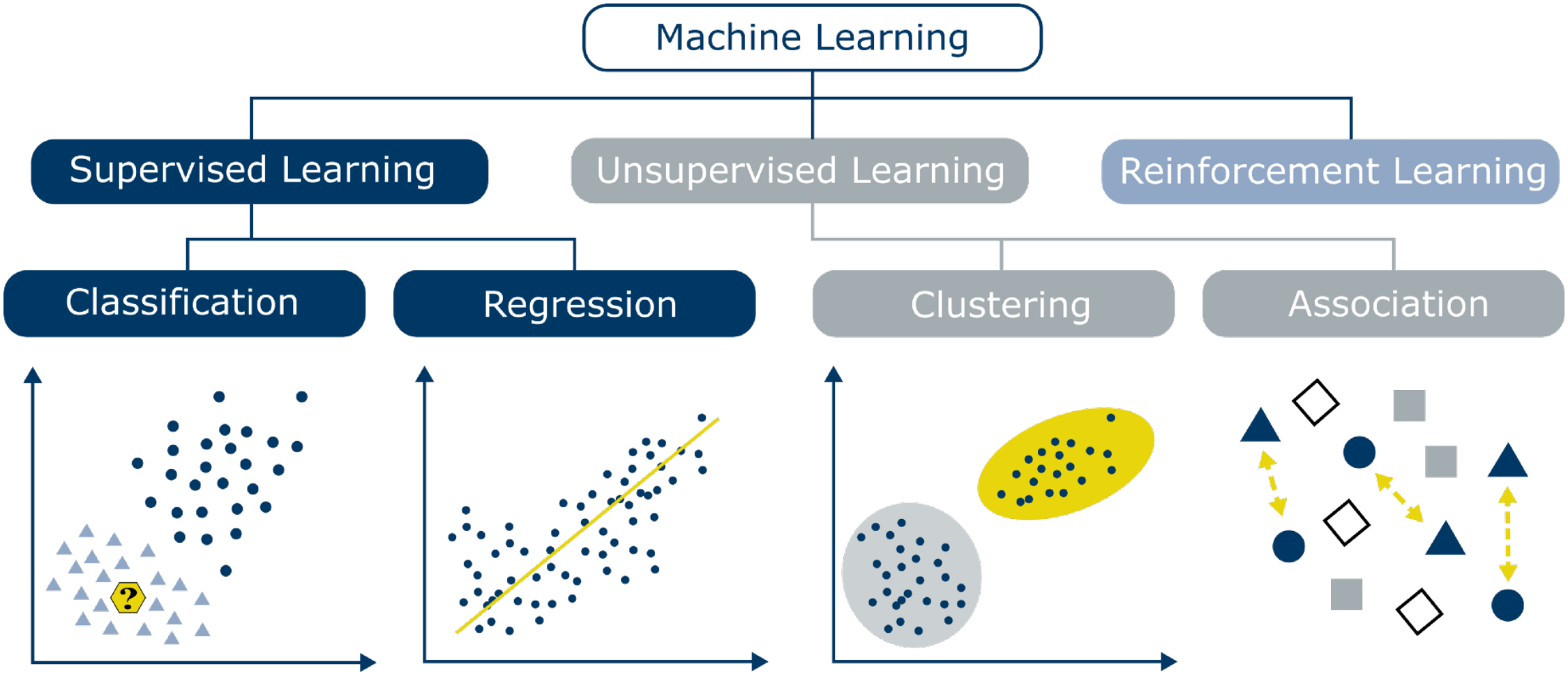
Machine Learning and its Applications

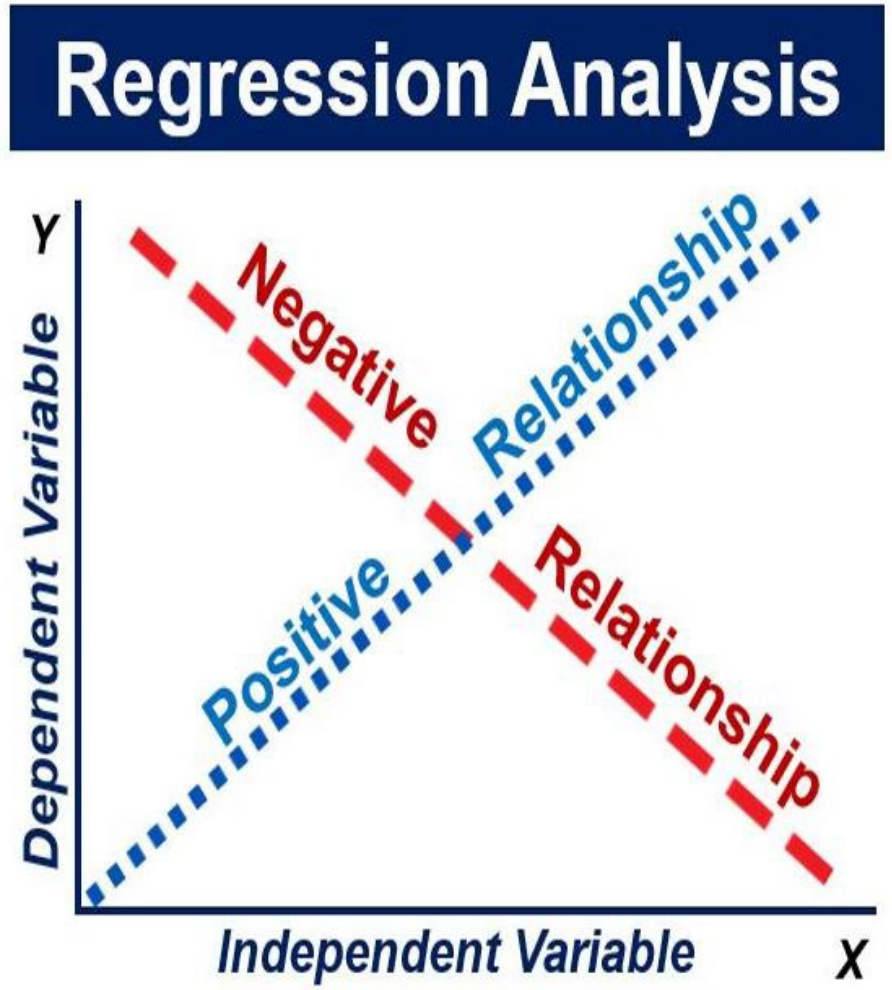
Supervised Learning: Regression

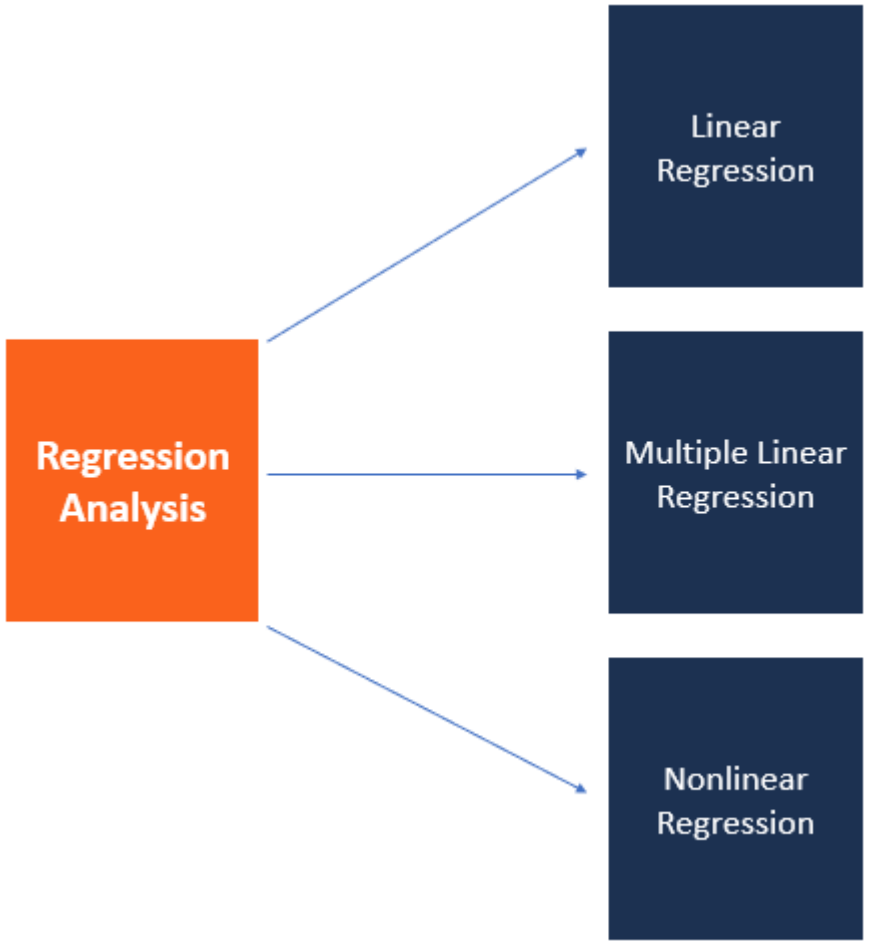
Urban Information Lab



Agenda for today

What is Regression?





