Fusemachines
AI School

Popular ML Models

Learning Outcomes

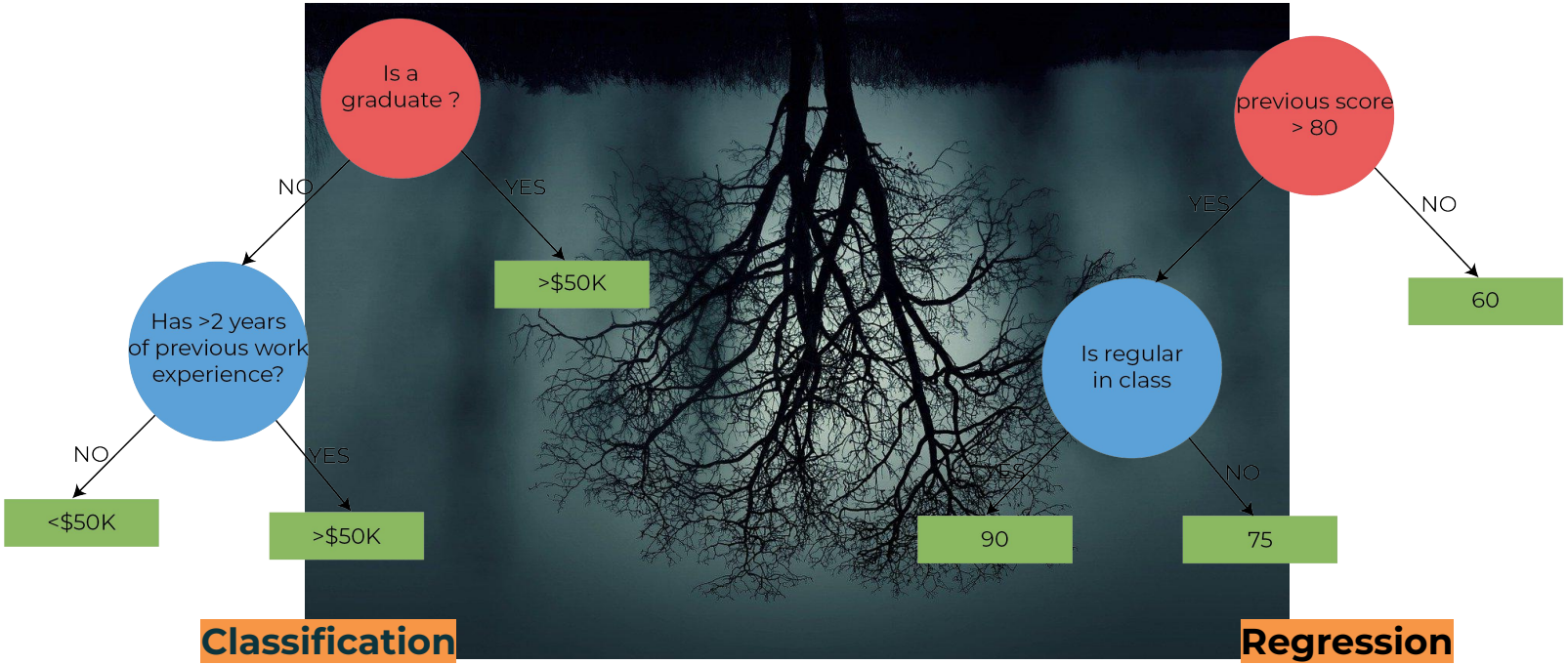
1. Understand the key concepts of some popular ML models
    - Decision Trees
    - Regularization of Trees
    - Error Decomposition
    - Ensemble of Trees: Random Forest and Gradient Boosted Trees
    - Nearest Neighbours (KNN) (Next Class)
    - Intuition behind Support Vector Machines (SVM) (TA)

