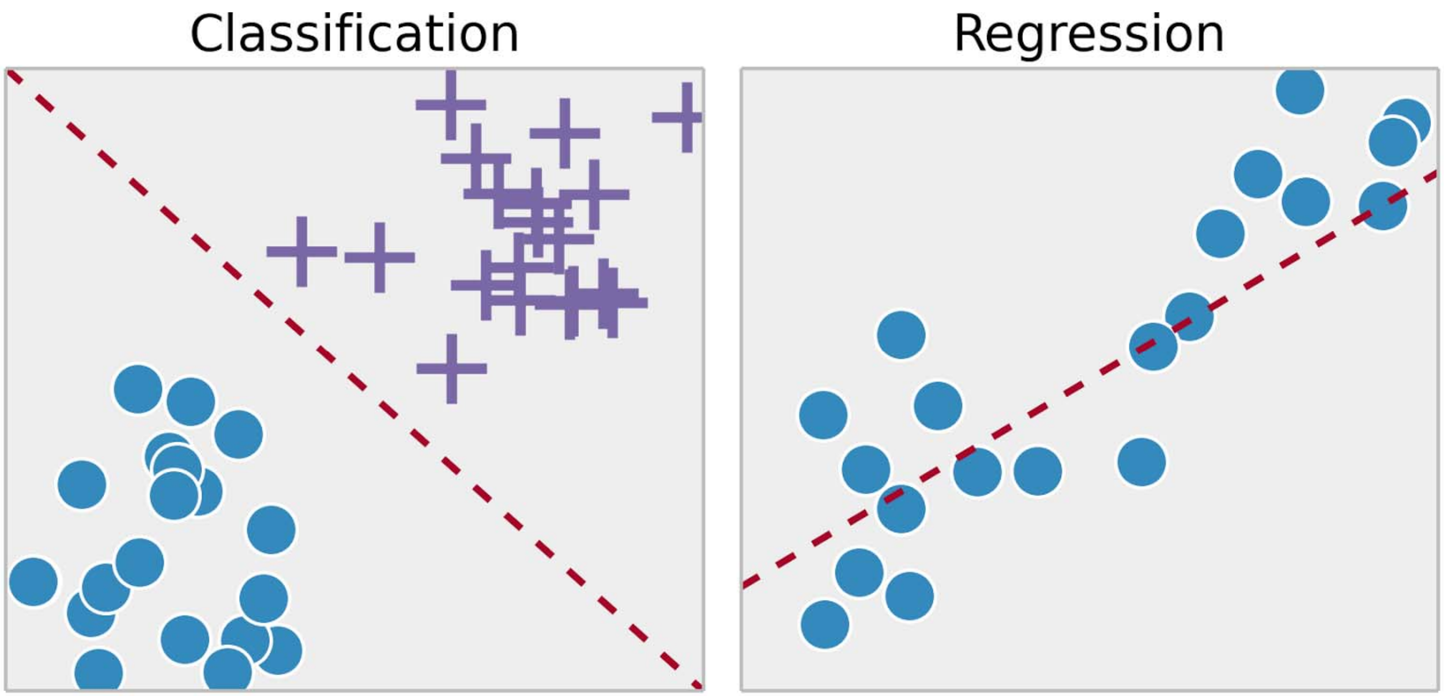
<input>, <correct output>

↓

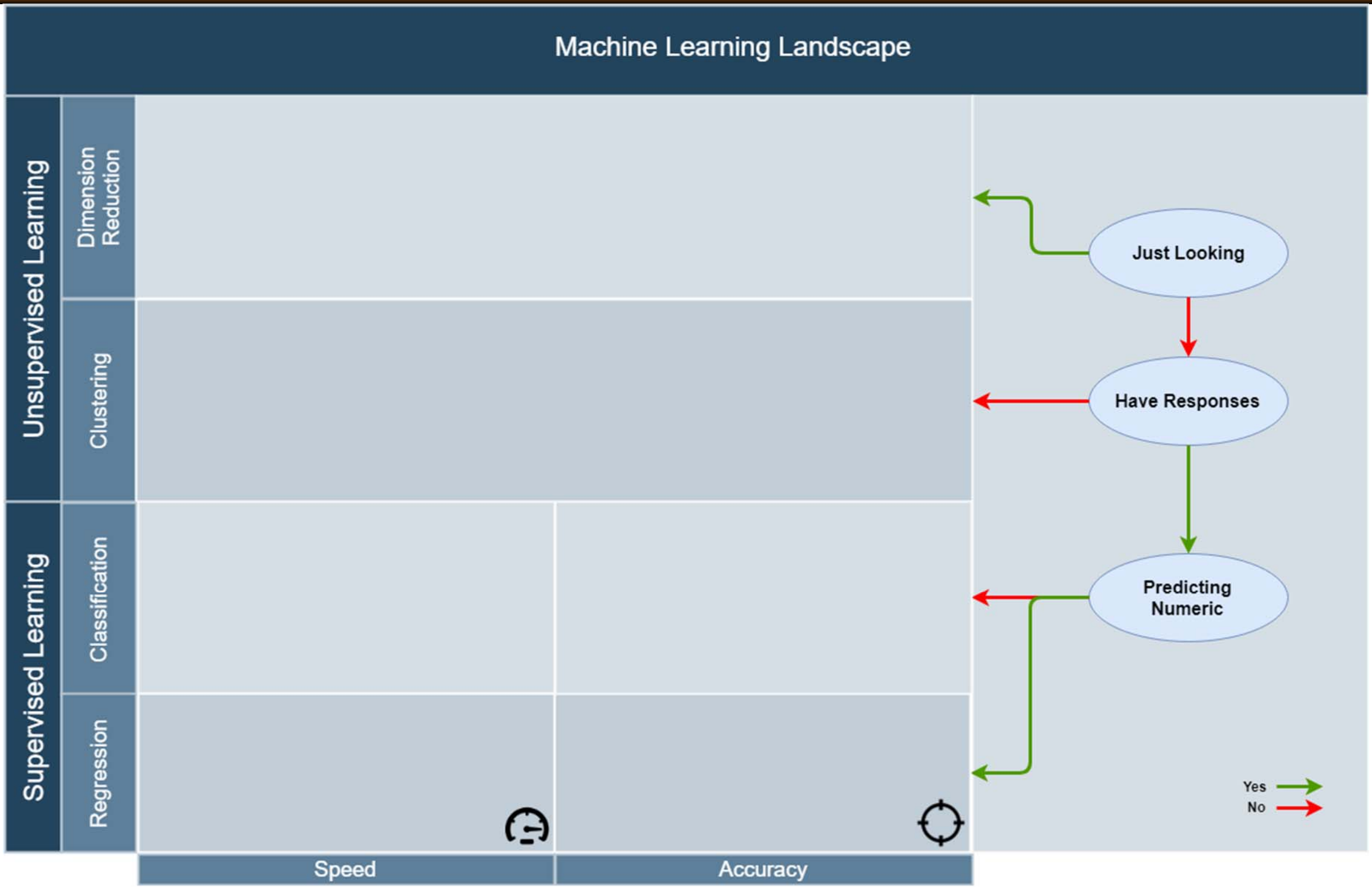
<input>, <some output>, <grade for this output>



Classification vs Regression



ML Landscape



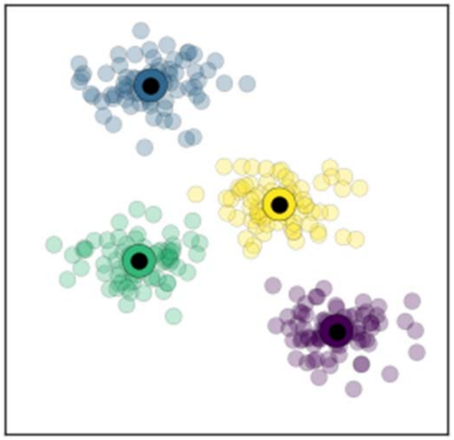
Unsupervised Learning - Clustering

Clustering

- Color “clusters” of points in a homogenous cloud of data.