Most commonly used

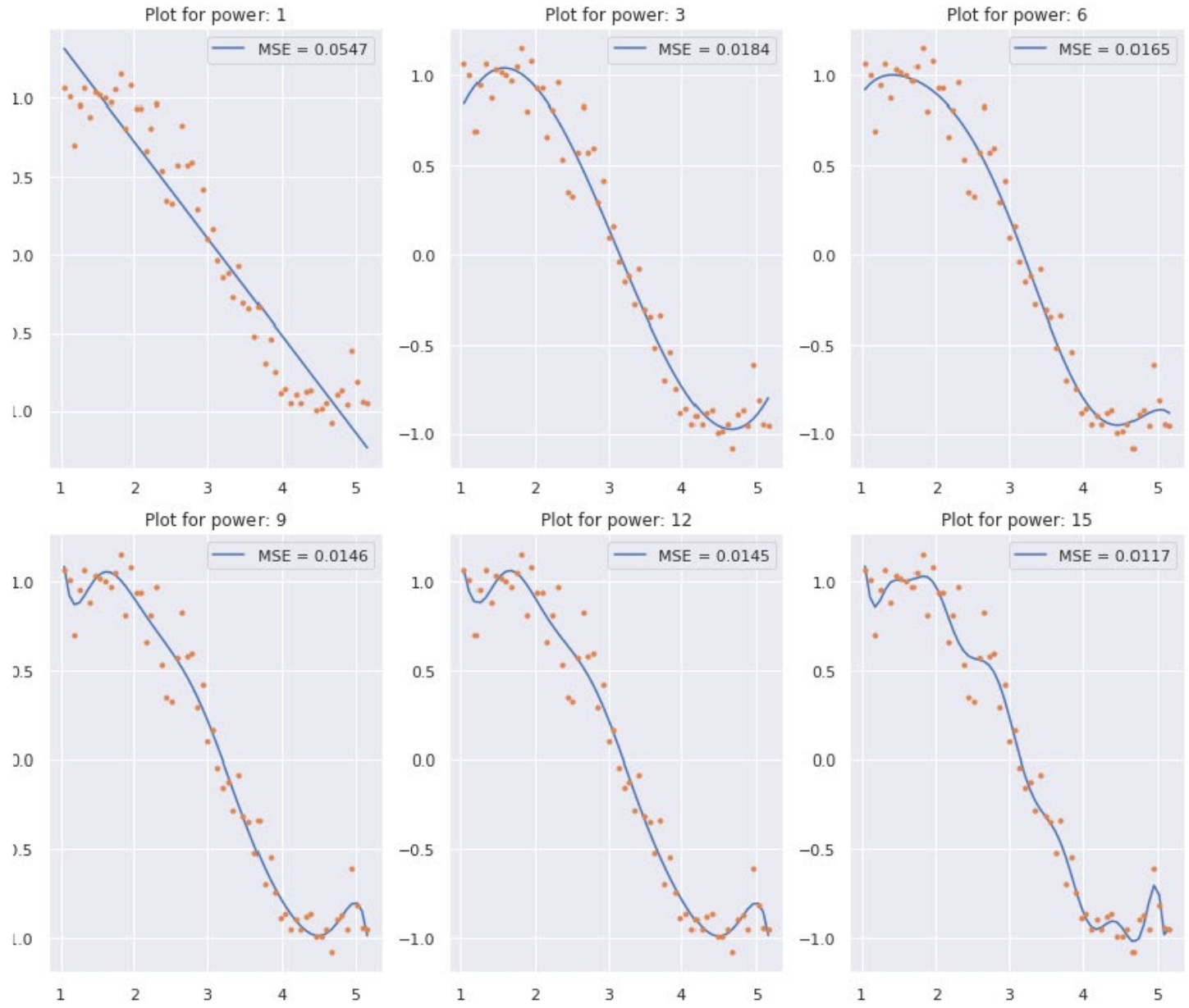
Regression

Used for predicting continuous Output

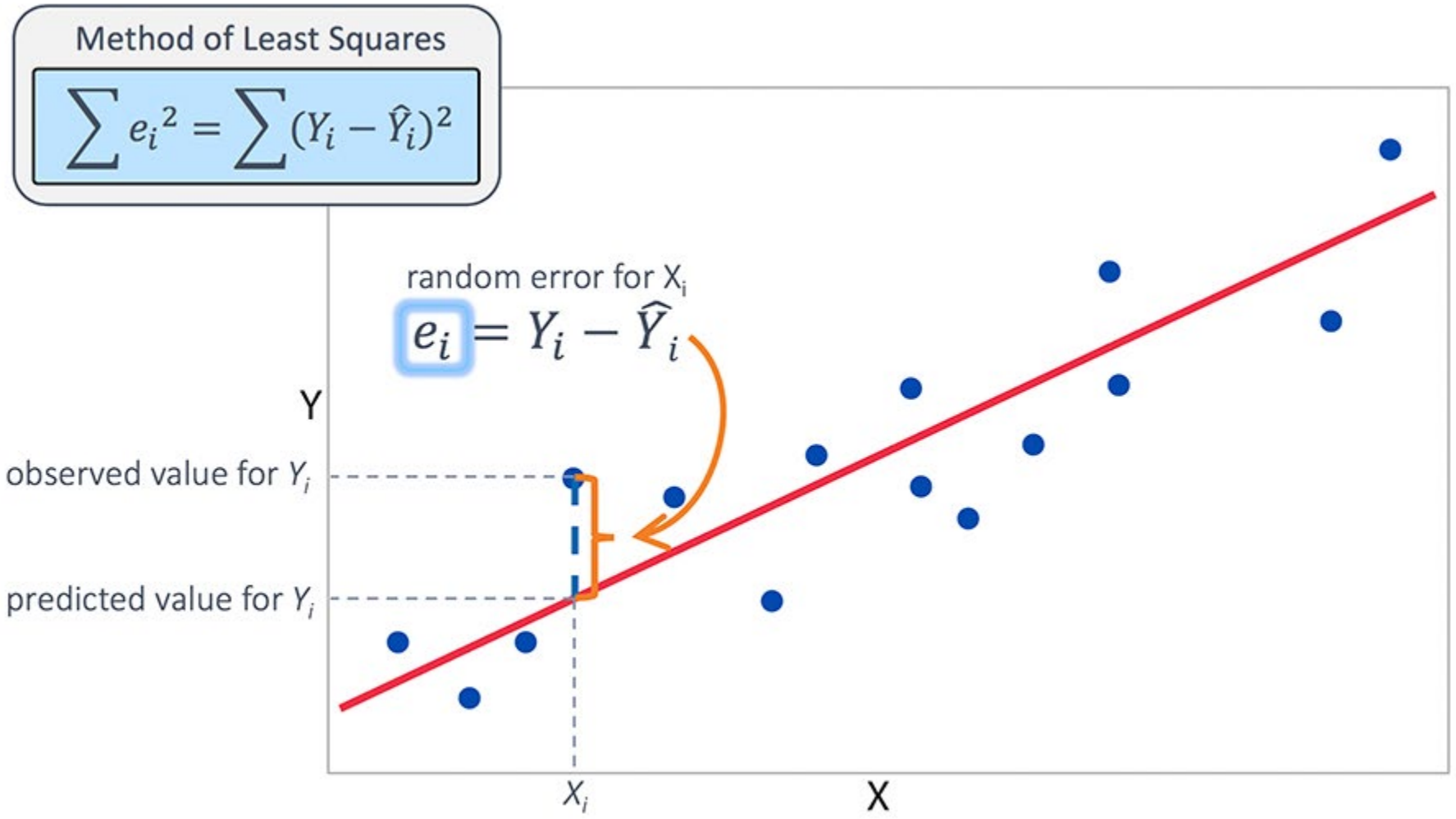
- Inputs $\mathbf{x}$; Output y: $\mathbf{x} \in R^d$
- Training data: $(\mathbf{x}^i, y^i), i = 1, 2, ..., N$.
- A Model: $\hat{y} = f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + ..$
- A **Loss Function** $L(\hat{y} - y) = ||y - \hat{y}||$.
- Optimization:
  - Analytic Solution
  - Convex Optimization
- Design good features [feature engineering] and feed them to a linear model.

- Consider one dimensional data.
- $f(x) = a_0 + a_1 x_1$
- $f(x) = a_0 + a_1 X + a_2 x^2$
- ..
- $f(x) = \sum_0^n a_n X^n$.
- All these are linear in model parameters a_i
- .
- .
- Which order polynomial?
- In the example, we try polynomials of order 1,..,15 on a noisy dataset

Linear Regression Examples



How to calculate error



How to evaluate your ‘regression’ model

Most commonly used

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Error Square = RSS (Residual Sum of Square)

How to evaluate your ‘regression’ model

Most commonly used

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Good Regressor ML = Having lower Error Score

How to evaluate your ‘regression’ model 2

- Model Evaluation: **R-square** It determines how much of the total variation in **y** (dependent variable) is explained by the variation in **X** (independent variable). Mathematically, it can be written as:

$$R_{square} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - y_{mean})^2}$$

- The value of R-square is always between 0 and 1, where 0 means that the model does not model explain any variability in the target variable (y) and 1 meaning it explains full variability in the target variable.