Fusemachines
AI School

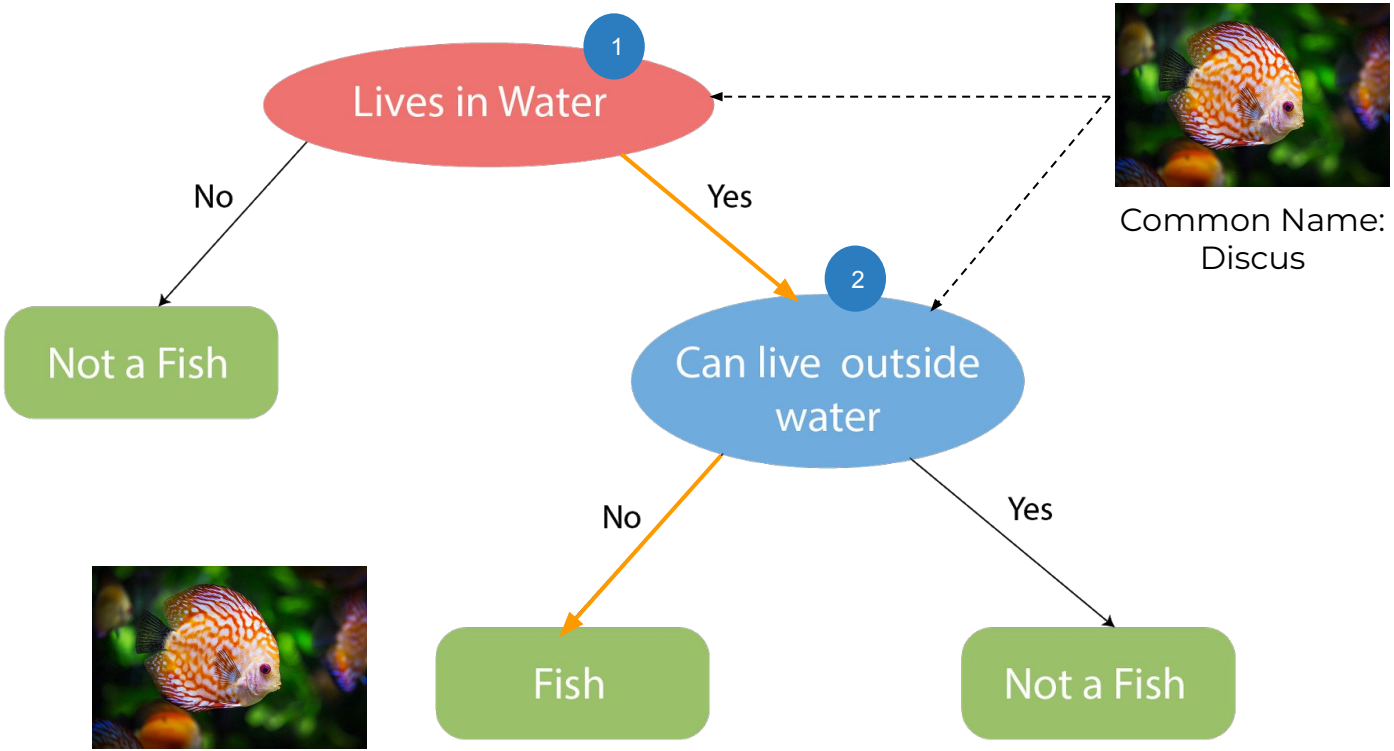
DECISION TREE

DECISION TREE

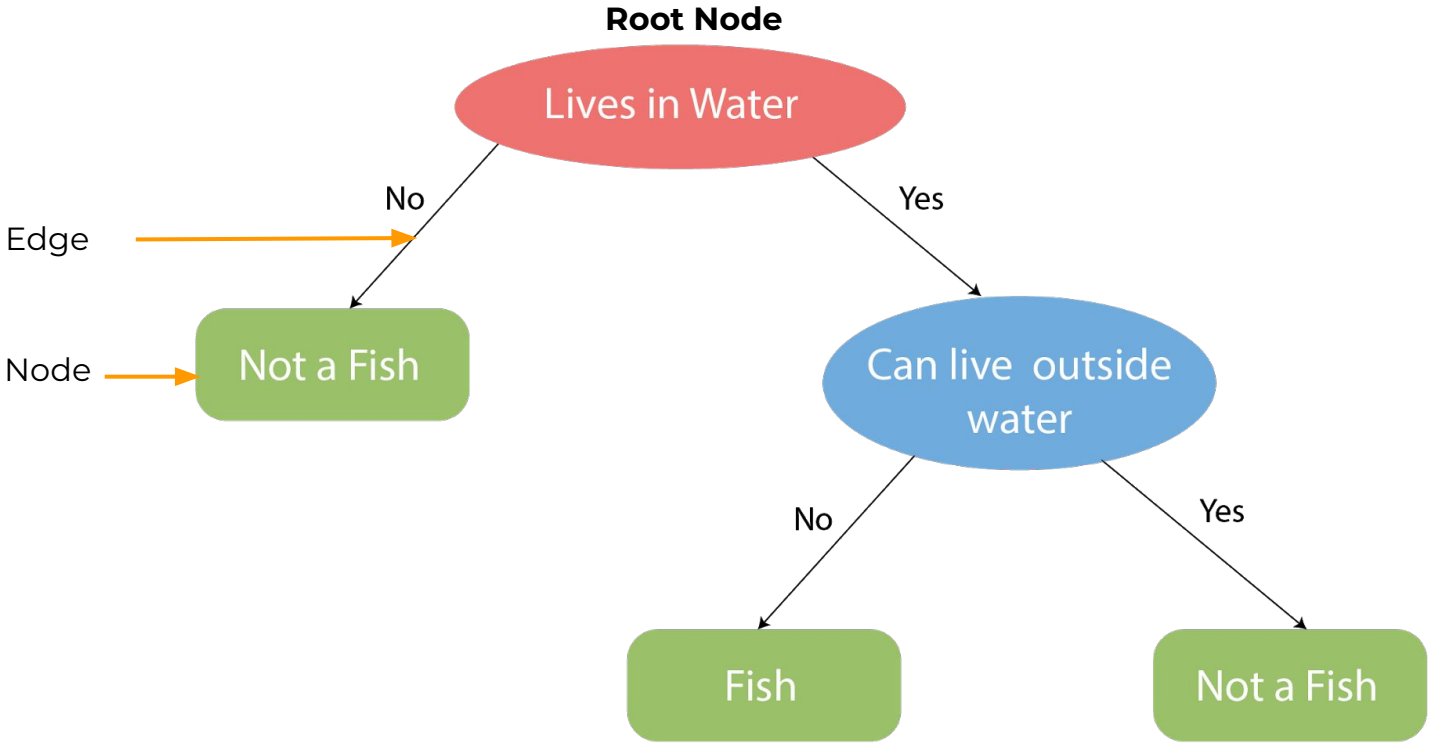
An inverted tree



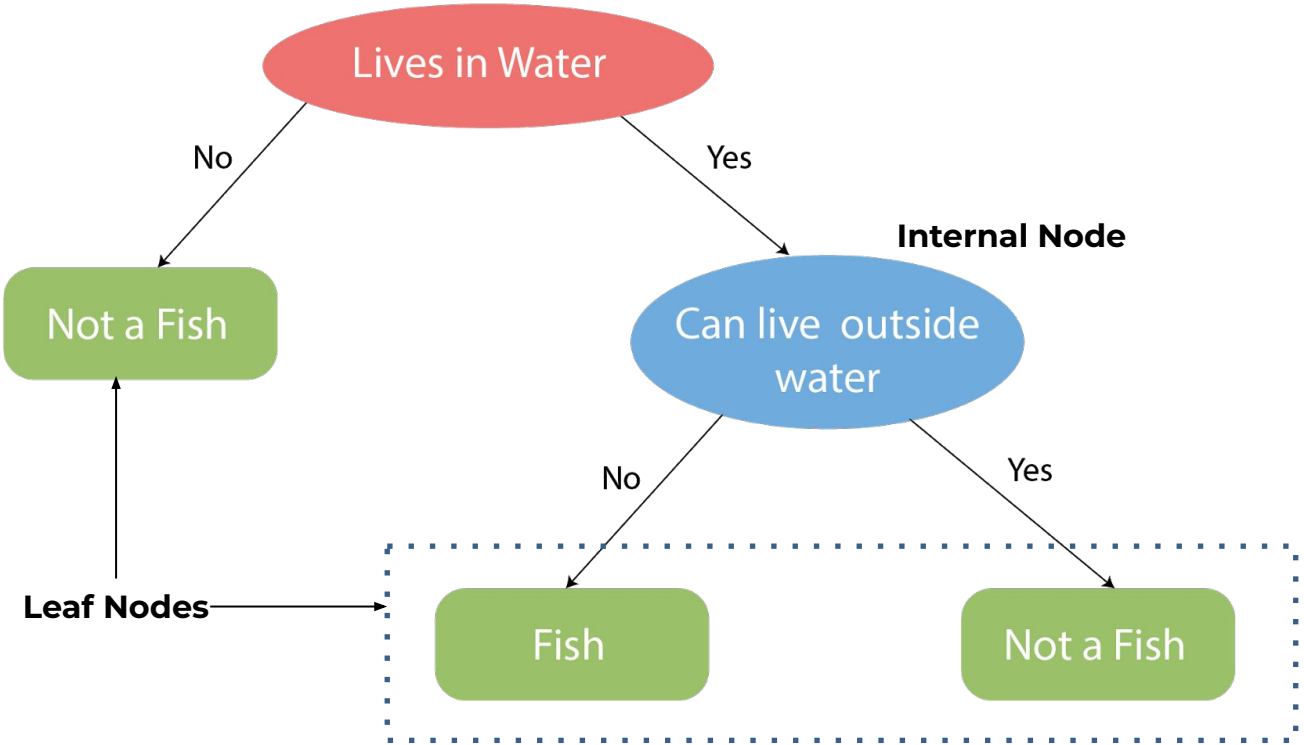
DECISION TREE: FISH CLASSIFIER



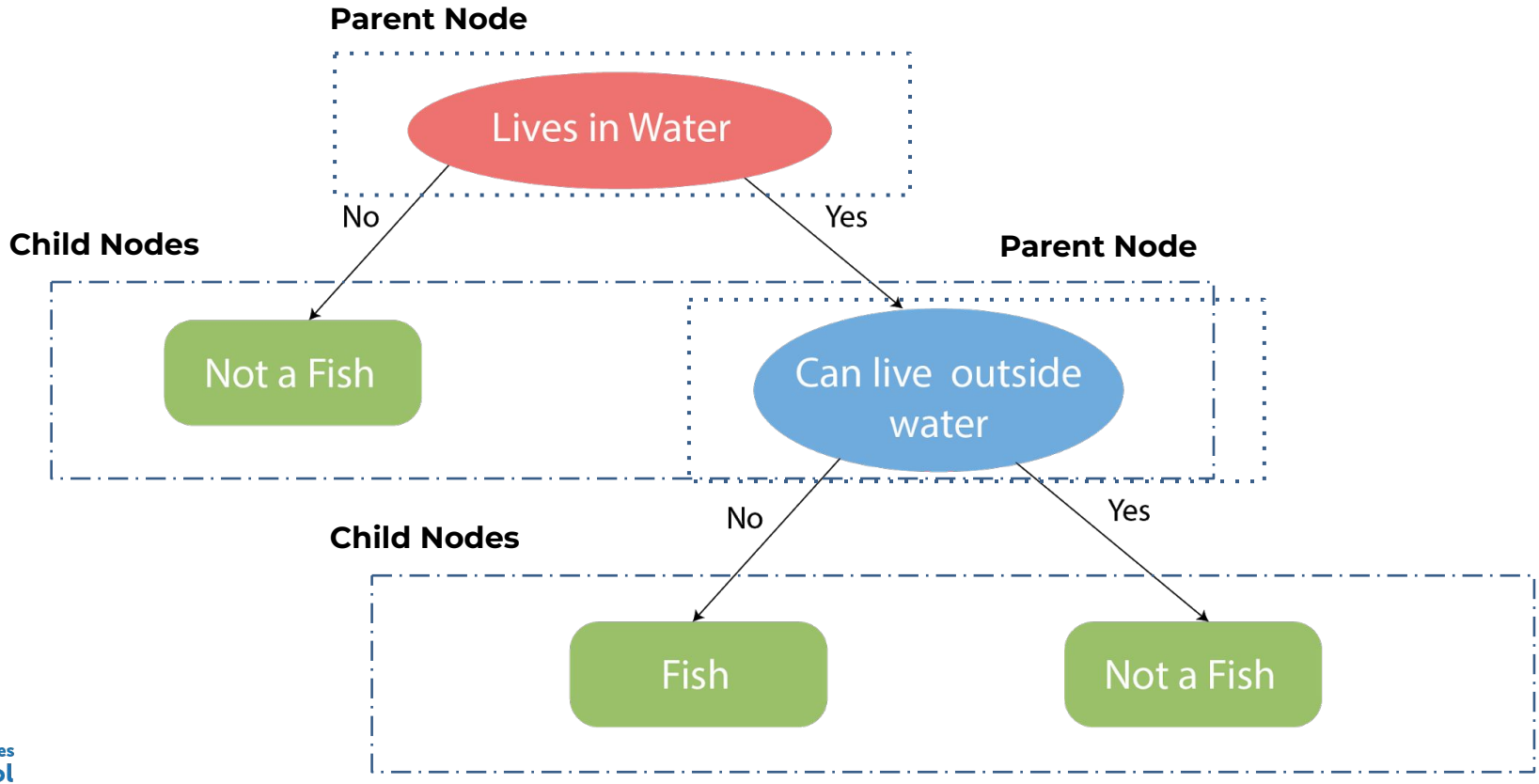
TYPES OF NODES: INTERNAL AND LEAF NODE



TYPES OF NODES: INTERNAL AND LEAF NODE



TYPES OF NODES: PARENT AND CHILD NODE



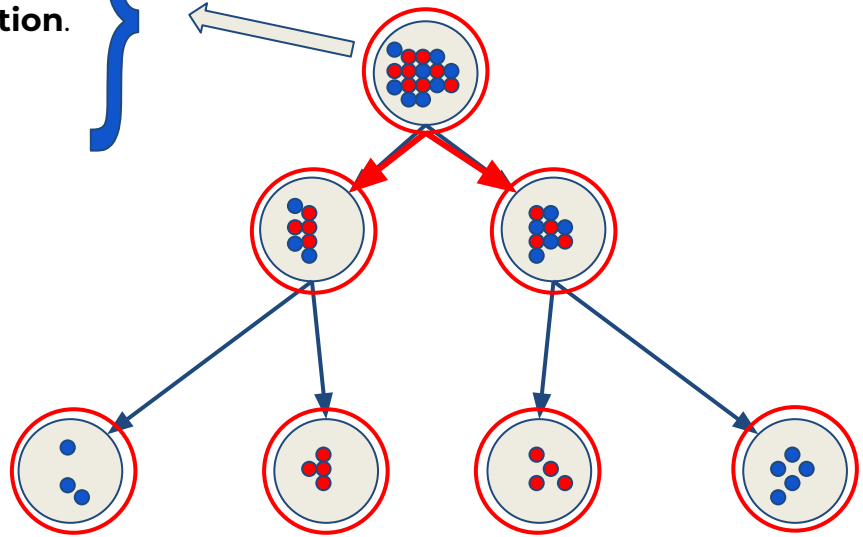
Basic Decision Tree algorithm

1. If all the instances in a node belong to **same class**, or any of the **early stopping criteria** is met, replace node with leaf node.

Else,

2. Compute **impurity metrics** for all the attributes.
3. Select attribute that yields **largest impurity reduction**.
4. **Create split** based on the selected attribute.

x_1	x_2	Label
..	..	●
..	..	●
..	..	..



Based on the different characteristics like Impurity metrics, pruning methods, attribute types and use cases, three algorithms: **ID3, C4.5, CART** are popular.

Characteristics→ / Algorithm ↓	Impurity Metric	Attribute Type	Pruning Method	Application
ID3	Information gain	Handles only categorical attributes	None	Only used for classification
C4.5	Gain ratio	Handles both categorical and numerical attributes	Error Based Pruning	Only used for classification
CART	Information gain and twoing criteria for classification and MSE, RSS, and Variance for regression	Handles both categorical and numerical attributes	Cost Complexity Pruning	Used for classification and regression