- **Use Cases**

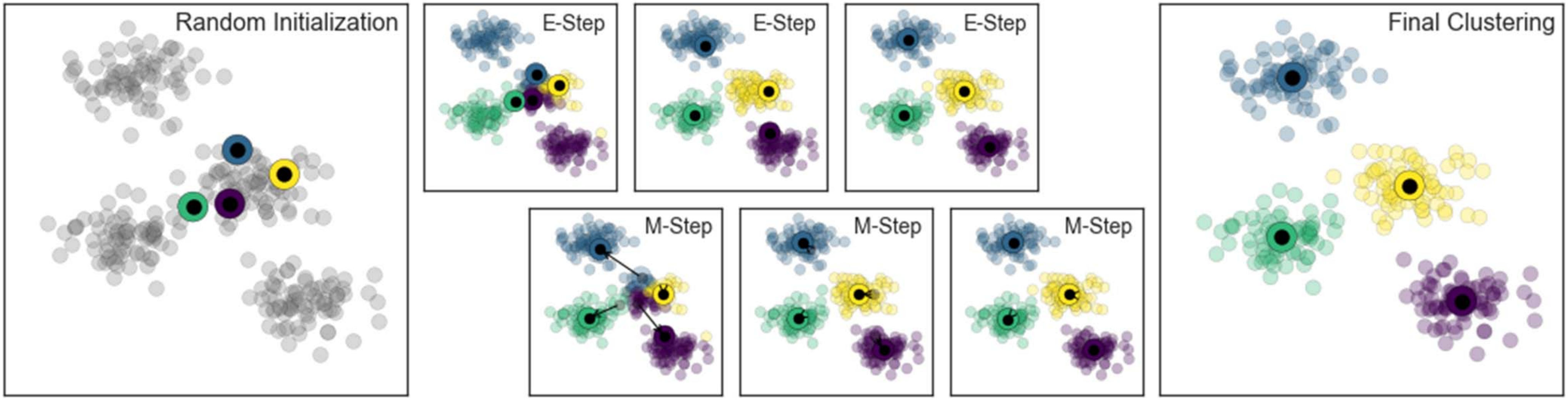
  - Behavioral Segmentation in Marketing
  - Useful as a pre-processing step before applying other classification algorithms.
    - Cluster ID could be added as feature for each data point.



Unsupervised Learning - Clustering

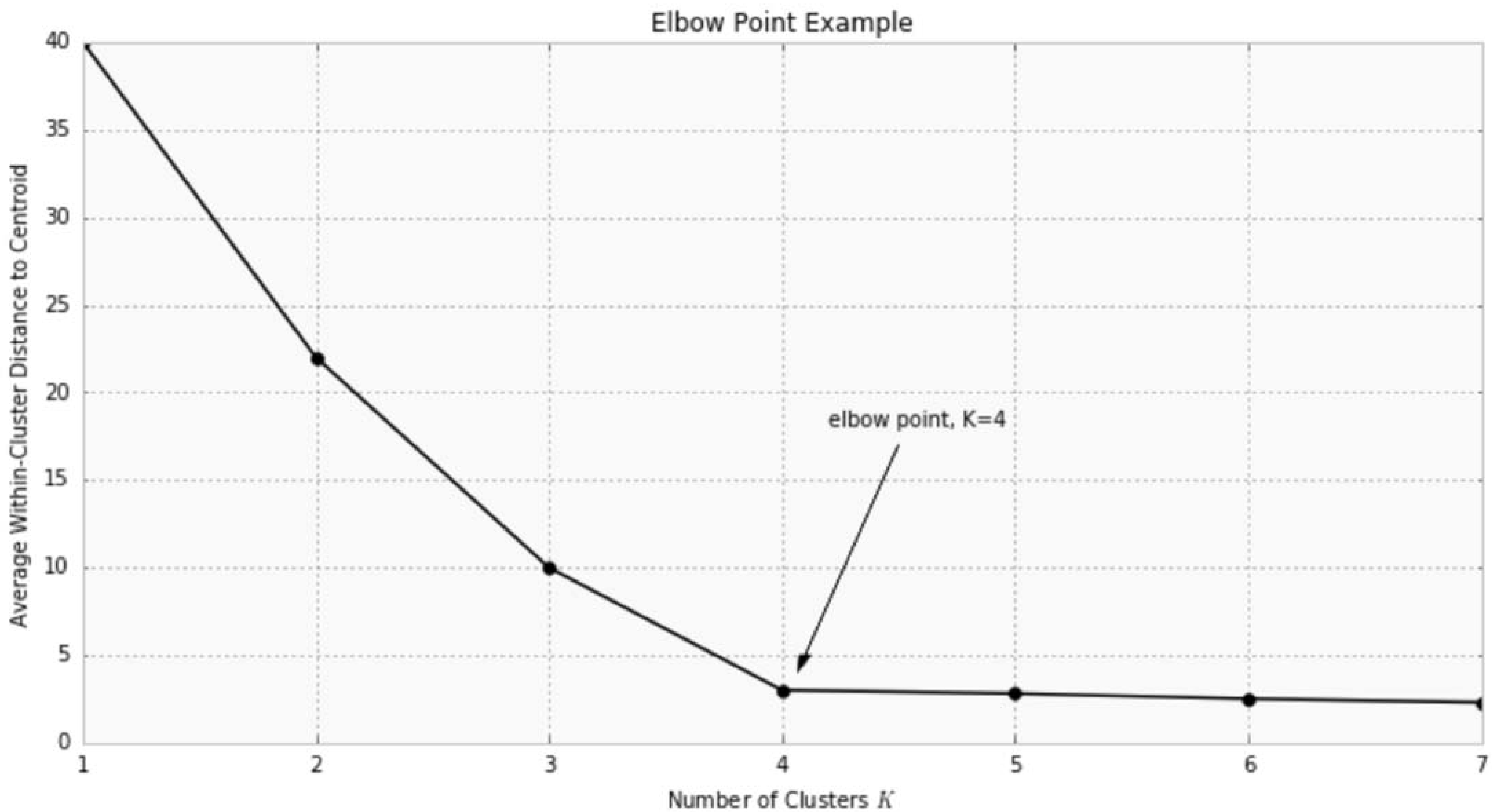
k-means Algorithm

- Guess some cluster centers
- Repeat until converged
  - *E-Step*: assign points to the nearest cluster center
  - *M-Step*: set the cluster centers to the mean