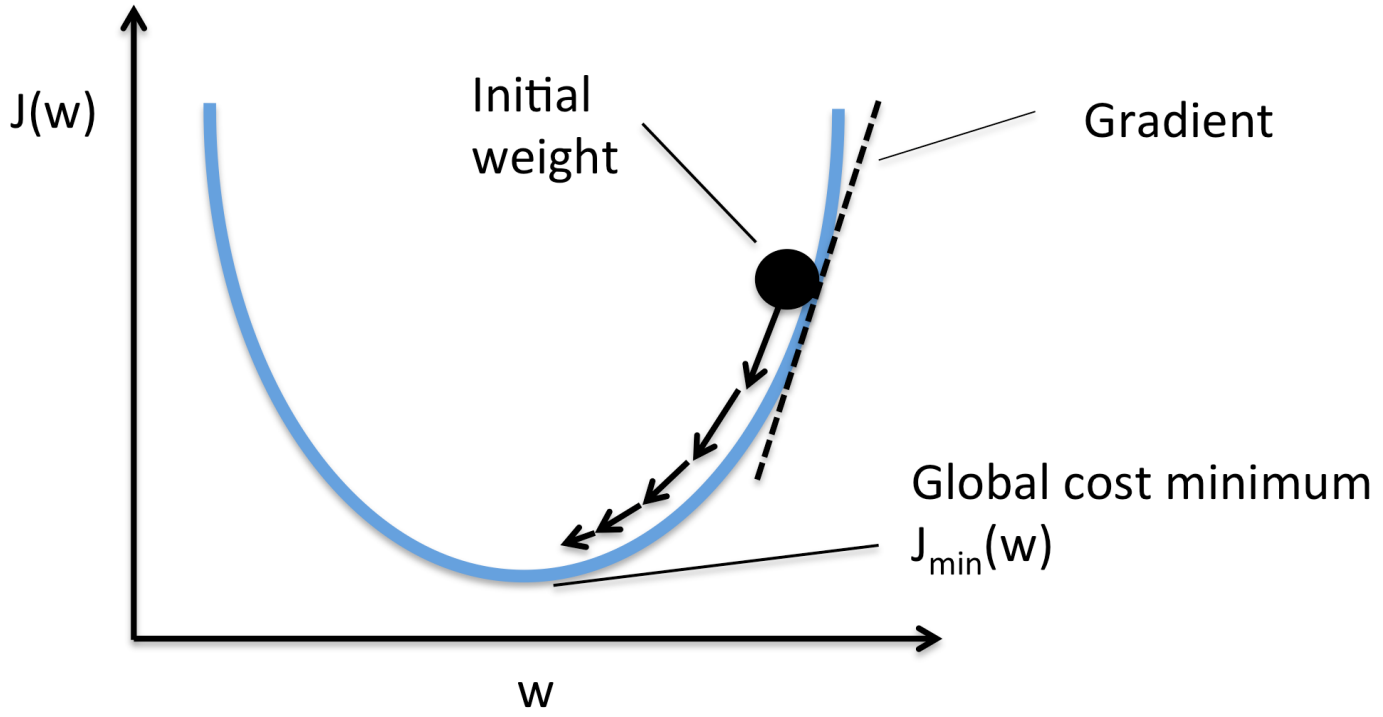
A way to reduce error: Gradient Descent

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

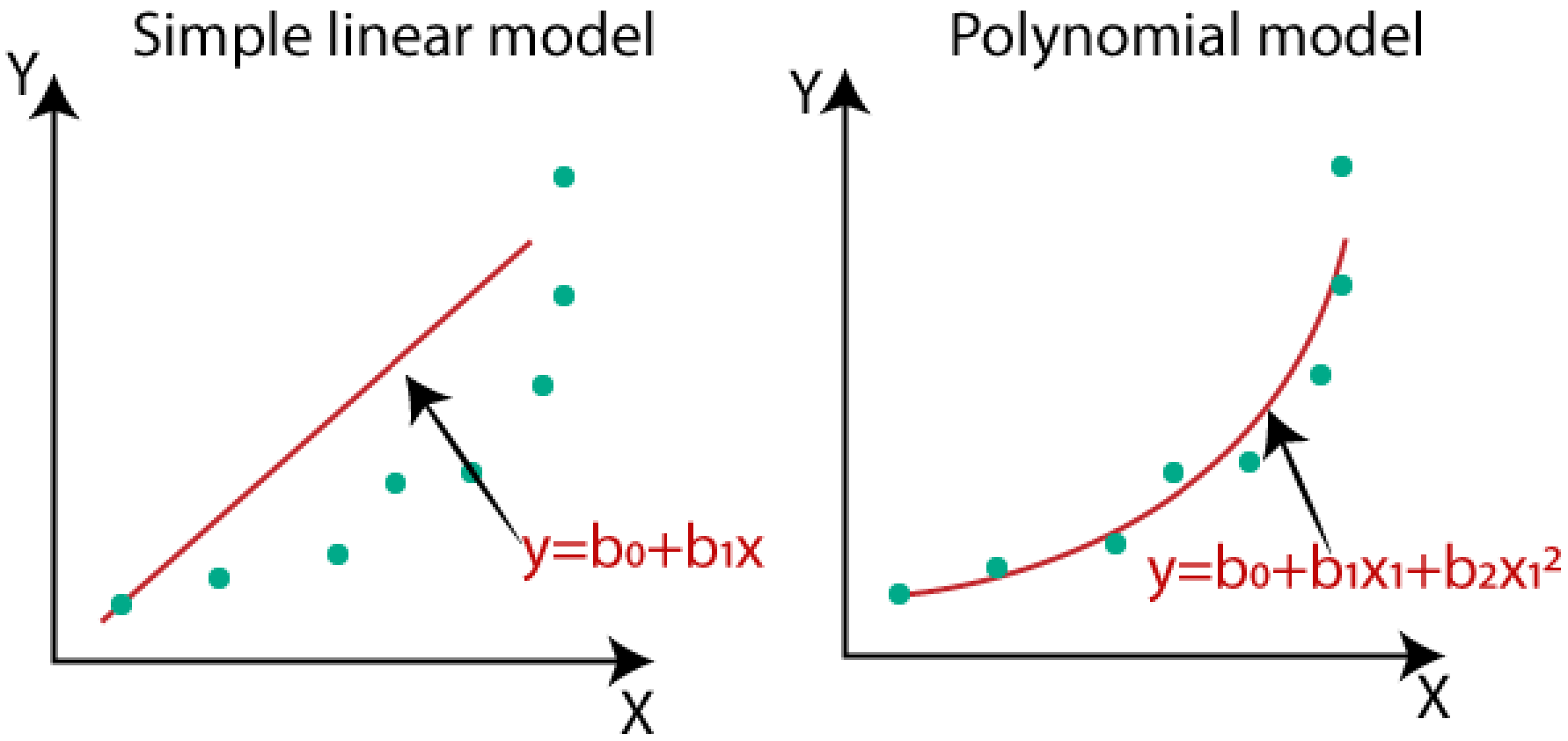
Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

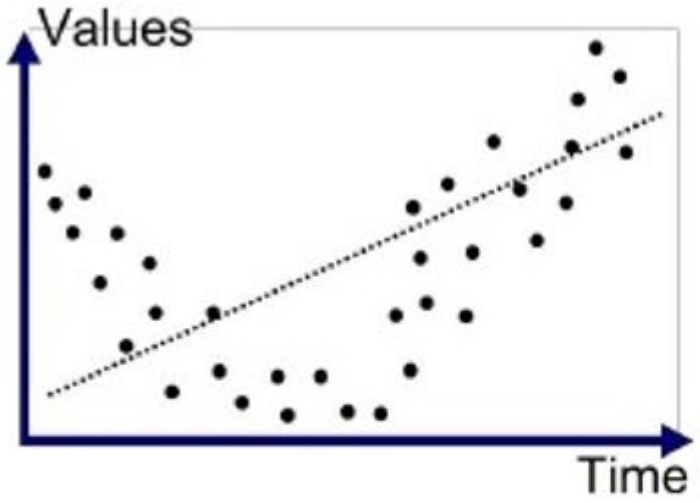


Polynomial Regression

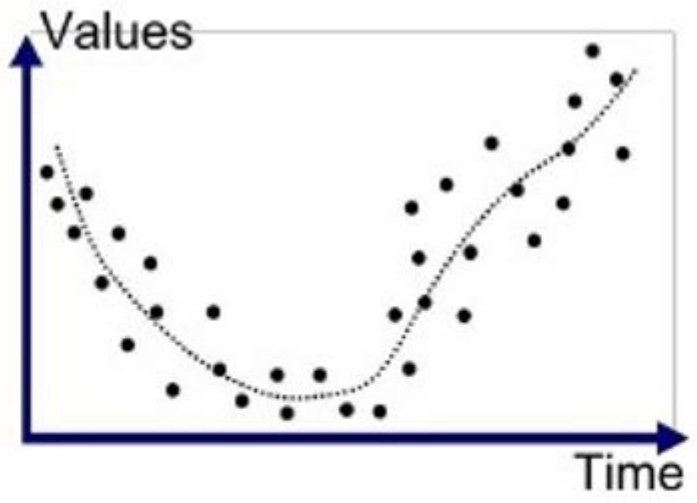


Polynomial models help you model complicated features

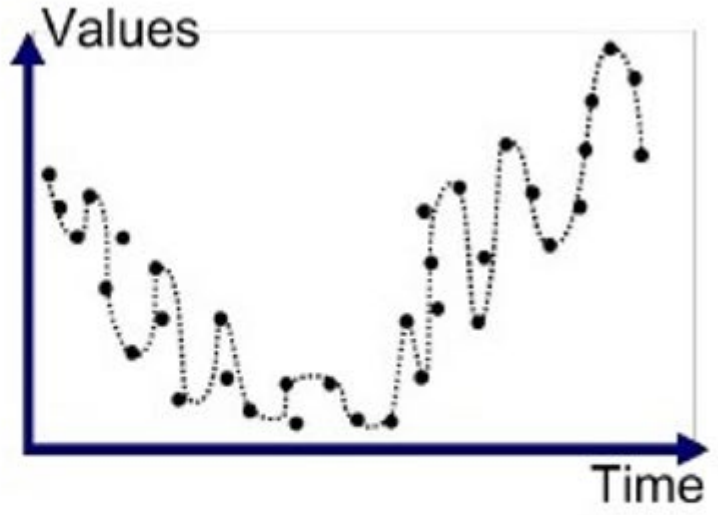
However, when polynomial degree gest higher the risk of overfitting increases