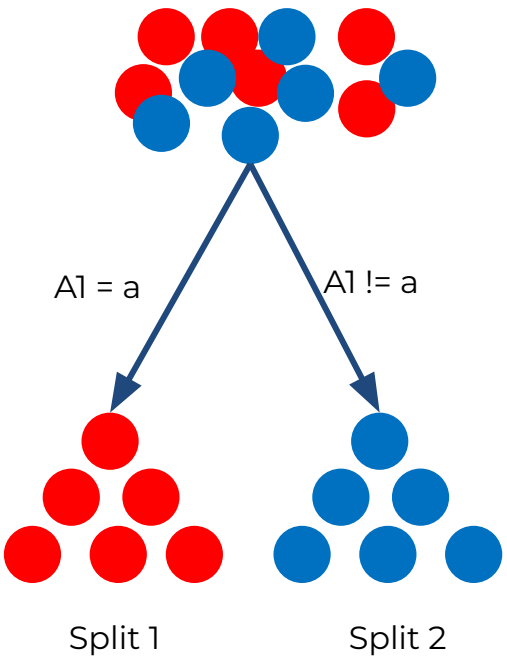
Fusemachines
AI School

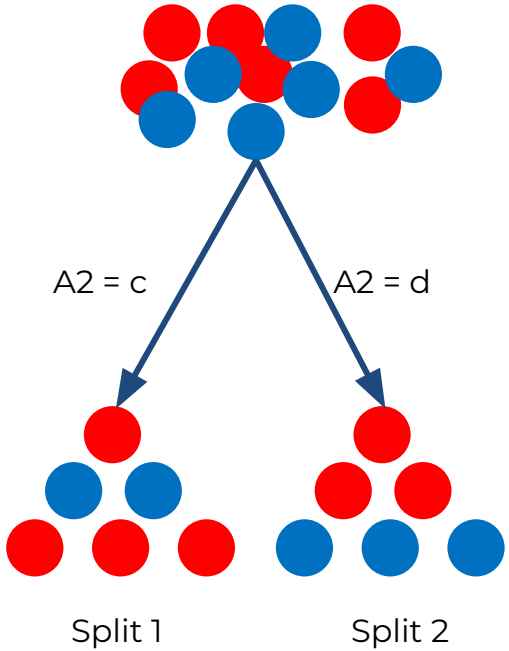
IMPURITY METRICS

OVERVIEW

1. INTRODUCTION
2. GINI
3. ENTROPY
4. INFORMATION GAIN
5. GAIN RATIO

Partitioning and Purity





Impurity Metrics

- **Gini**
- **Entropy**
- **Information Gain**
- Gain Ratio
- Mean Squared Error(MSE) for regression trees

GINI (Gini Index/impurity)

Probability of misclassifying a data point.

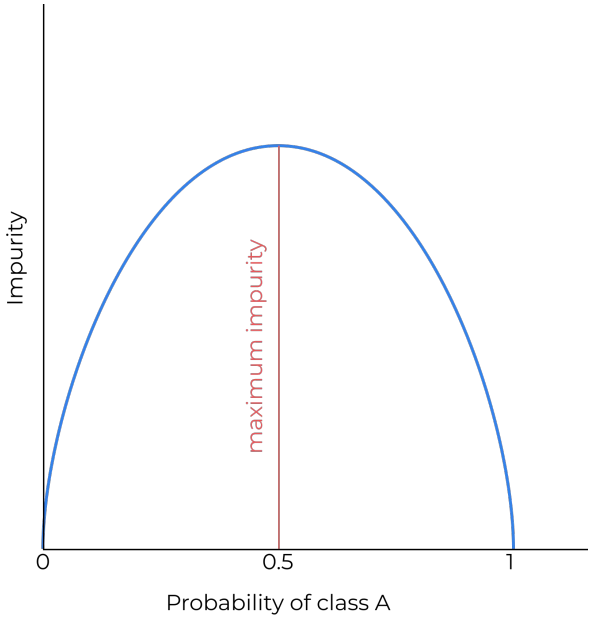
Lower gini is preferred → homogenous node, contains predominantly observations form a single class

$$\text{Gini} = 1 - \sum_{i=1}^{c} P_i^2$$

where P_i is the probability of an object being classified to a class i

Interpretation:

- Squared probability gives the probability that the class assignment of a randomly selected item is correct. First *p* as the probability of selection and the second *p*, the probability of assigning it the correct class.
- Subtracting it from 1 gives the probability of misclassification



GINI Example- Classification

Fish	Not a Fish	Remarks
10	0	Pure Data
1	9	Unevenly Distributed Impure Data
7	3	Unevenly Distributed Impure Data
5	5	Evenly Distributed Impure Data

Gini index
0
0.18
0.42
0.5

$$= 1 - (\frac{7}{10})^2 - (\frac{3}{10})^2$$
$$= 1 - \frac{49}{100} - \frac{9}{100}$$
$$= \frac{100-49-9}{100}$$
$$= \frac{42}{100} = 0.42$$

- Impurity is minimum when the node is homogenous (samples only from a single class)
- Impurity is maximum when there are equal number of samples from each class