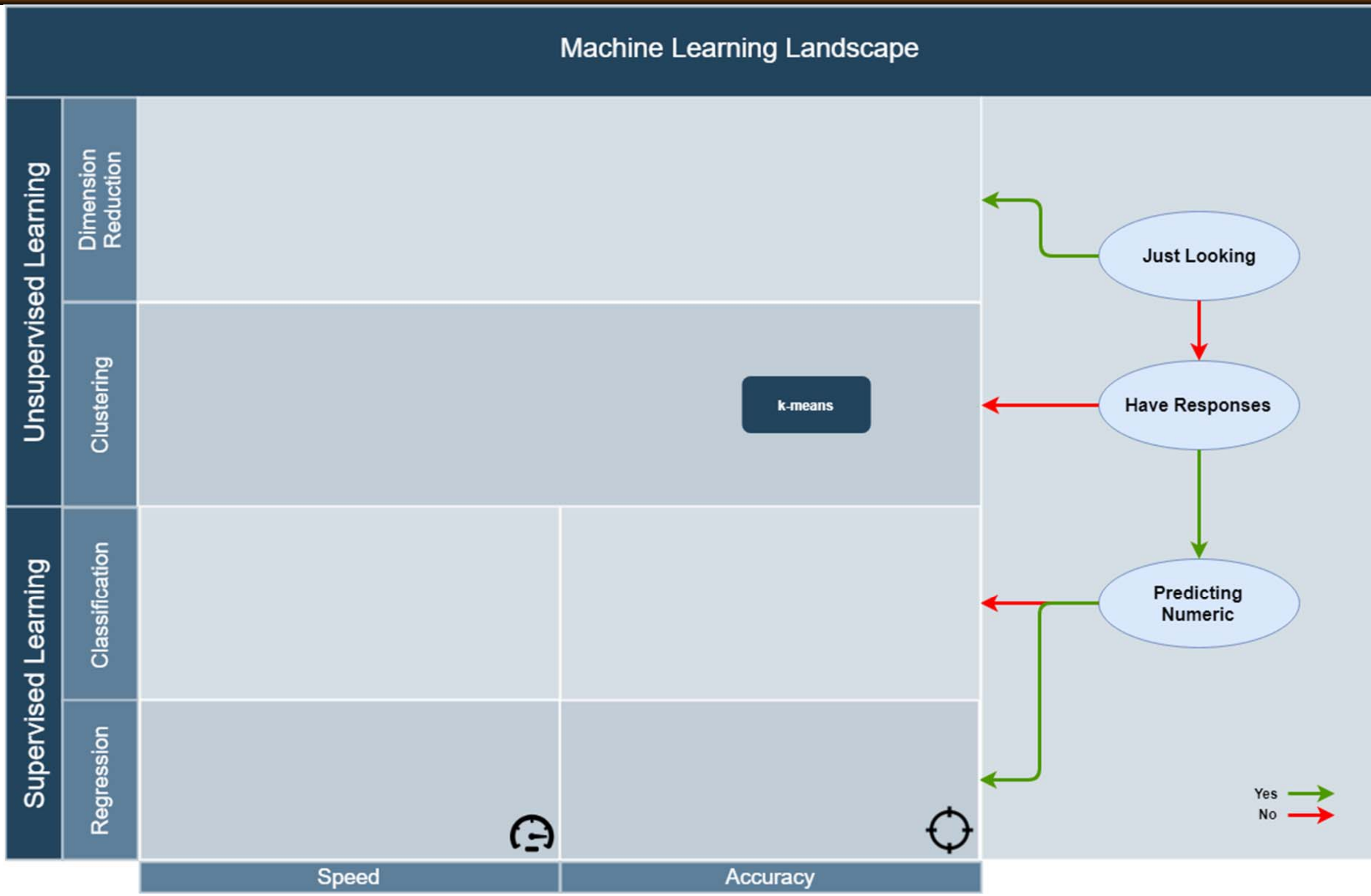
Unsupervised Learning - Clustering

Choosing k

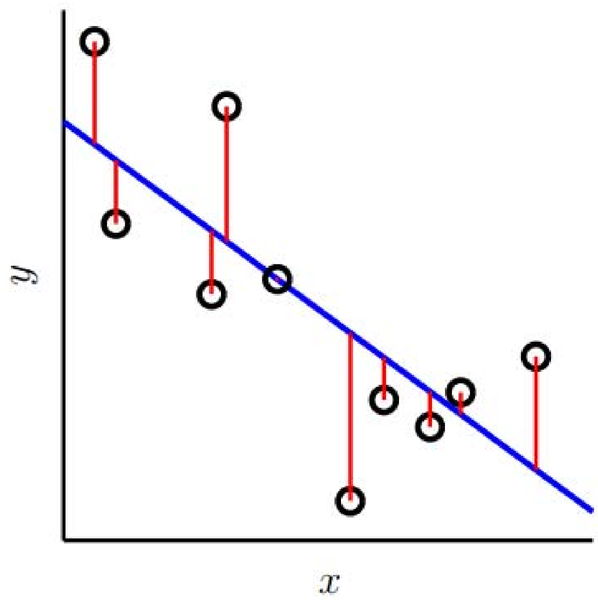


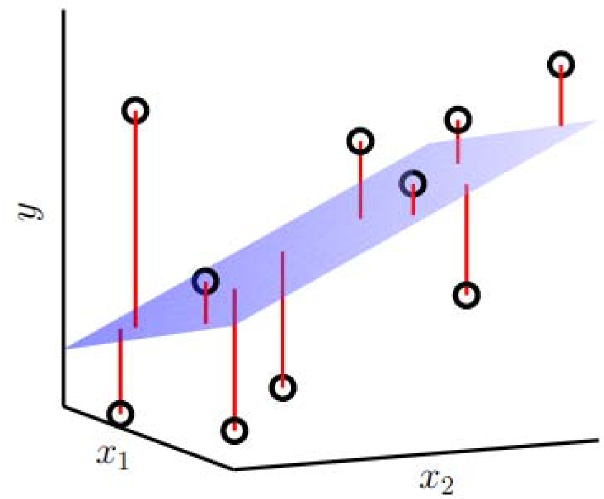
CENTER FOR STATISTICS AND MACHINE LEARNING

ML Landscape



Linear Regression





Linear Regression

$$X = \begin{bmatrix} 1 & x_1{}^1 & x_2{}^1 & \dots & x_n{}^1 \\ 1 & x_1{}^2 & x_2{}^2 & \dots & x_n{}^2 \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & x_1{}^m & x_2{}^m & \dots & x_n{}^m \end{bmatrix}$$

$$y = \begin{bmatrix} y^1 \\ y^2 \\ \dots \\ y^m \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_n \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \dots \\ \hat{y}^m \end{bmatrix}$$

Define a **Hypothesis**

$$h_\Theta(X) = \hat{y} = X\,\Theta$$

Define a **Cost Function** (a measure of how bad we're doing)

$$\mathrm{MSE}(X, h_\Theta) = \frac{1}{m}\sum_{i=1}^{m}[\hat{y}^i - yi]^2$$

Repeat until convergence:

- Calculate Cost Function on chosen Θ
- Calculate slope of Cost Function
- Tweak Θ so as to move downhill (reduce Cost Function value)

Θ is now optimized for our training data.

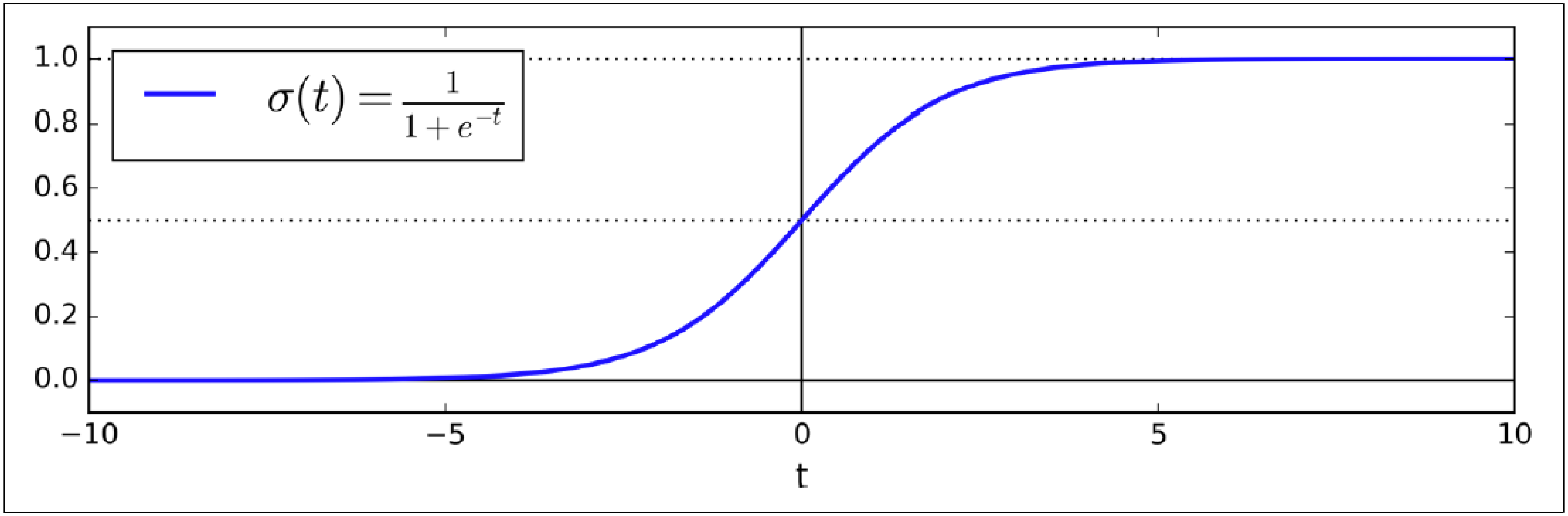
Logistic Regression

Used to estimate the probability that an instance belongs to a particular class.

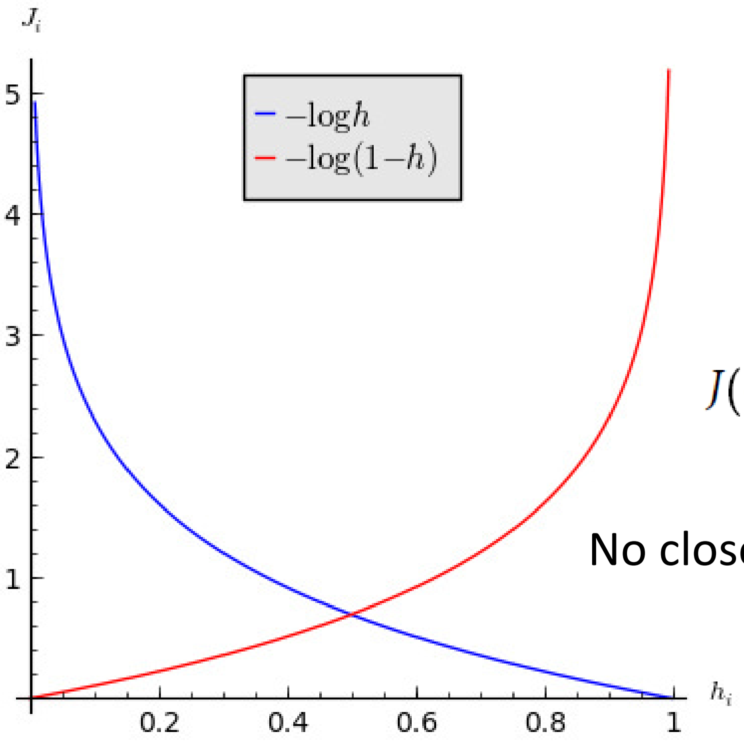
$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