Underfitted



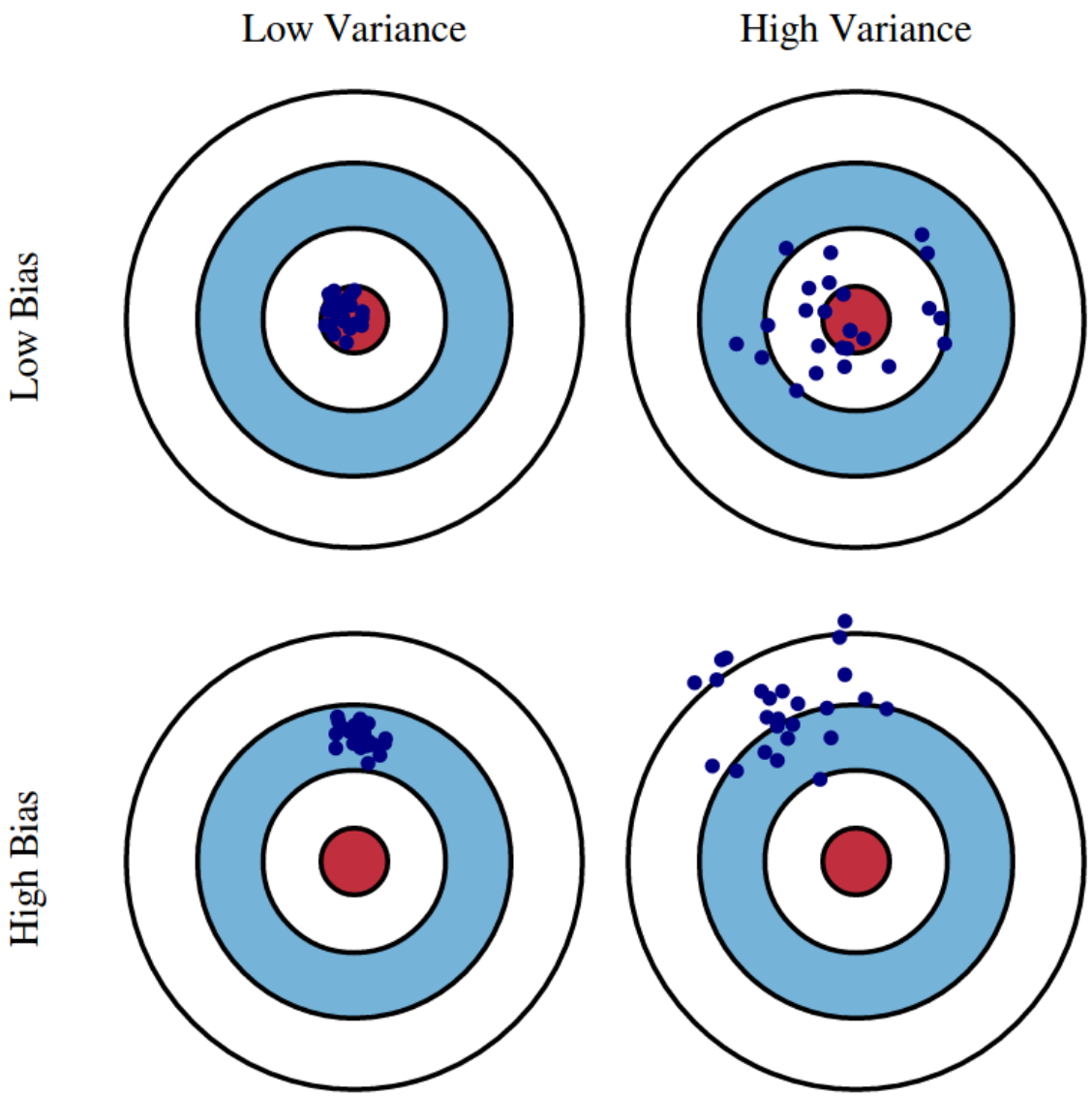
Good Fit/Robust

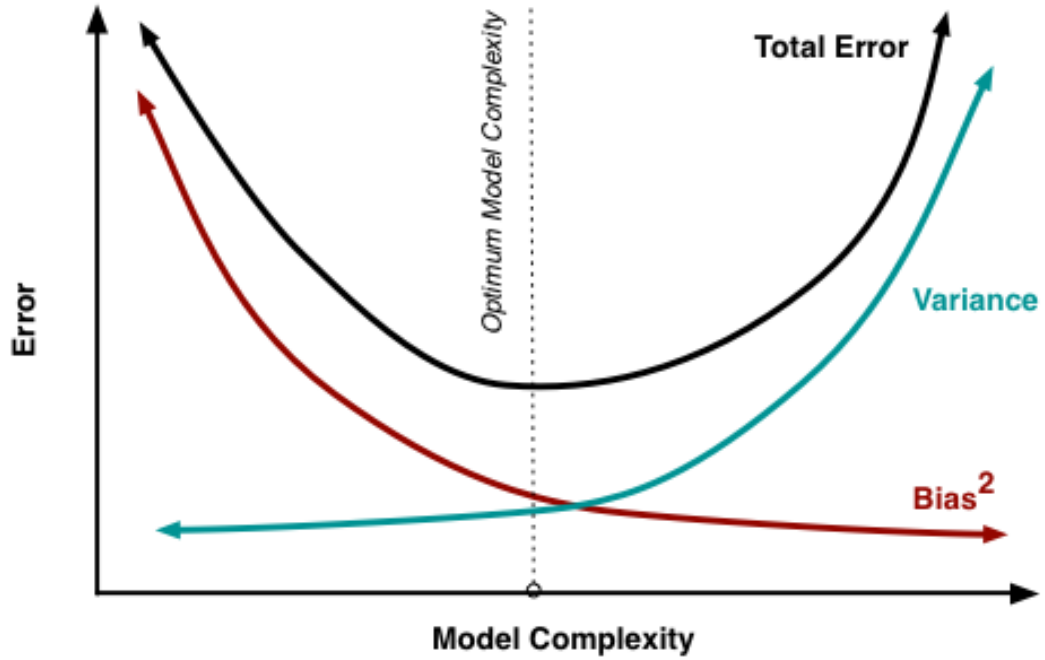


Overfitted

To set right parameters

Bias-Variance Trade off





To set right parameters

Regulation Linear Models

L1 Regulation (Lasso)

- Loss function

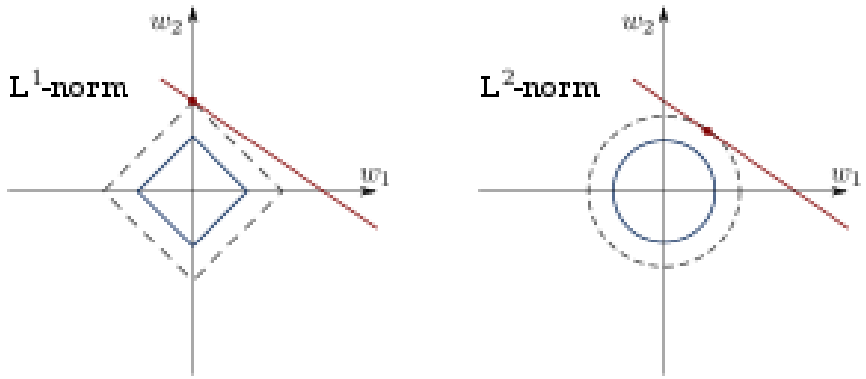
$$L(\mathbf{w}) = ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2 + \lambda||\mathbf{w}||_1$$

- No Analytic Solution!
- Equivalent to Laplace Prior!
- Choice of λ is critical!
- **Lasso** (least absolute shrinkage and selection operator)

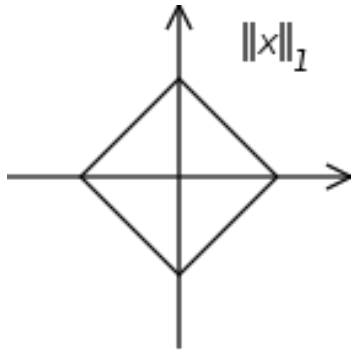
To set right parameters

L2 Regulation (Ridge)

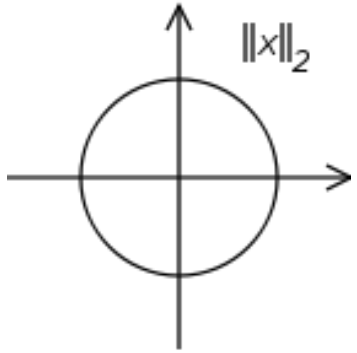
- In L1 regularization, we penalize the absolute value of the weights while in L2 regularization, we penalize the squared value of the weights.
- In L1 regularization, we can shrink the parameters to zero while in L2 regularization, we can shrink the parameters to as small as possible but not to zero. So, L1 can simply discard the useless features in the dataset and make it simple.



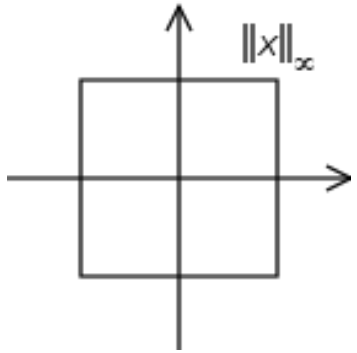
Of course there can be more norms



L1 Norm



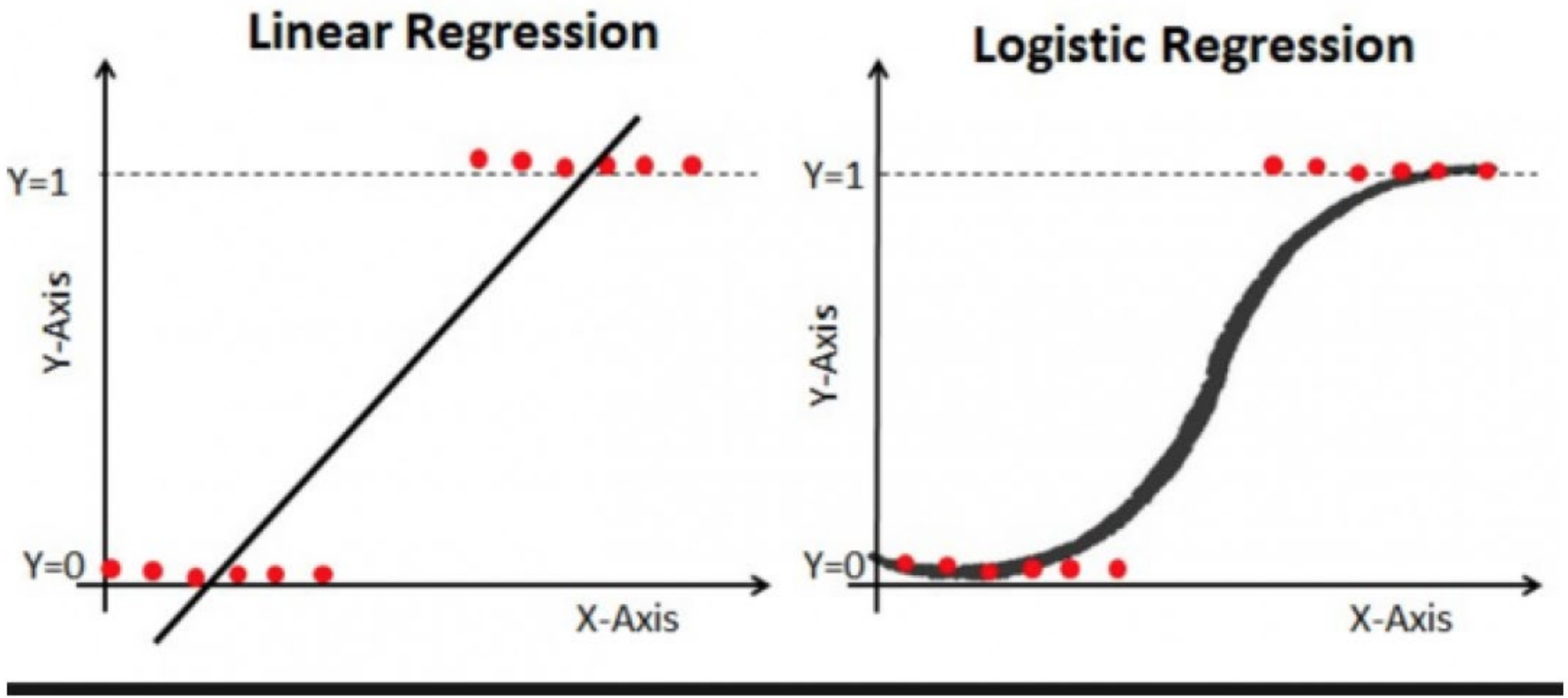
L2 Norm



Ln Norm

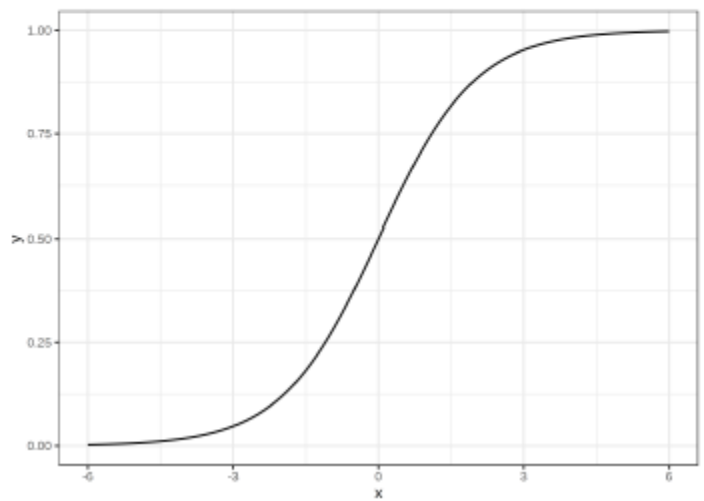
Logistic Regression

- Consider the **binary classification problem**
- y can take only two values, 0 and 1.
- $\mathbf{x}$ is a vector of real-valued features. $\mathbf{x} = [x_1.x_2, x_3, ...]^T$.
- We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given $\mathbf{x}$.
- However, it doesn't make sense for f(x) to possibly take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$.