ENTROPY & INFORMATION GAIN

$$\text{Entropy} = -\sum_{i=1}^{c} P_i \log P_i$$

where P_i is the probability of an object being classified to a class i

Entropy of parent node

$$\text{Information Gain}(G) = \text{Entropy}(L) - \text{Entropy}(L, A)$$

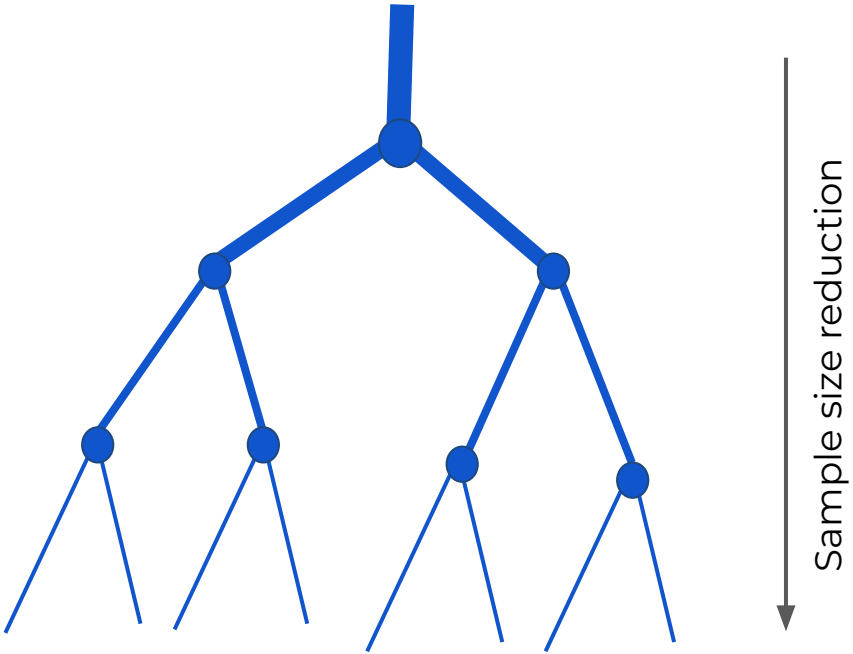
L denotes the label/class

Entropy of child nodes after splitting parent node using attribute A

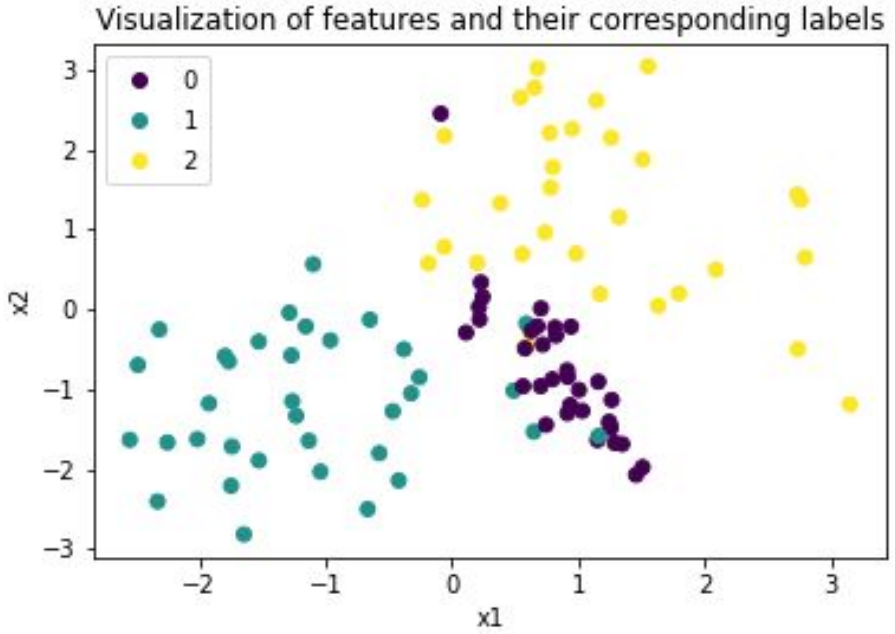
Fusemachines
AI School

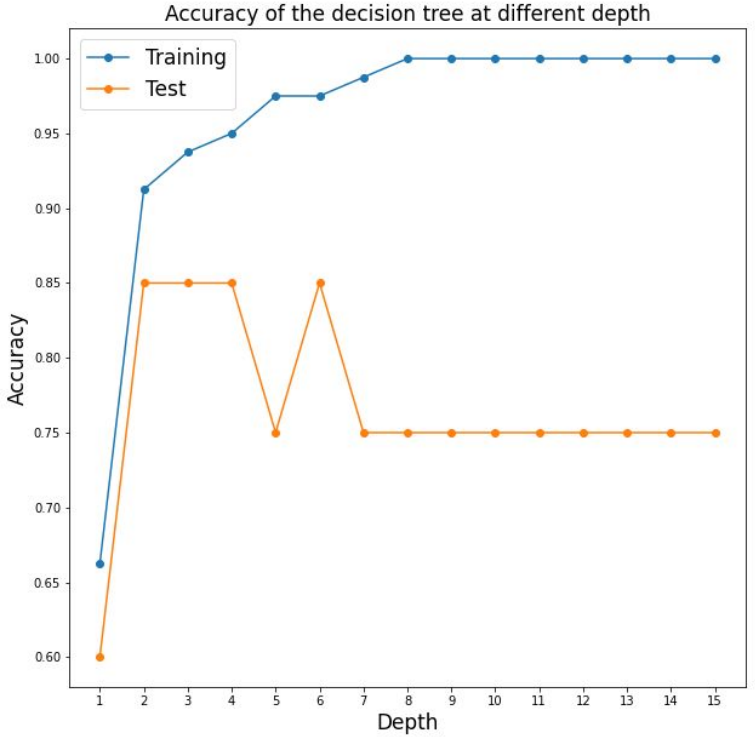
Overfitting

Deep trees are highly prone to overfitting

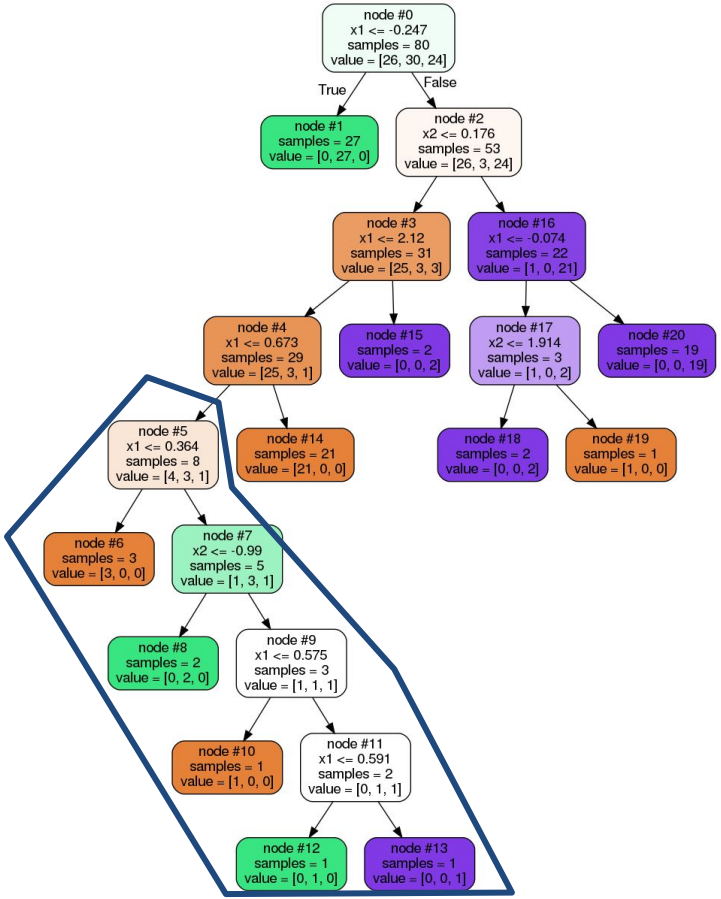


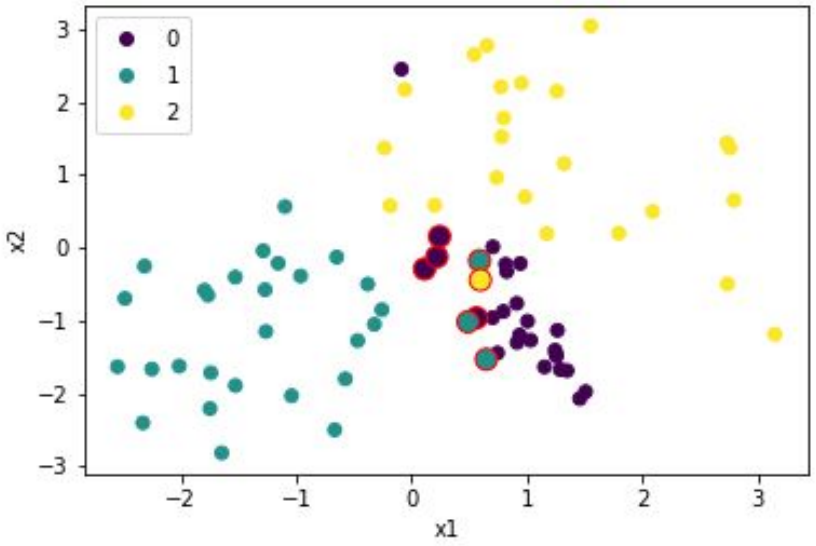
Synthetic Dataset



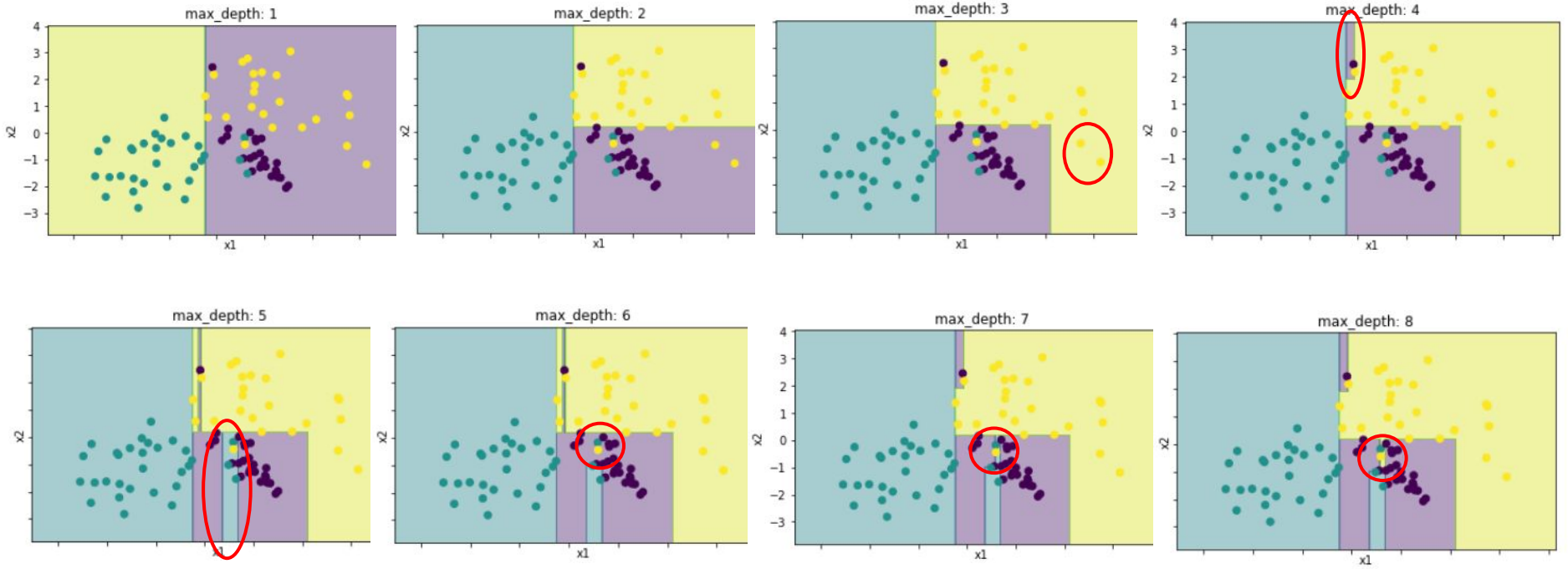


Tree Structure Visualization





Decision Surface Visualization



Regularization Techniques

1. Early Stopping