Logistic Regression

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Logistic Regression

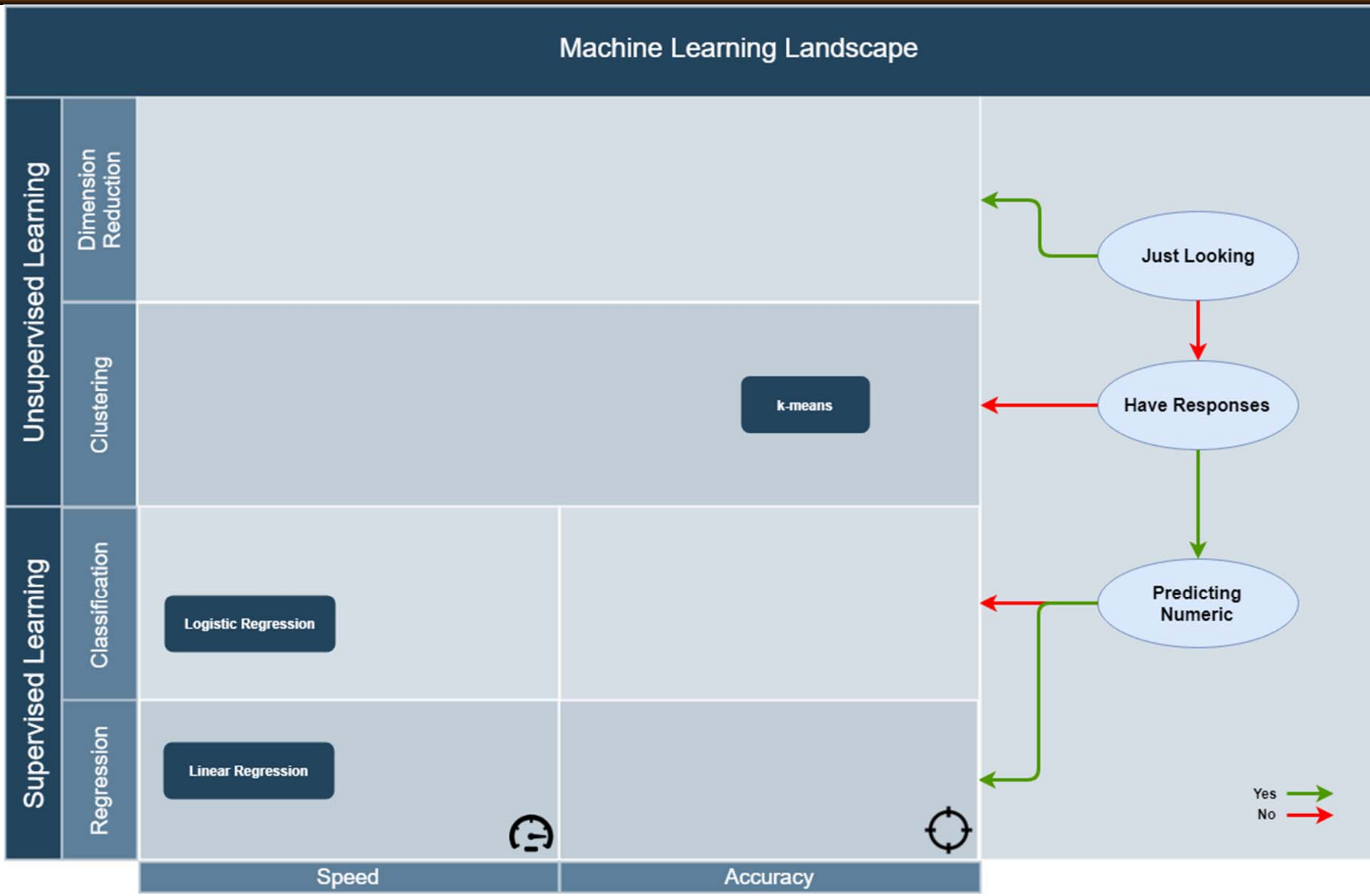


$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1-\hat{p}) & \text{if } y = 0. \end{cases}$$

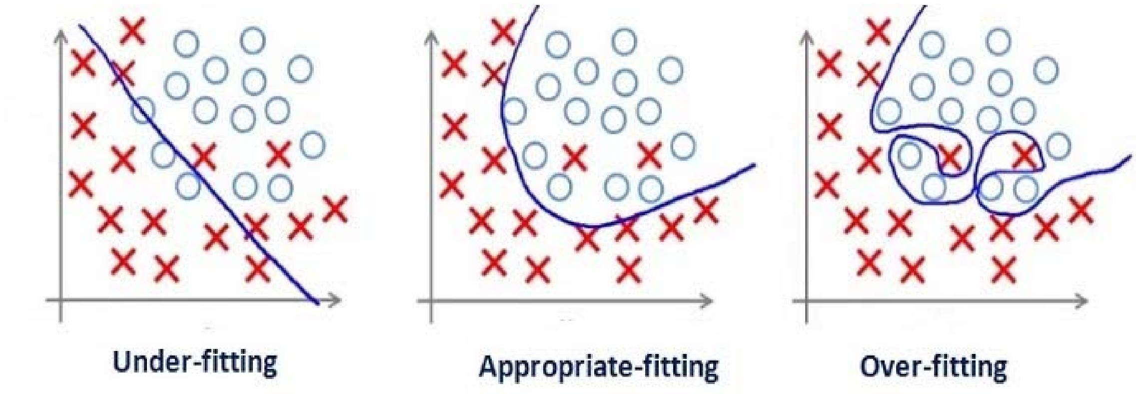
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)log\left(1 - \hat{p}^{(i)}\right)\right]$$

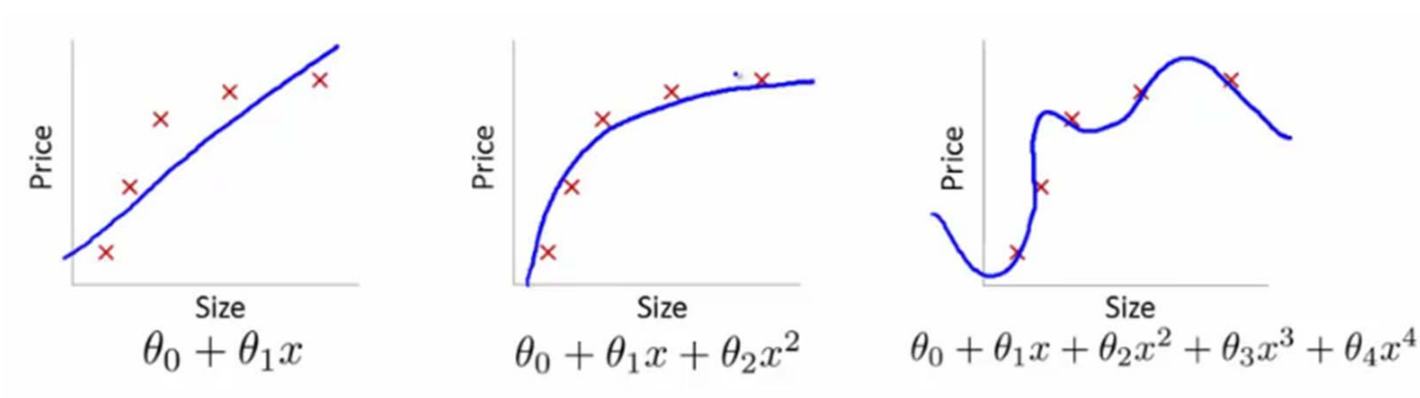
No closed-form solution, but we can use Gradient Descent!

ML Landscape

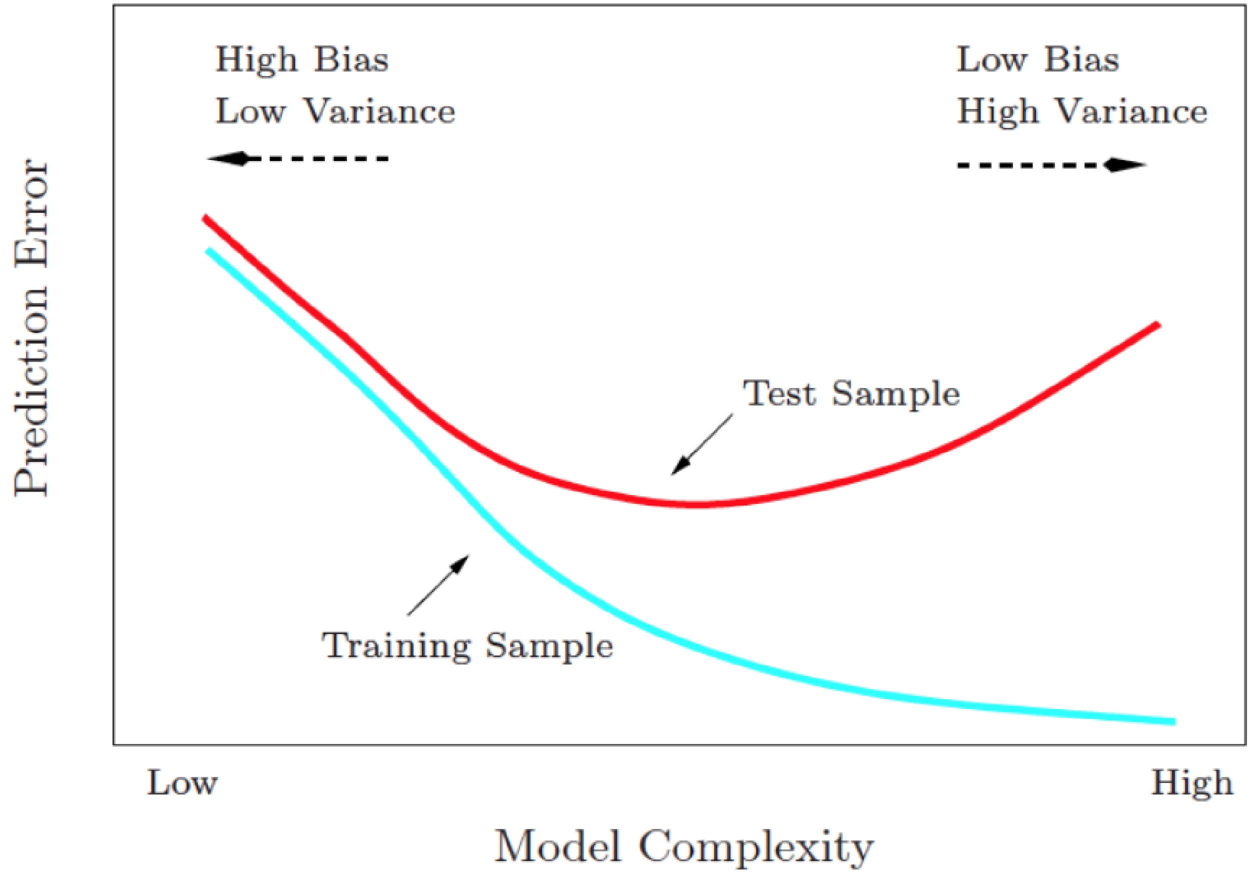


Overfitting and Underfitting





Bias-Variance Tradeoff



Regularization

How to ensure that we're not *overfitting* to our training data?