Linear Regression
Y=1
Y-Axis
Y=0
X-Axis
Logistic Regression
Y=1
Y-Axis
Y=0
X-Axis

- A solution for classification is logistic regression. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.
- The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + \exp(-\eta)}$$



- In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \ldots + w_n x_n(i).$$

- For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function:

$$p(y^{(i)} = 1) = \frac{1}{1 + \exp\left(-\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \ldots + w_n x_n(i)\right)\right)}.$$

This forces the output to assume values between 0 and 1.

- We can generalize linear regression to the (binary) classification setting by making two changes.
- First, we replace the Gaussian distribution for y with a Bernoulli distribution,which is more appropriate for the case when the response is binary,$y \in \{0, 1\}$. That is, we use

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\mu(\mathbf{x})),$$

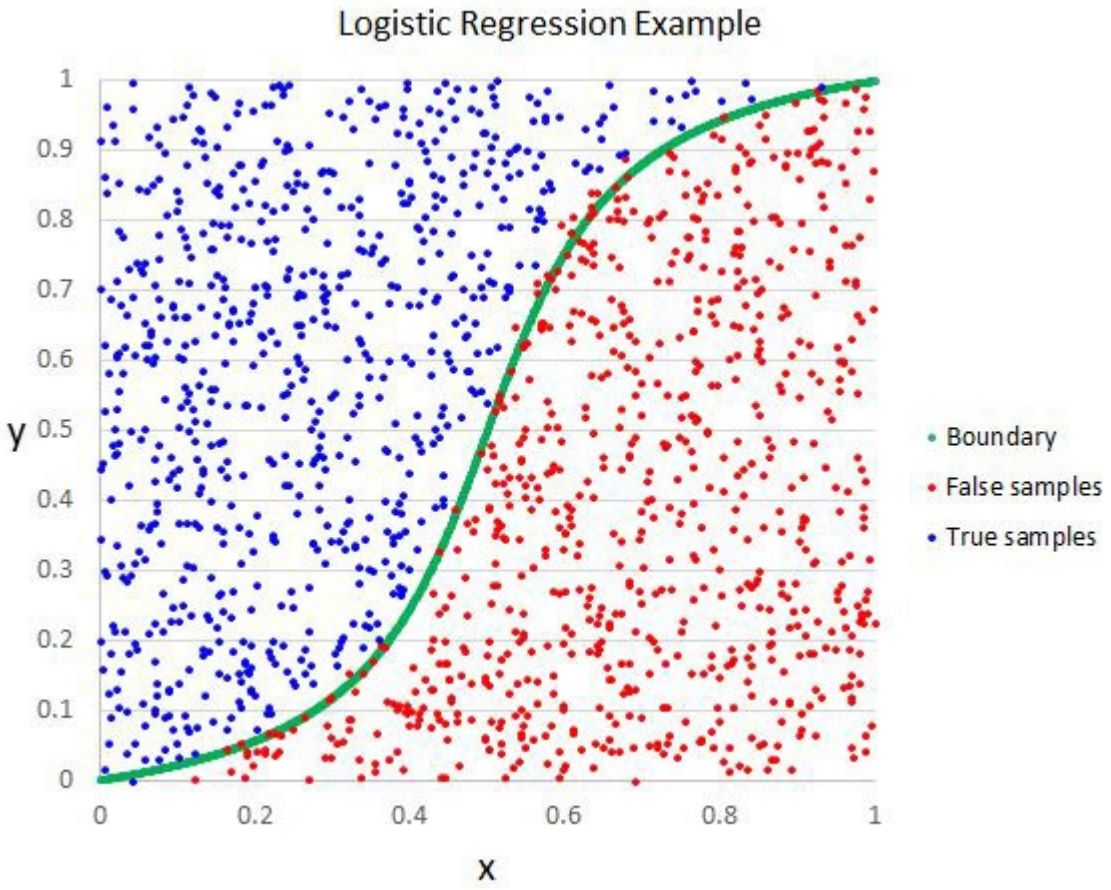
$$\mu(\mathbf{x}) = E(y|\mathbf{x}) = p(y = 1|\mathbf{x}).$$

- Second, we compute a linear combination of the inputs, as before, but then we pass this through a function that ensures $0 \leq (\mathbf{x}) \leq 1$ by defining (sigmoid/logistic/logit)

$$\mu(\mathbf{x}) = \text{sigm}(\mathbf{w}^T\mathbf{x}); \;\; \text{sigm}(\eta) = \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}.$$

- Thus,

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\text{sigm}(\mathbf{w}^T\mathbf{x})).$$



Compared to linear regression models Logistic regression uses sigmoid functions to identify optimal function, then use calculated value by its function as probability of true v. false

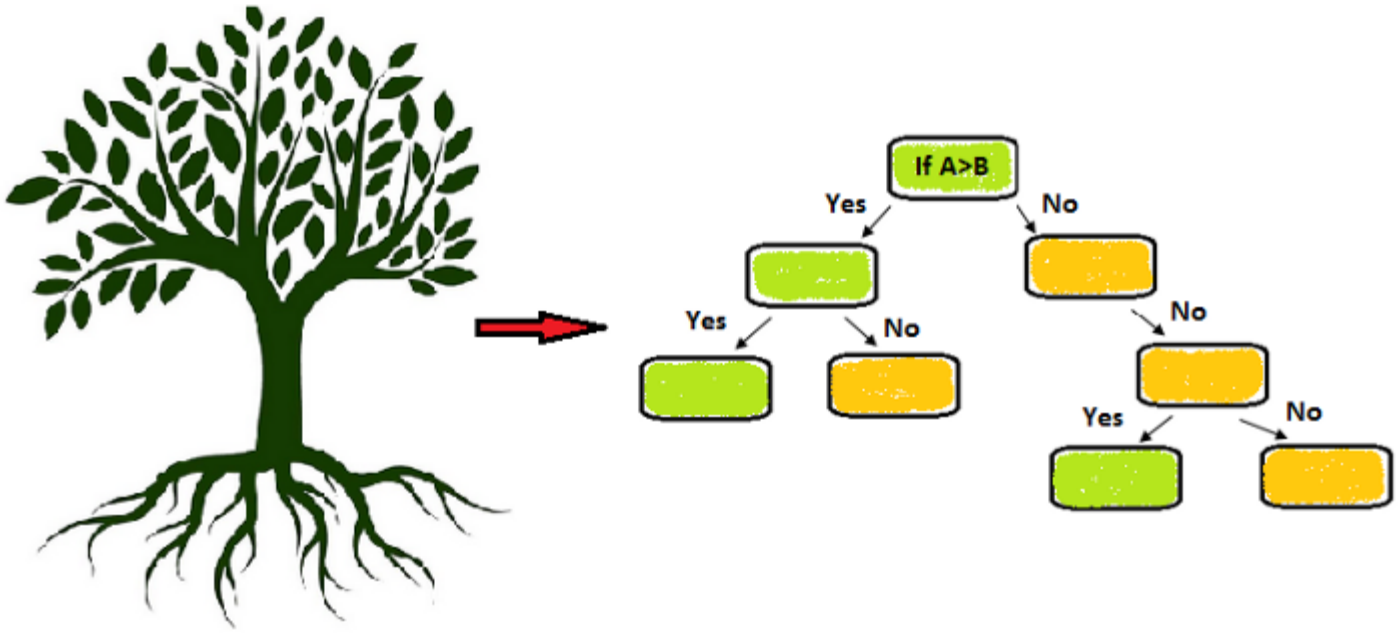
Thank You!

Machine Learning and its Applications

Supervised Learning: Decision Tree in Regression

Urban Information Lab

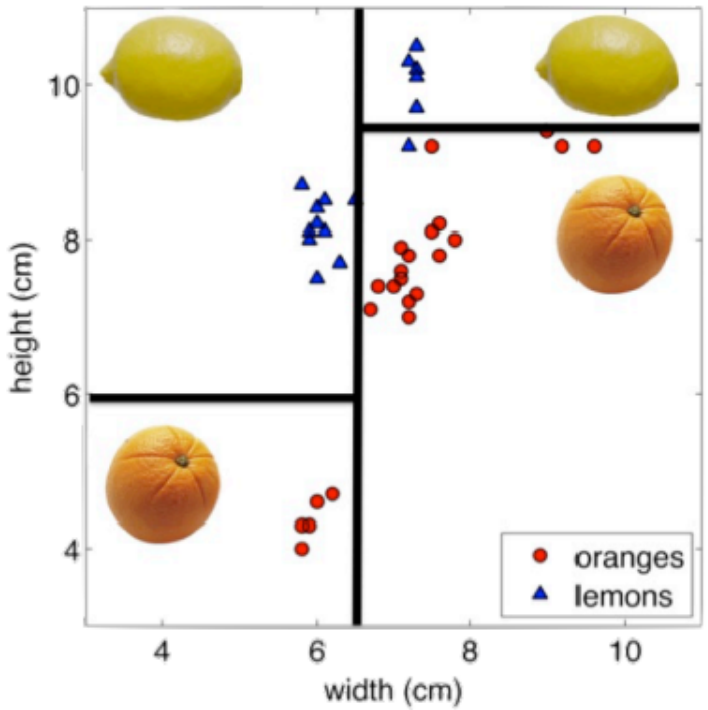
Introduction to Decision Tree

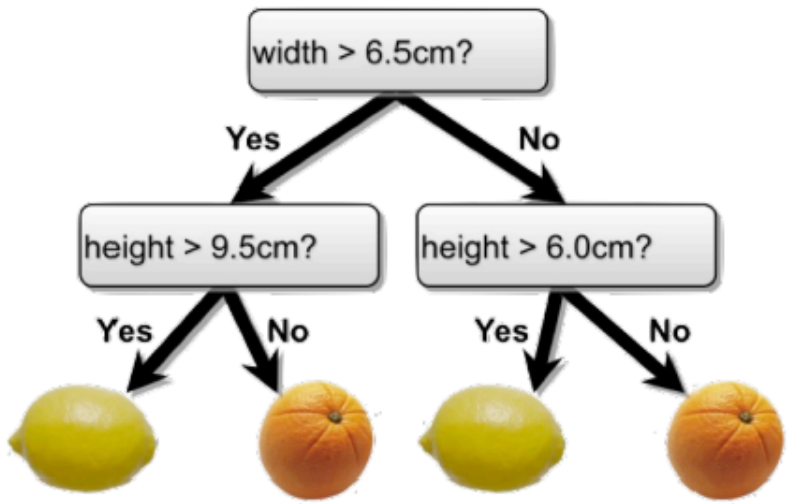


A ***decision tree model*** is an interpretable model in which the final output is based on a series of comparisons of the values of predictors against threshold values.