2. Pruning

Early stopping criteria are specified before a tree is built.

Early Stopping Techniques

1. Depth Limitation

Grow tree upto a given depth and then stop.

2. Impurity Threshold

Splits only if the next split results reduction of impurity greater than specified value.

$$R = \text{impurity} - \frac{N_{tR}}{N_t} * \text{impurity\_right} - \frac{N_{tL}}{N_t} * \text{impurity\_left}$$

impurity = impurity of current node
impurity_right = impurity of right child
impurity_left = impurity of left child

N_{tR} = Number of samples in right child

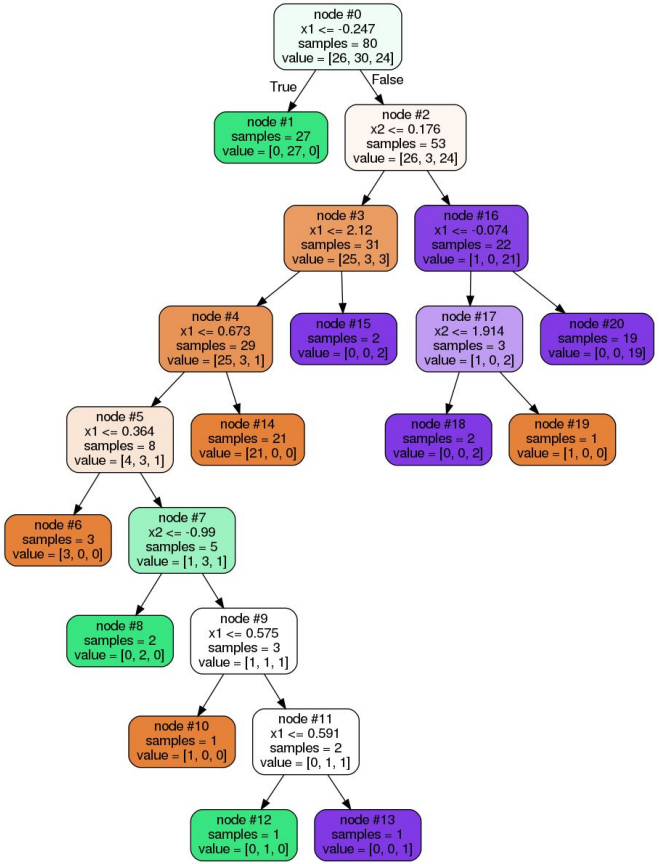
N_{tL} = Number of samples in left child

N_t = Number of samples in current node

3. Minimum number of samples

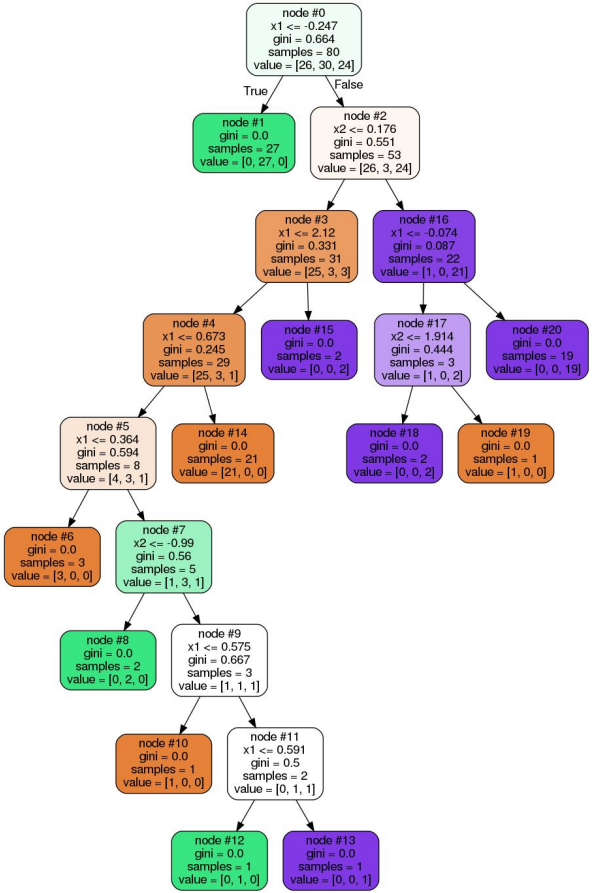
Splits only if the node contains at least specified number of samples.