Impose a small penalty on model complexity.

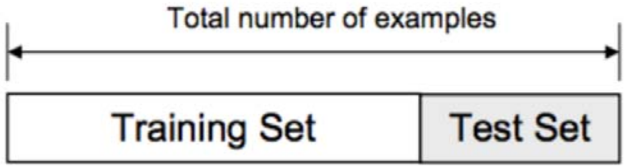
l1 penalty (Lasso Regression)

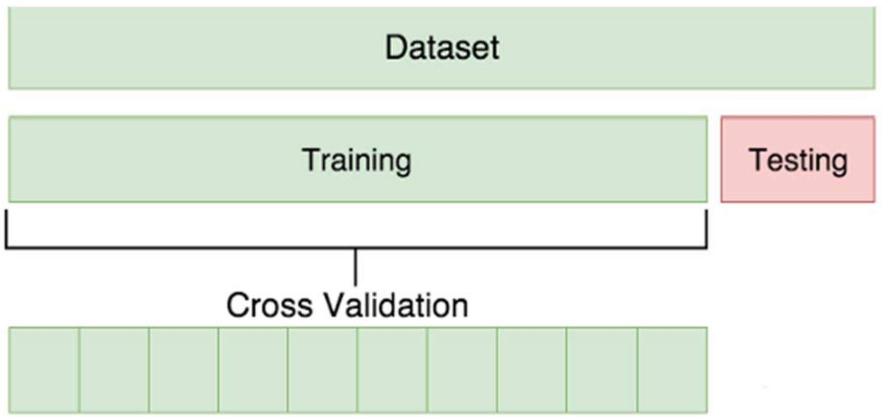
$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

l2 penalty (Ridge Regression)

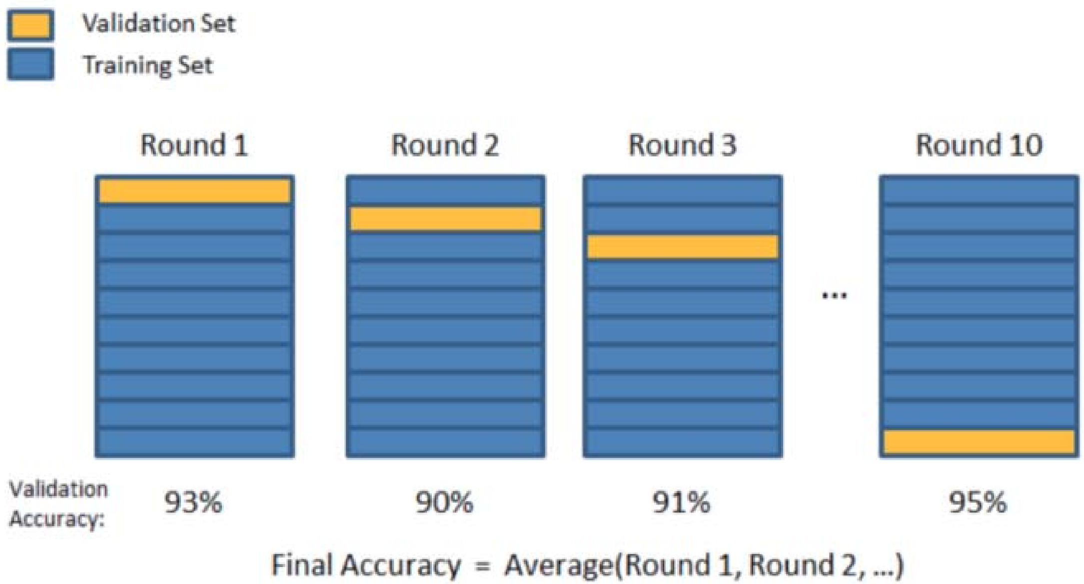
$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Testing and Validation





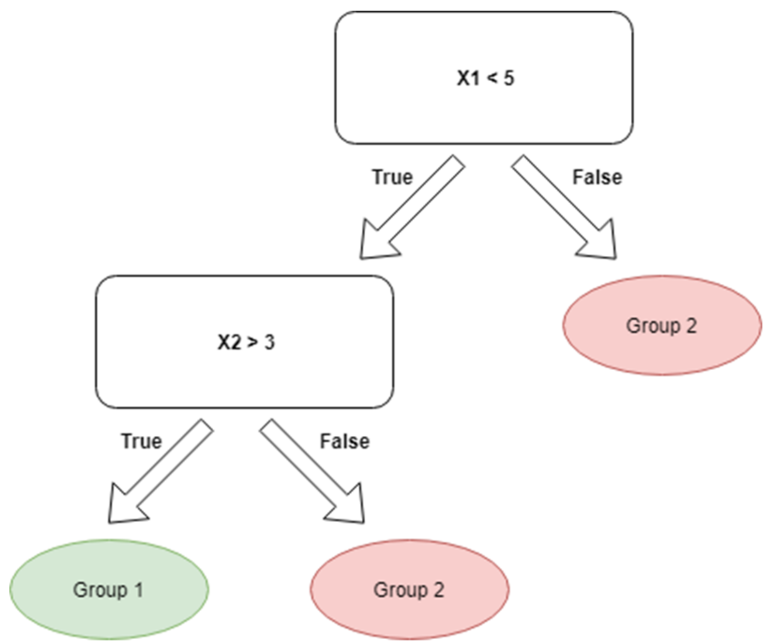
K-fold Cross Validation



Decision Tree

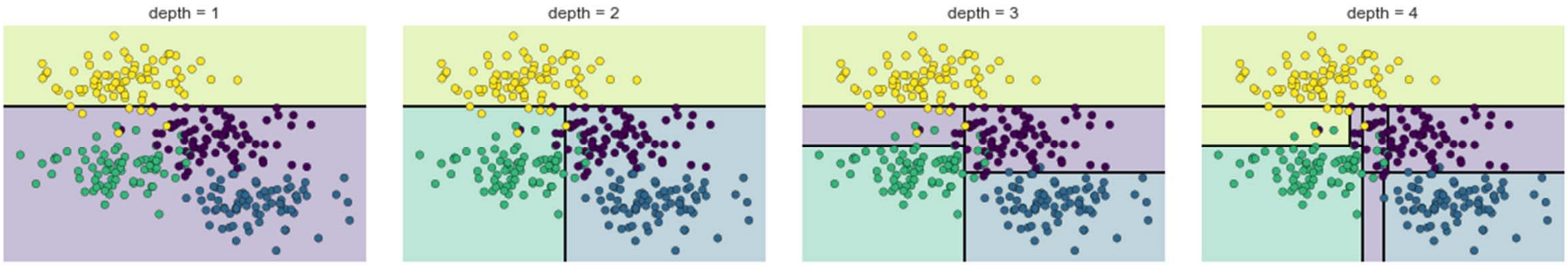
Basic Idea

- Construct a tree to ask a series of questions from your data.



Decision Tree

Let's see how it works on a *real* dataset.



Decision Tree

How is the tree built?