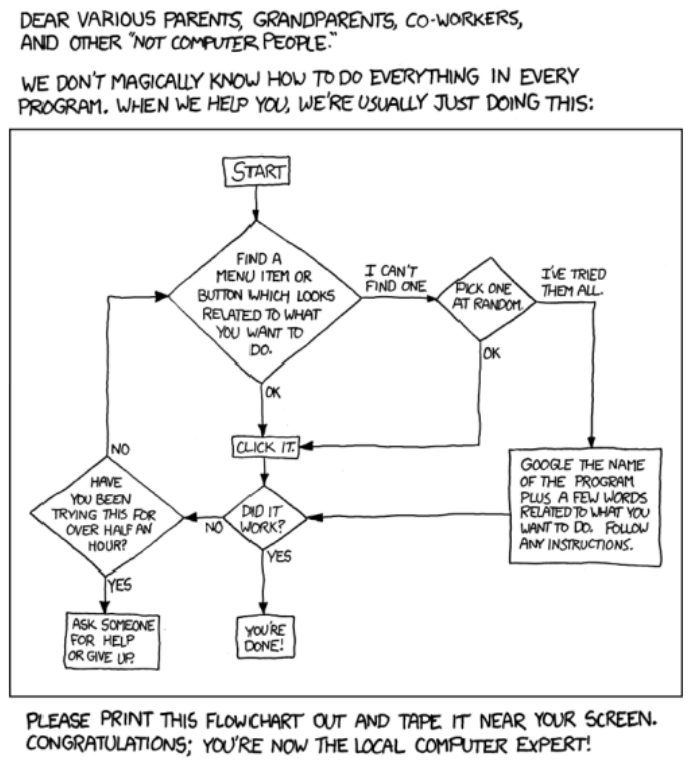
Graphically, decision trees can be represented by a flow chart.

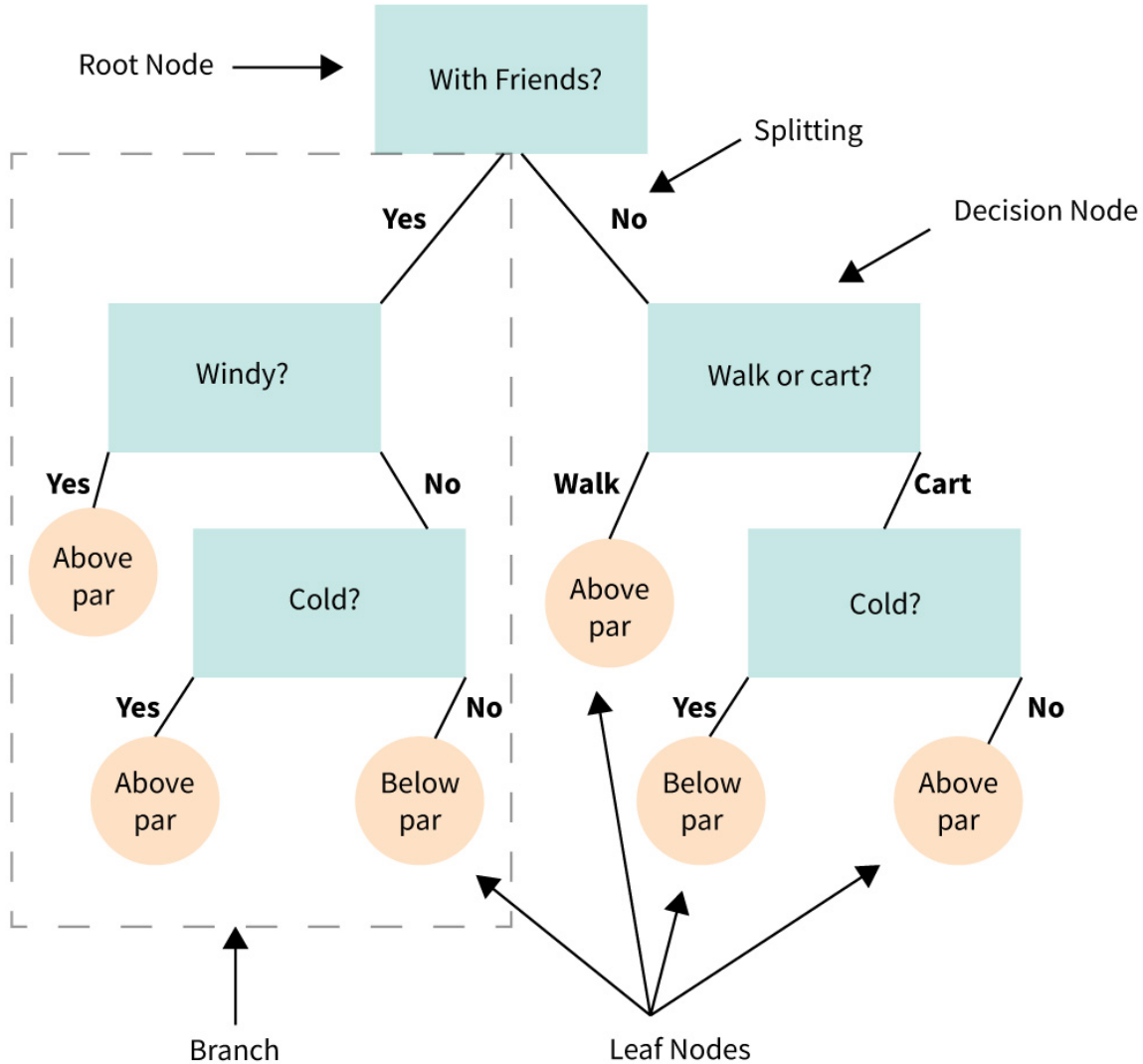
Geometrically, the model partitions the feature space wherein each region is assigned a response variable value based on the training points contained in the region.

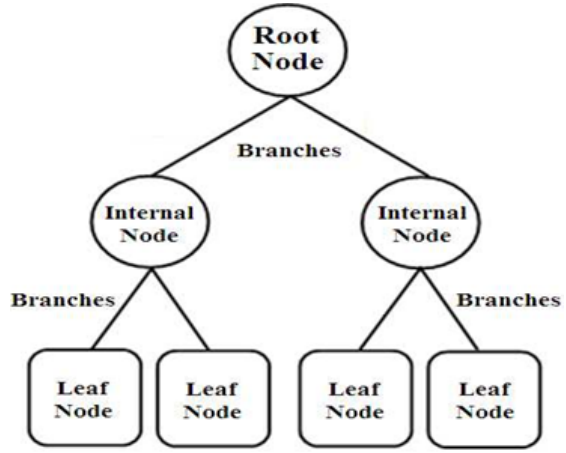




Decision Flowchart





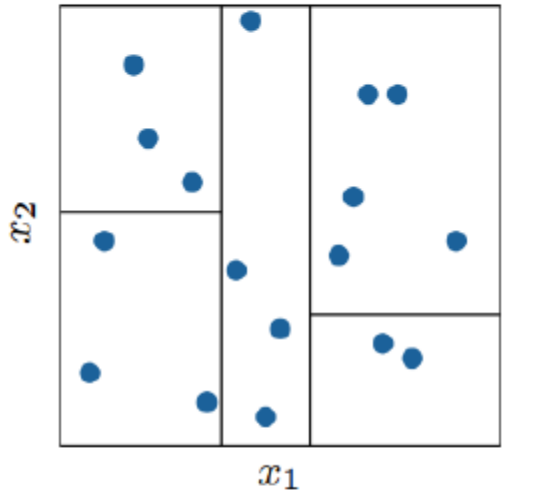


- **Root node**: no incoming edge, zero or more outgoing edges.
- **Internal node**: one incoming edge, two (or more) outgoing edges.
- **Leaf node**: each leaf node is an assigned a class label (if nodes are pure; otherwise majority vote).
- **Parent and child nodes**: If a node is split, we refer to that given node as the parent node, and the resulting nodes are called child nodes, respectively.

Works both in regression & classification

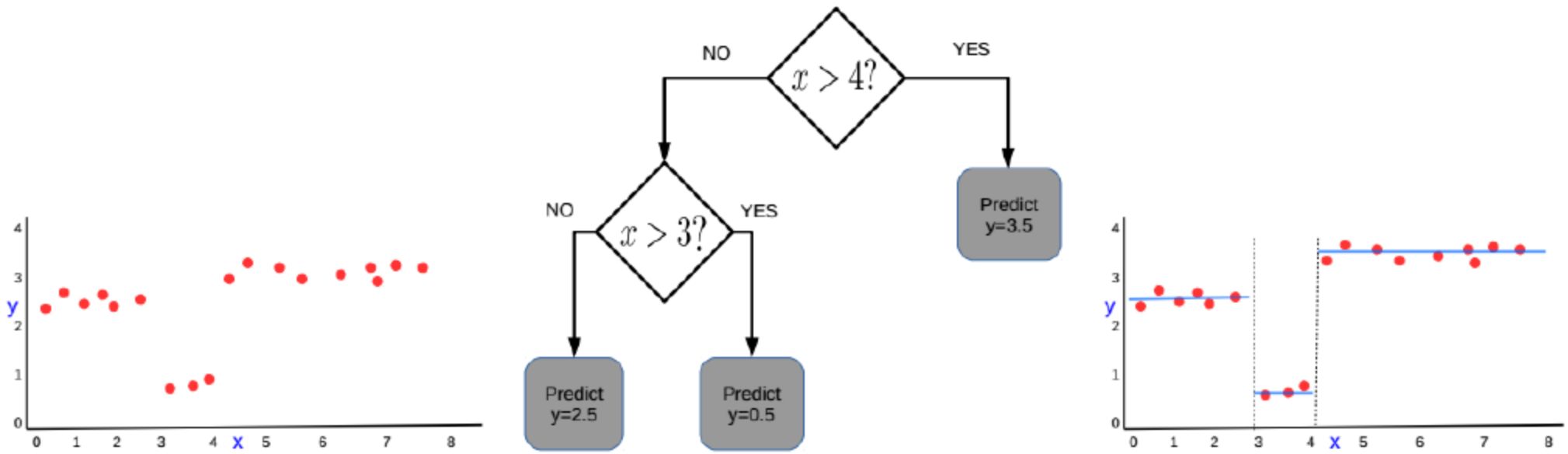
In both regression and classification settings we seek a function $\hat{y}(\mathbf{x})$ which maps the input $\mathbf{x}$ into a prediction.