Depth Limitation

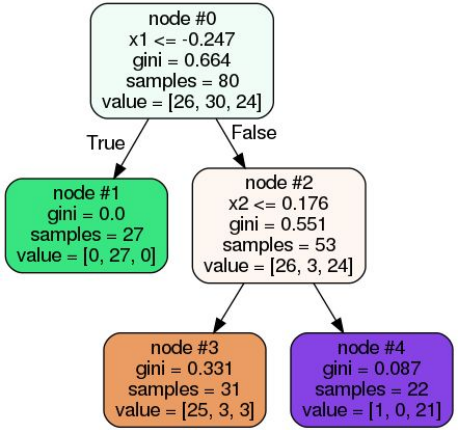


max_depth = 3

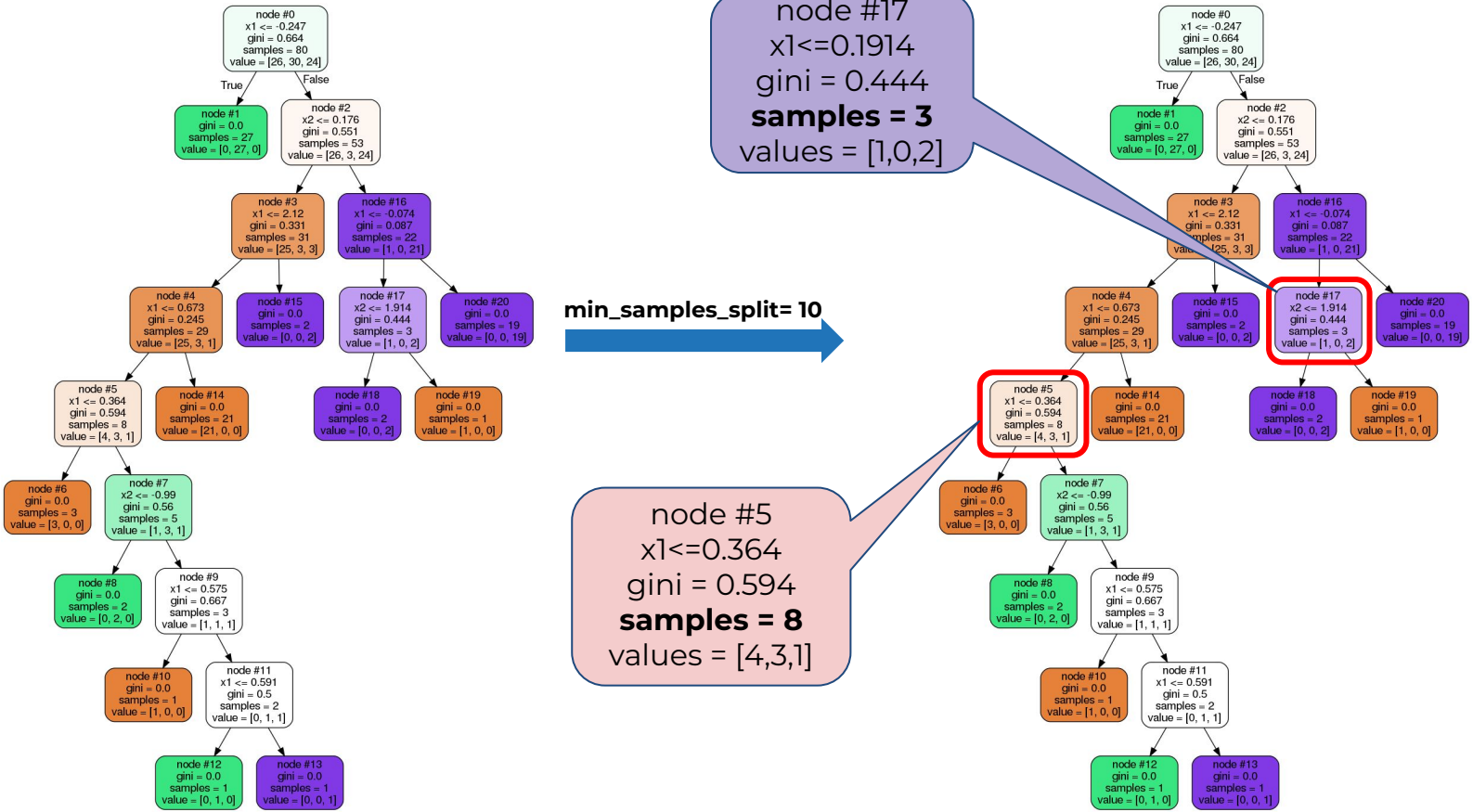
Impurity Threshold



min_impurity_decrease= 0.1



Minimum Samples in a node



Fusemachines
AI School

Problem with Early Stopping

Arbitrarily chosen(Not data driven)

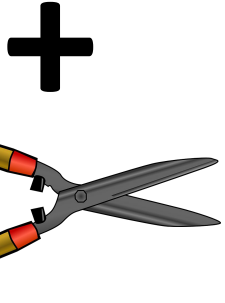
Leads to

- Overfitting
- Underfitting

Fusemachines
AI School

Regularization: Pruning

Introduction



Pruning means growing a decision tree to its entirety and then recursively cutting/removing the nodes or sub-tree in a bottom up approach

Pruning Techniques

1. Cost-Complexity Pruning
2. Reduced Error Pruning
3. Minimum Error Pruning
4. Error-based Pruning

Cost-Complexity Pruning: Basic Idea

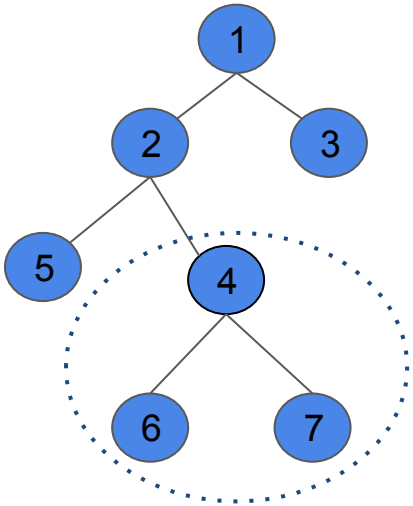
Cost-Complexity = **Cost(error)** + **Complexity**

If the presence of a branch doesn't reduce the error, prune it!!

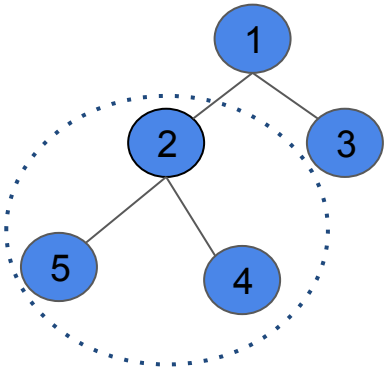
Basic Idea

1. Grow a fully grown decision tree
2. Weakest-link: Largest branch with least error reduction.
3. Prune the weakest-link.
4. Generate a series of trees by pruning weakest link.
5. Select the tree that yields best performance on validation set.