- Define a *Cost Function* that measures the *impurity* of a node.
- A node is *pure* (impurity = 0) if all training instances it applies to belong to the same class.
- One possible impurity measure is *Gini*:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

- $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

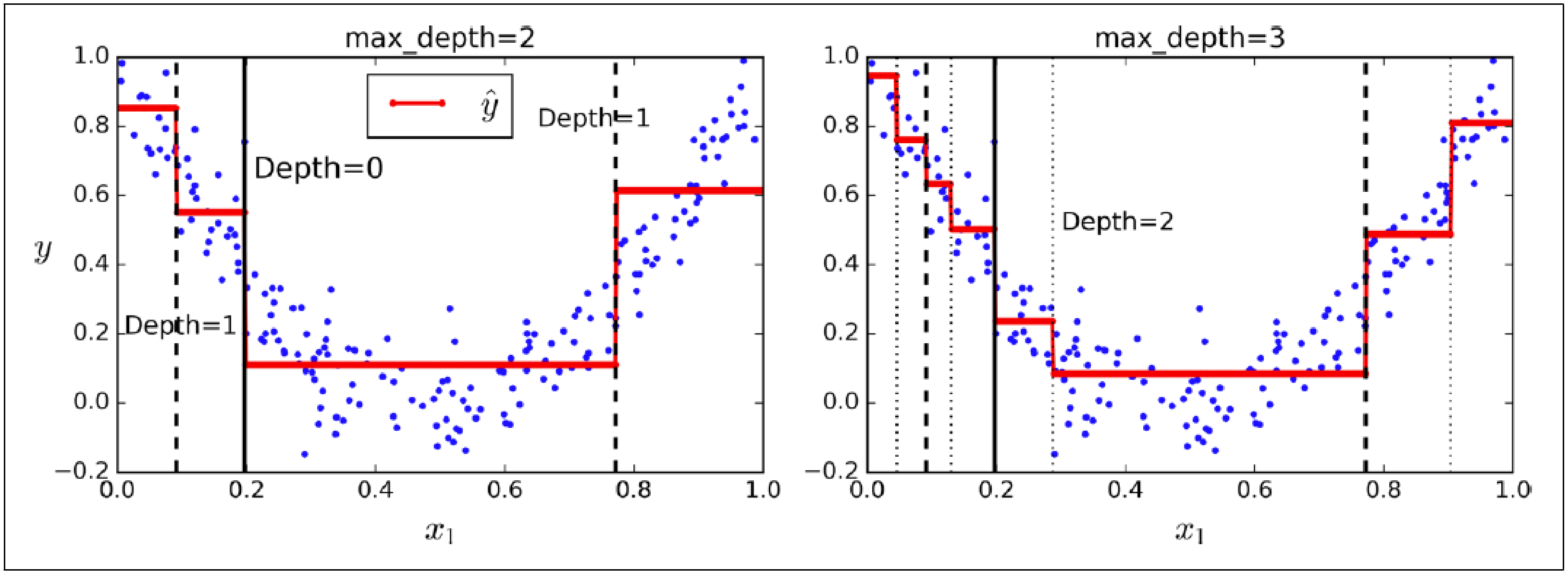
- Search for a feature and threshold that minimizes our Cost Function.
  - Gini scores of subsets thus produced are weighted by their size.
- *Greedy Algorithm* – may not produce the optimum tree.

CART Algorithm. ID3 Algorithm for non-binary trees.

Decision Tree

Decision Trees can be used for regression!

- Minimize MSE instead of impurity.



Decision Tree

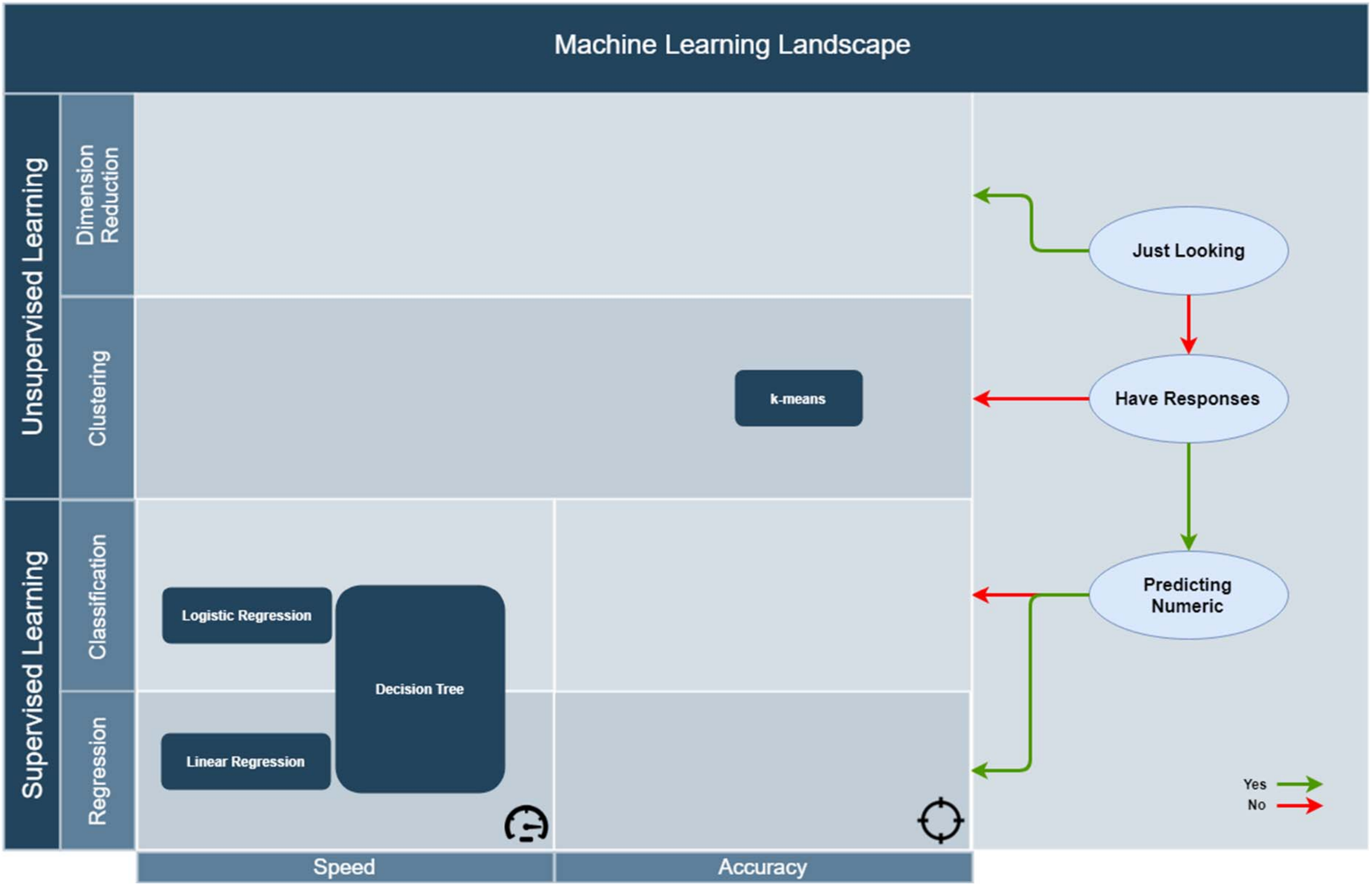
Advantages

- *White Box* – easily interpretable

Disadvantages

- Prone to overfitting
    - Regularize by setting maximum depth
- Comes up only with orthogonal boundaries
    - Sensitive to training set rotation – Use PCA!

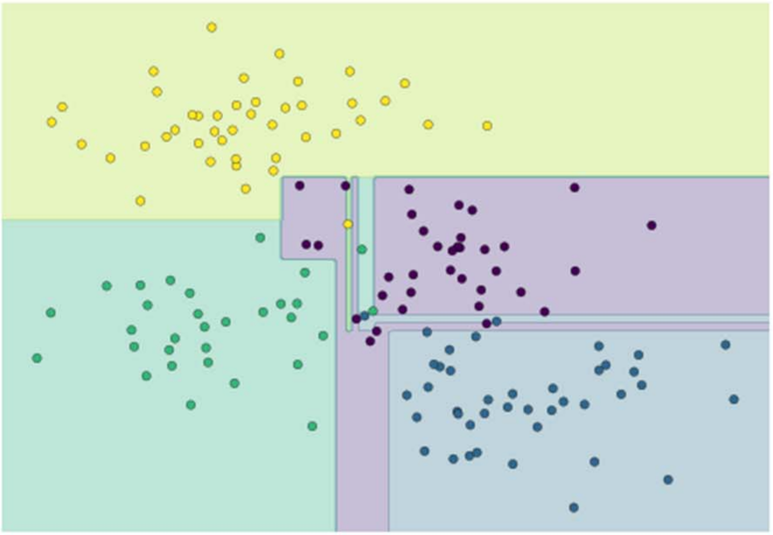
ML Landscape

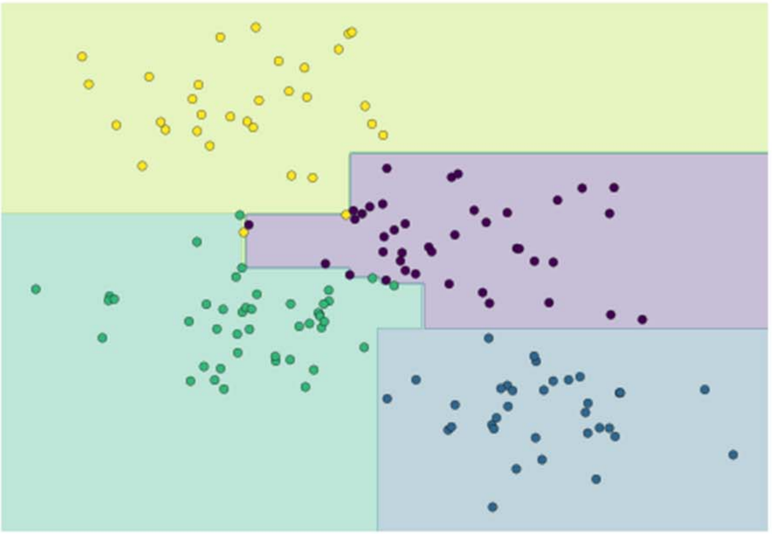


Ensemble Methods

Basic Idea

- Two Decision Trees by themselves may overfit. But combining their predictions may be a good idea!