One **flexible** way of designing this function is to partition the input space into disjoint regions and fit a simple model in each region.



- **Classification:** Majority vote within the region.
- **Regression:** Mean of training data within the region.

In regression

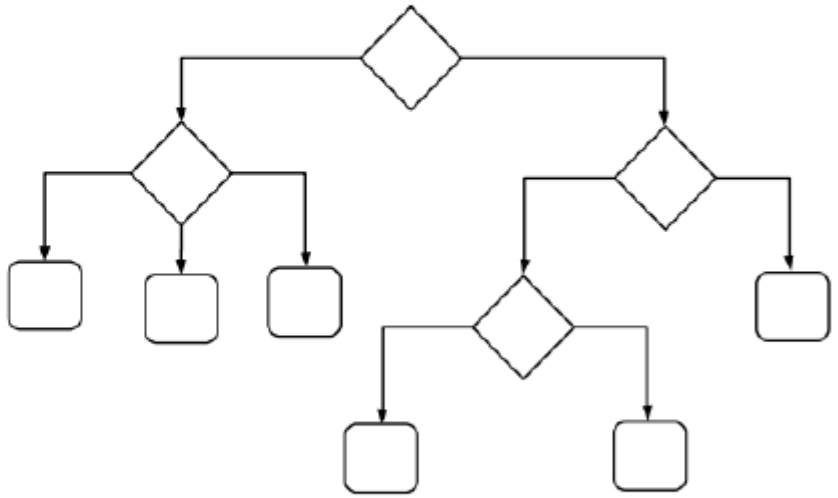


Here too, the DT partitions the training data into homogeneous regions (inputs with similar outputs)

https://www.cse.iitk.ac.in/users/piyush/courses/ml$_a$*utumn*18/*index.html*

Optimization

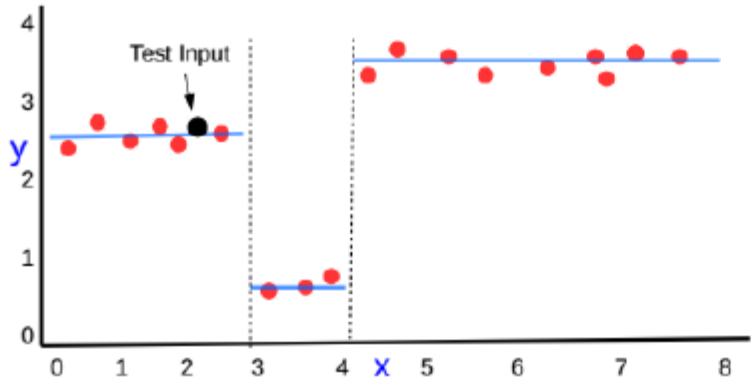
- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree

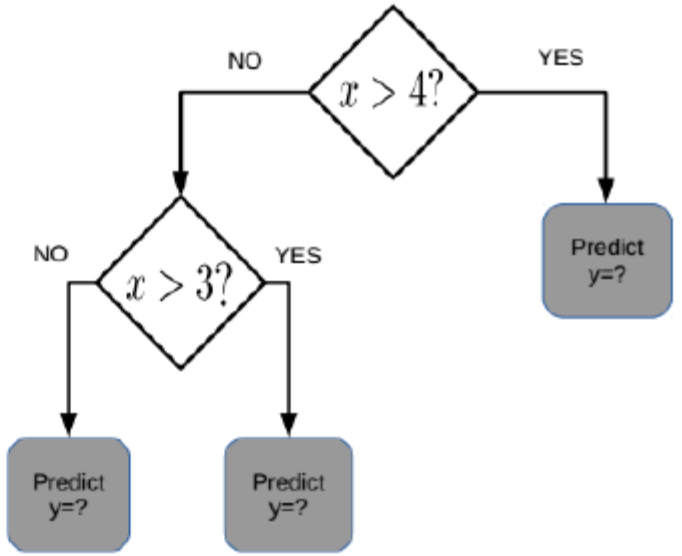


https://www.cse.iitk.ac.in/users/piyush/courses/ml_autumn18/index.html

Optimization

- What to do at each leaf node (the goal: make predictions)? Some options:
  - Make a constant prediction (majority/average) for every test input reaching that leaf node?
  - Use a nearest neighbors based prediction using all training inputs on that leaf node?



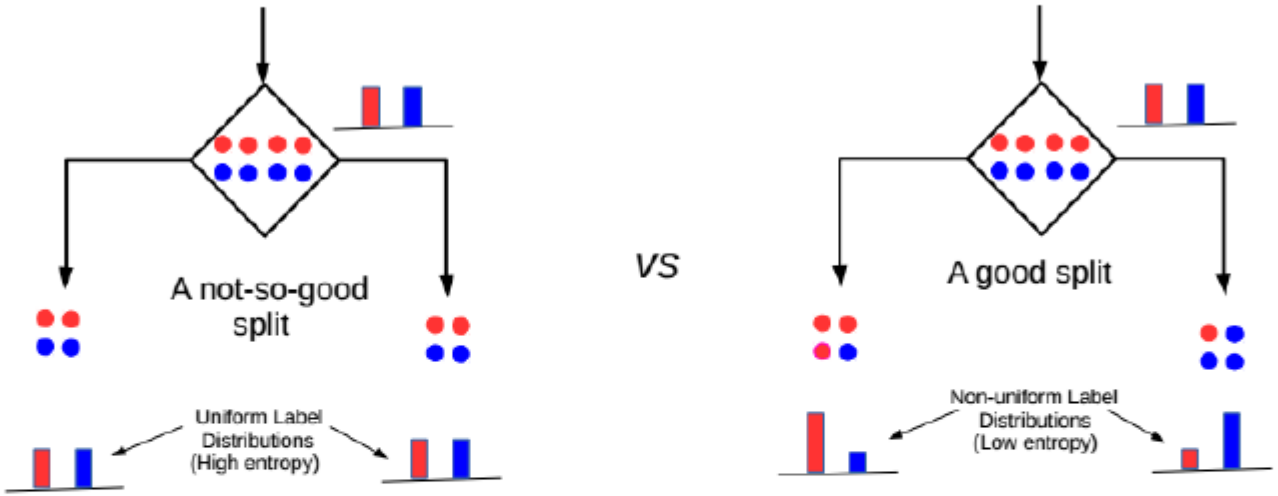


  - (Less common) Predict using an ML model learned using training inputs that belong to that leaf node?
- Constant prediction is the fastest at test time (and gives a piece-wise constant prediction rule)

https://www.cse.iitk.ac.in/users/piyush/courses/ml_autumn18/index.html

Optimization

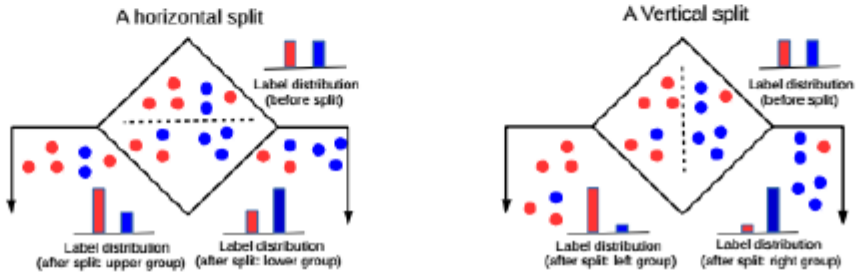
- How to split at each internal node (the goal: split the training data received at that node)?
- No matter how we split, the goal should be to have splits that result in groups as "pure" as poss



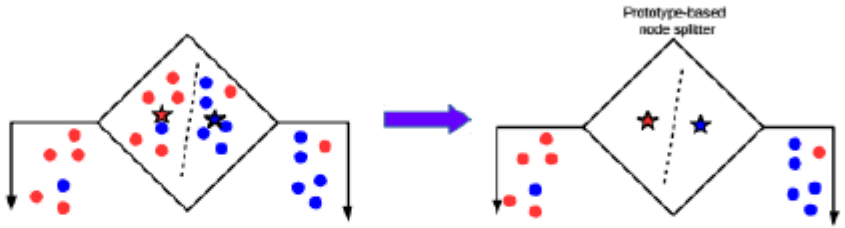
- For classification problems, entropy of the label distribution is a measure of purity
  - Low entropy ⇒ high purity (less uniform label distribution)
  - Splits that give the largest reduction (before split vs after split) in entropy are preferred (this reduction is also known as "information gain")
- For regression, entropy doesn't make sense (outputs are real-valued). Typically variance is used.

Optimization

- Note that splitting at internal node itself is like a classification problem
  - Data received at the internal node has to be routed along its outgoing branches
- Some common techniques for splitting an internal node
  - Splitting by testing a single feature (simplest/fastest; used in ID3, C4.5 DT algos). For example:



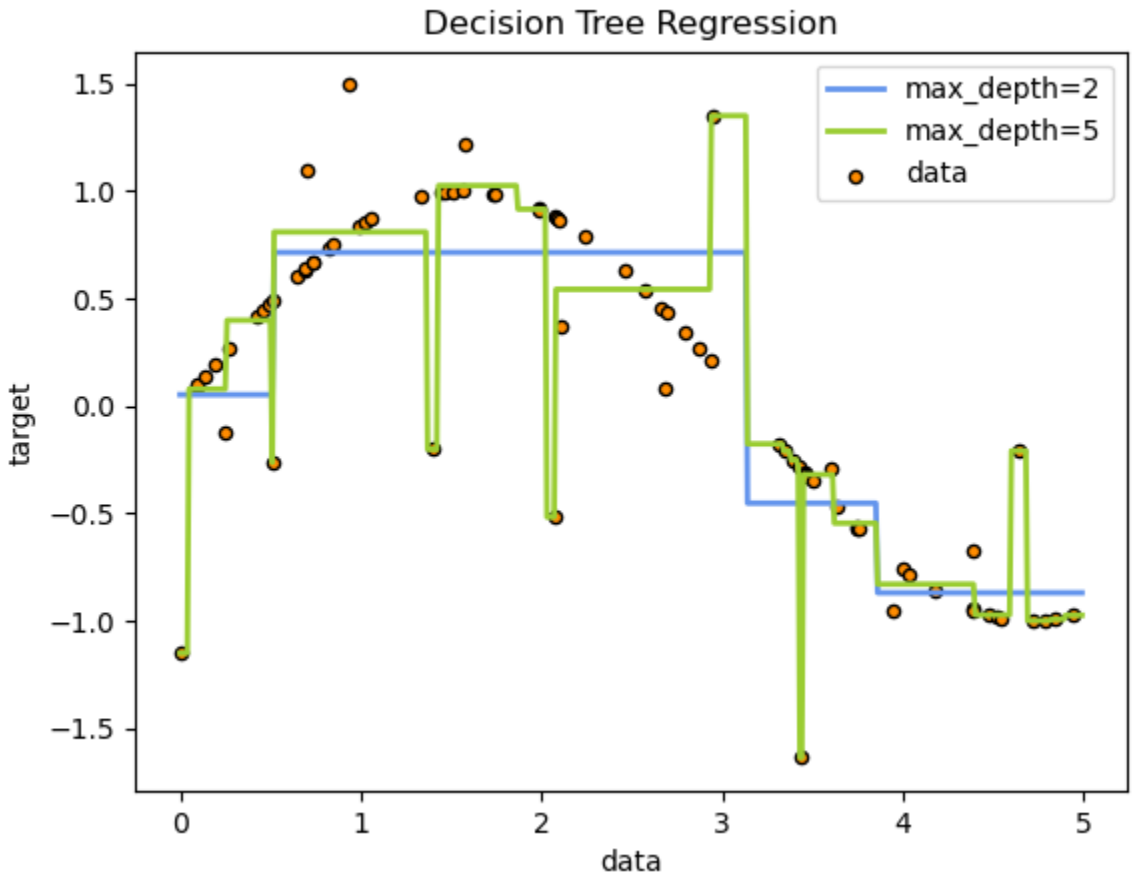
  - Splitting using a classifier learned using data on that node. For example, prototype based classifier



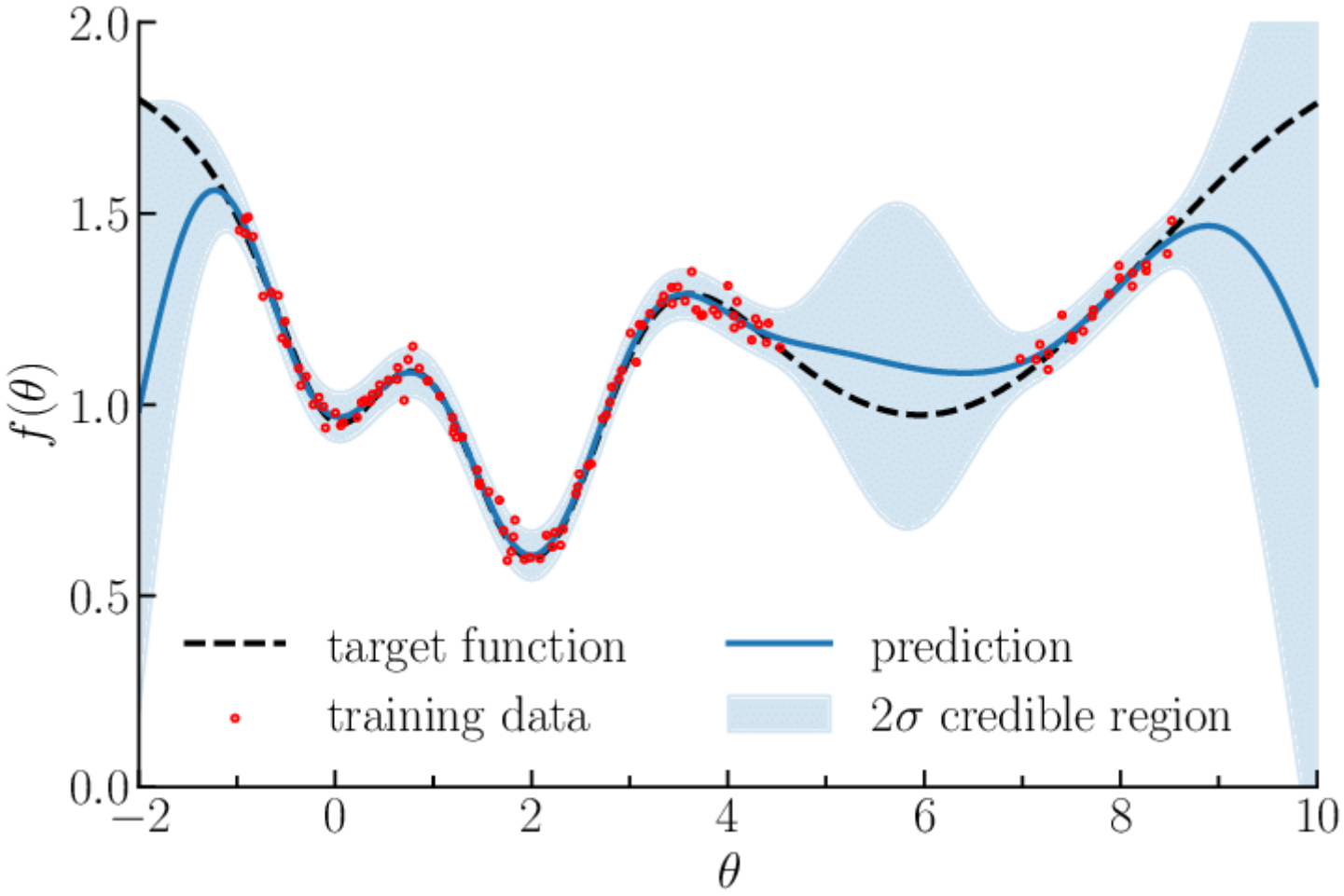
- The same splitting rule will be applied to route a test input that reaches this internal node

https://www.cse.iitk.ac.in/users/piyush/courses/ml_autumn18/index.html

Optimization Examples

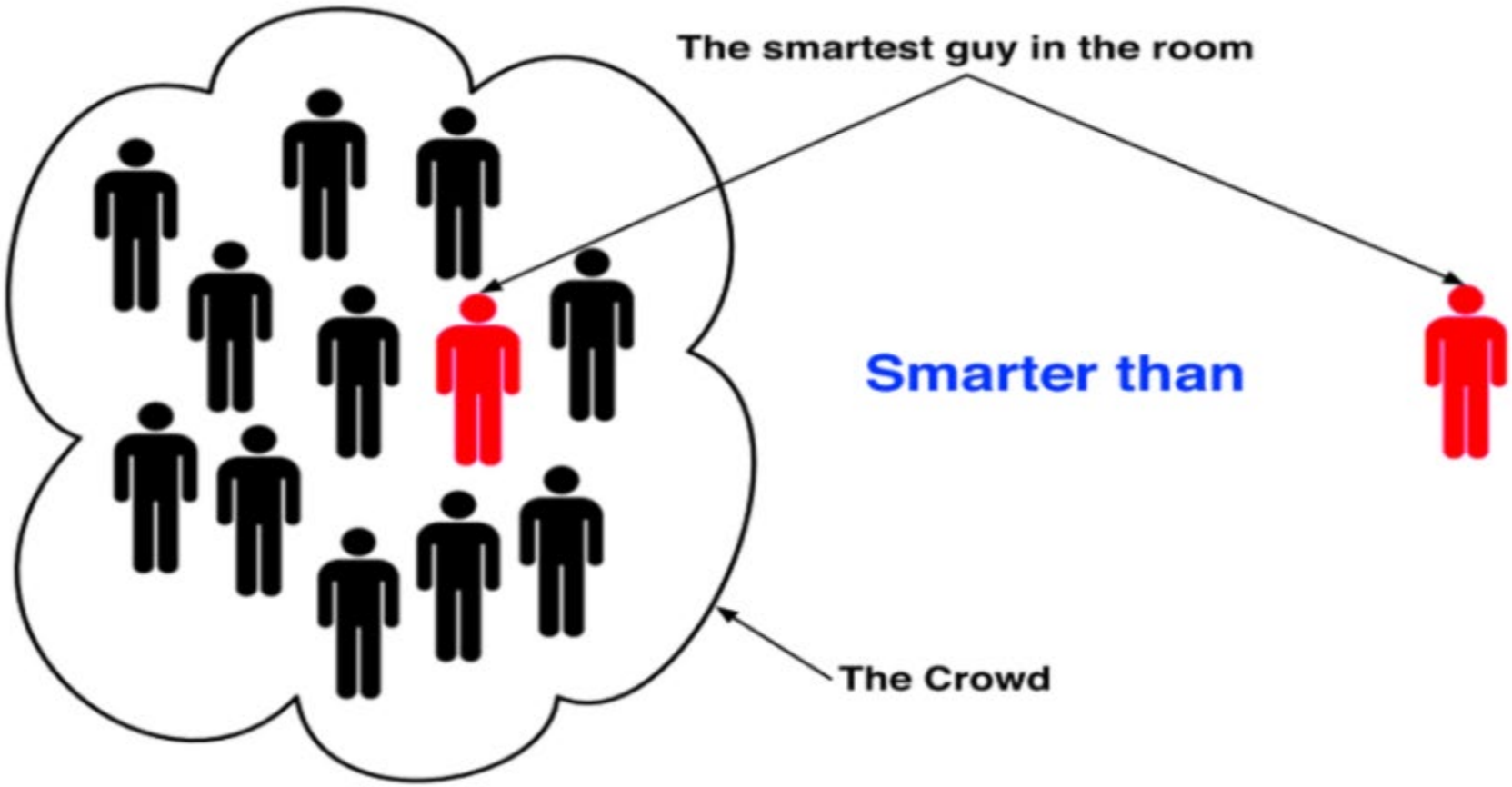


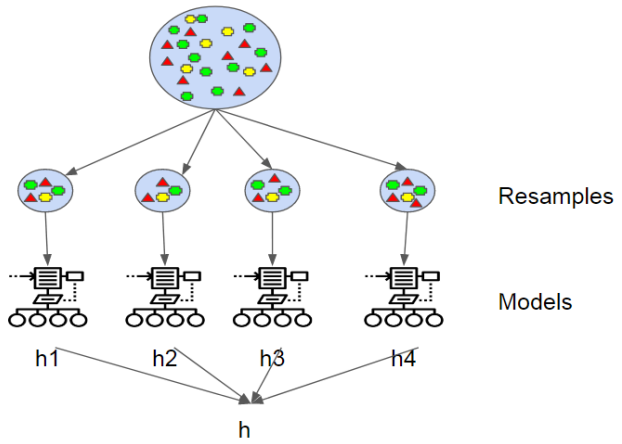
Optimization (Number of Estimators)



Gaussian Process

Ensemble Learning?





Decision Tree

Ensemble Learning

Voting

Bagging

Boosting

Staking

Regression

Classification

Ensemble Learning will be re-mentioned in classification

Thank You!