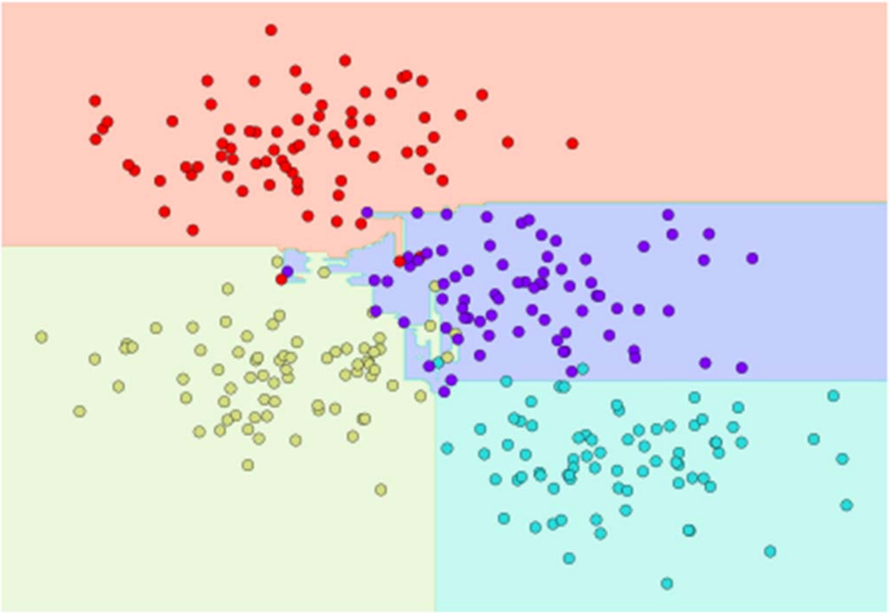




Ensemble Methods

Basic Idea

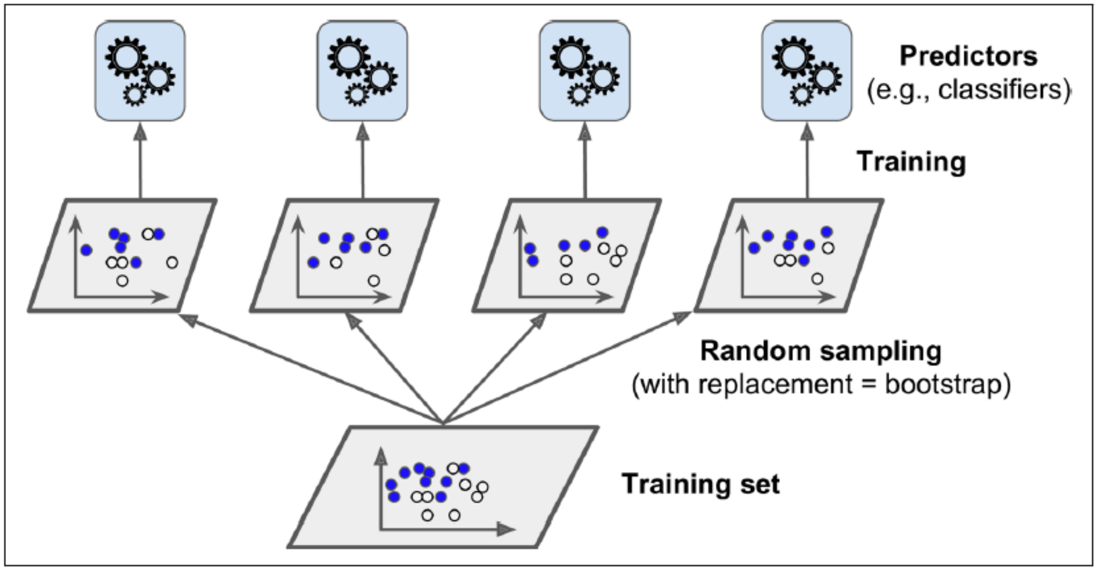
- Two Decision Trees by themselves may overfit. But combining their predictions may be a good idea!



Bagging

Bagging = Bootstrap Aggregation

- Use the same training algorithm for every predictor, but train them on different random subsets of the training set.

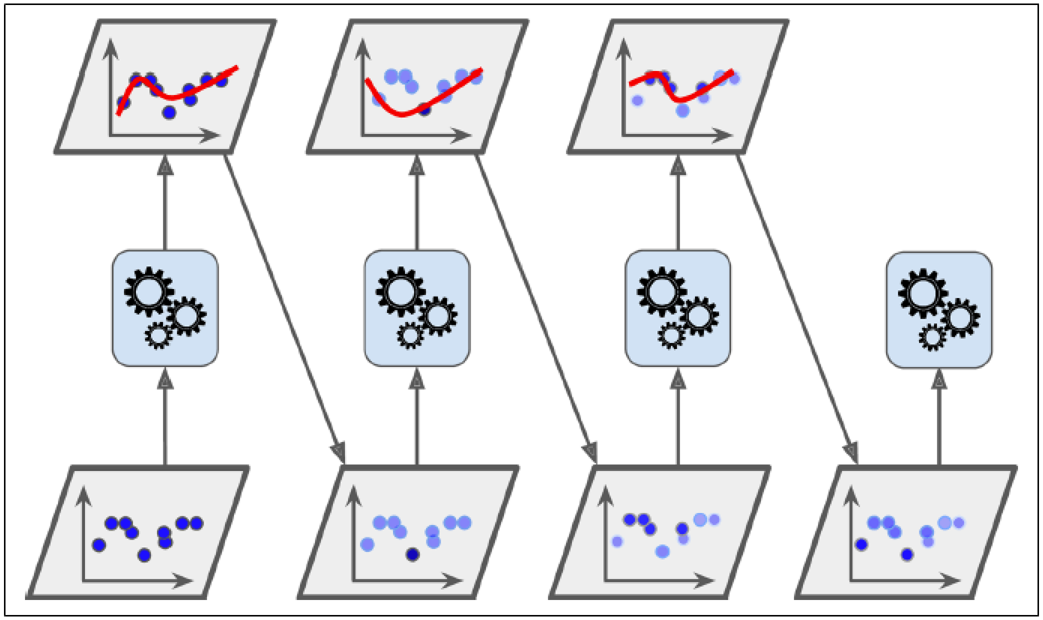


Random Forest is an *Ensemble* of Decision Trees, generally trained via the *bagging* method.

Boosting

Basic Idea

- Train several weak learners *sequentially*, each trying to correct the errors made by its predecessor.



Adaptive Boosting (ADABoost)

- Give more relative weight to the misclassified instances.

Boosting

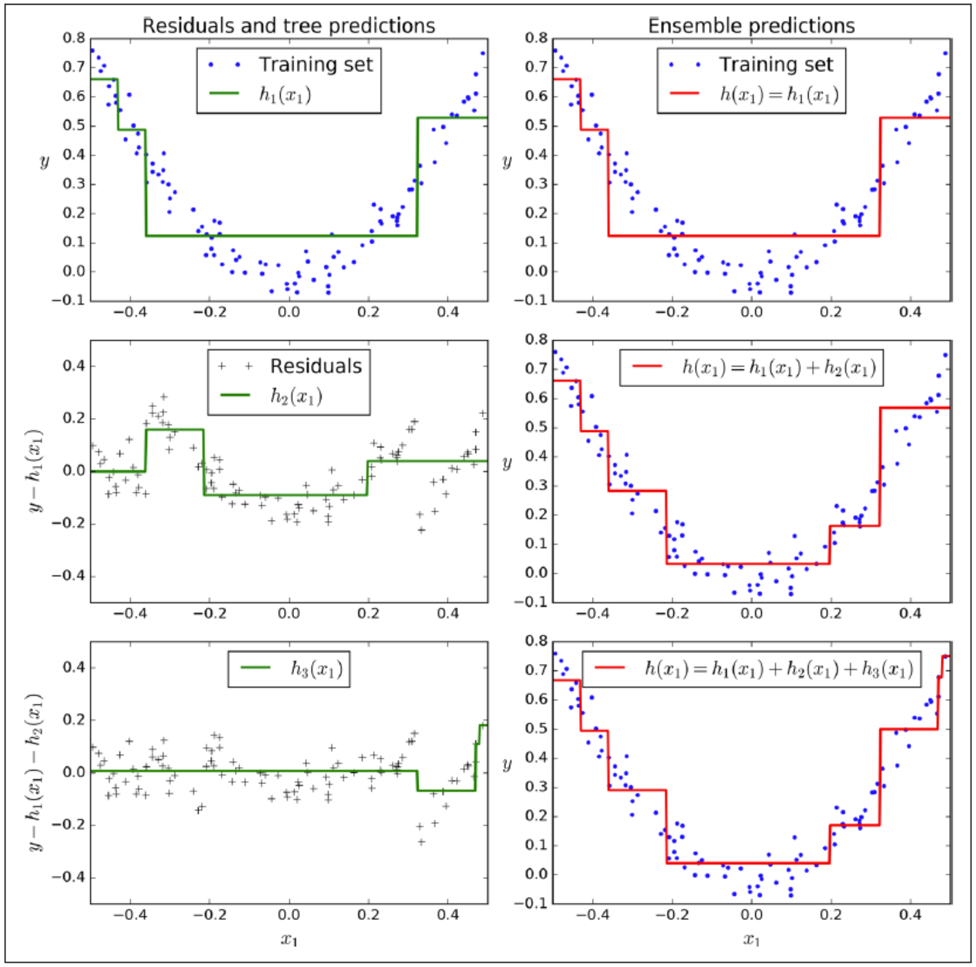
Gradient Boosting

- Try to fit a new predictor to the *residual errors* made by the previous predictor.

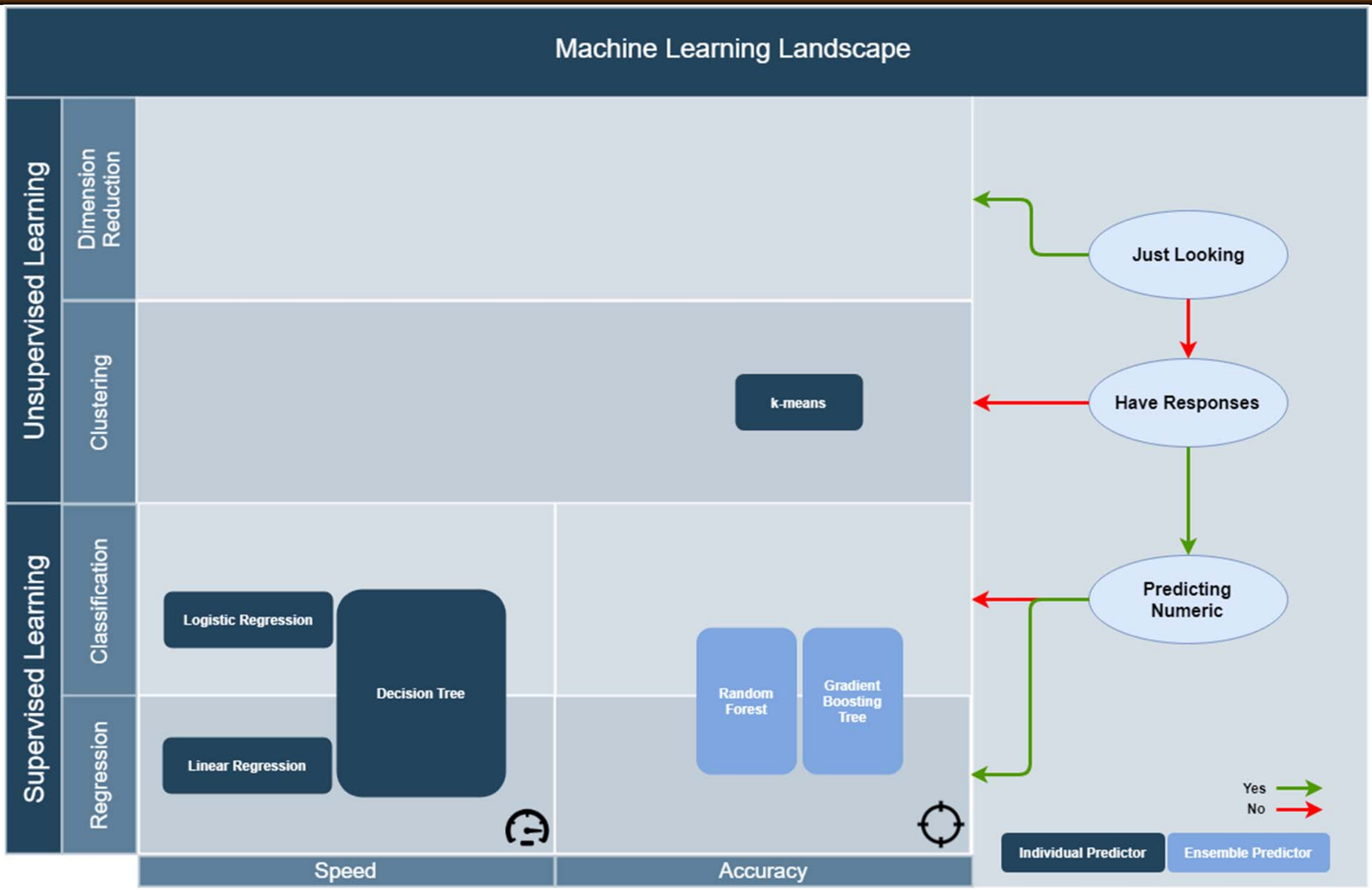
Best Performance

- Random Forests and Gradient Boosting Methods (implemented in the *xgBoost* library) have been winning most competitions on Kaggle recently on structured data.

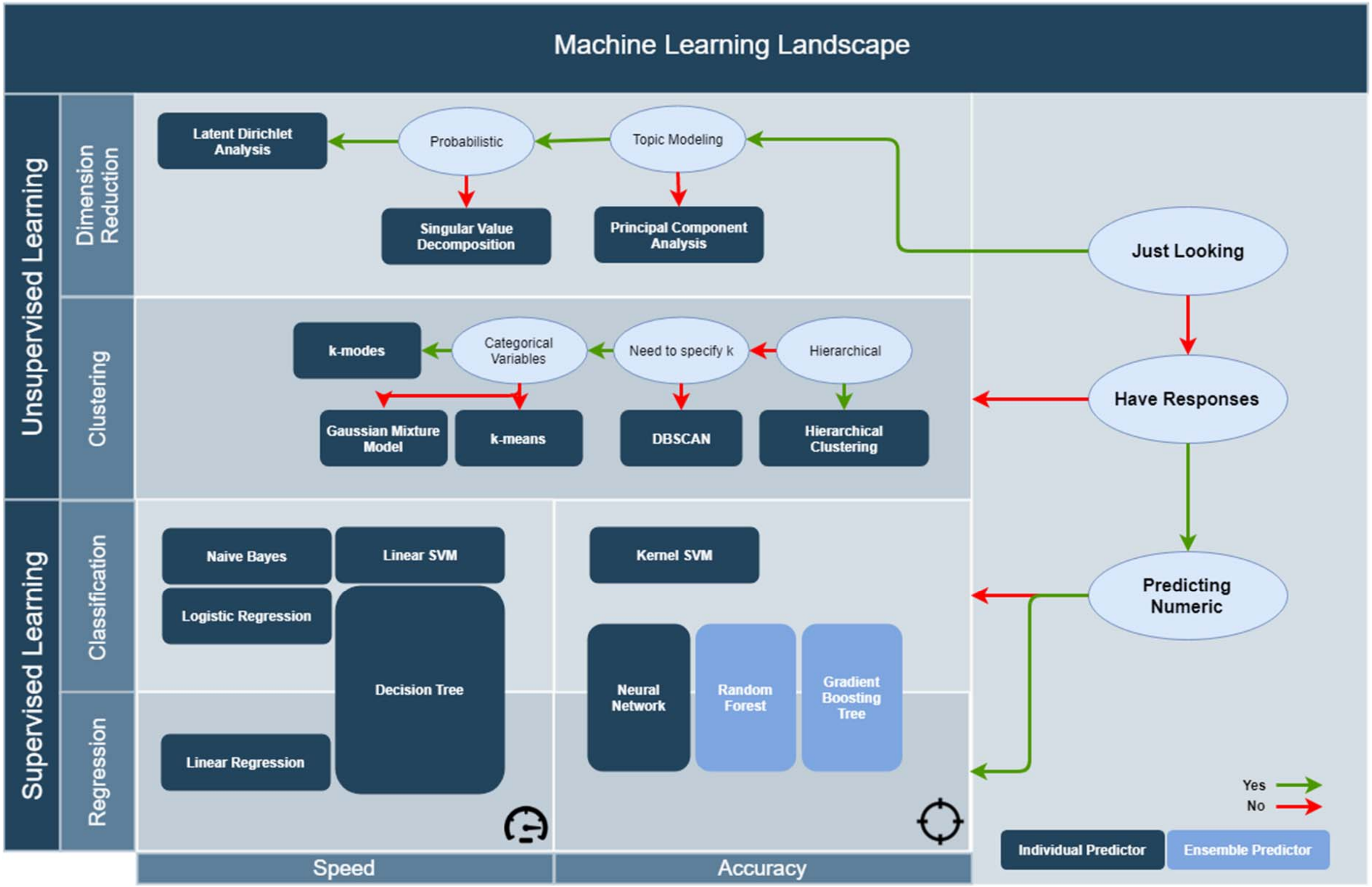
- Deep Learning (especially Convolutional Networks) is the clear winner for unstructured data problems (perception/speech/vision etc.)



ML Landscape



ML Landscape



Where to go from here?