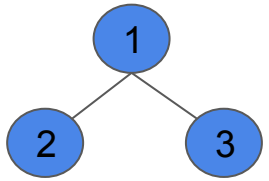
Cost-Complexity Pruning: Basic Idea



Tree 1 : T



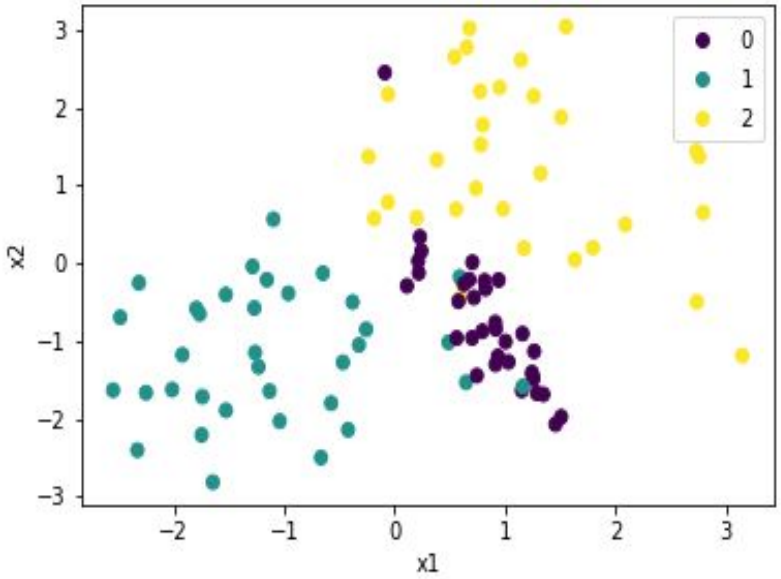
Tree 2 : $T - T_1$



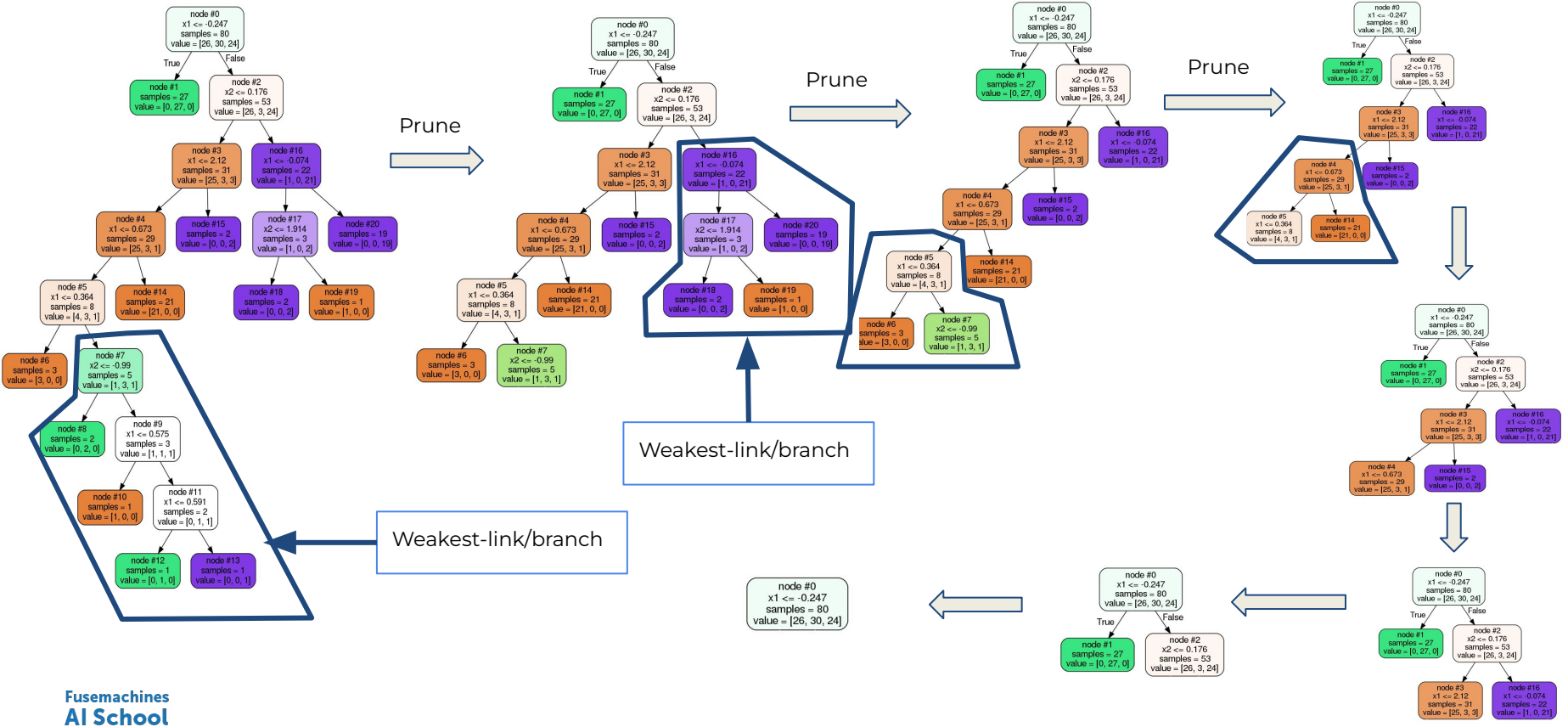
Tree 3 : $T - T_2$

Cost-Complexity Pruning

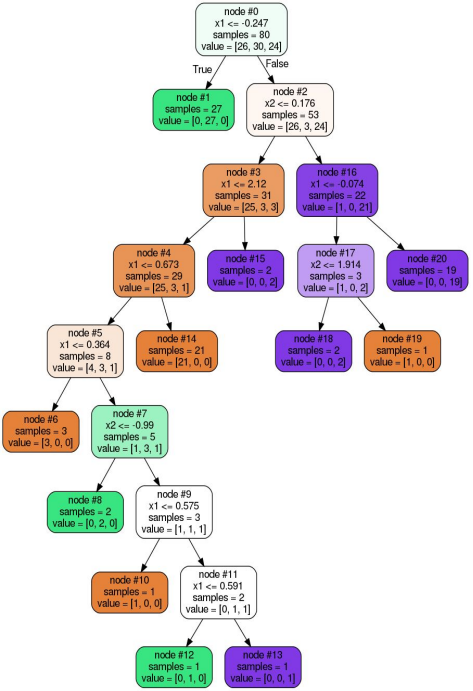
Synthetic data generated using **scikit-learn's make_classification()**



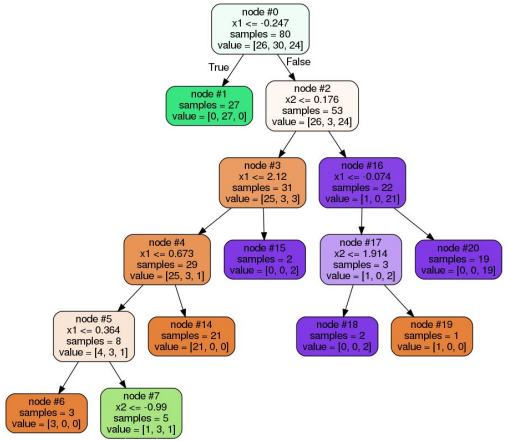
Cost-Complexity Pruning : Visualization



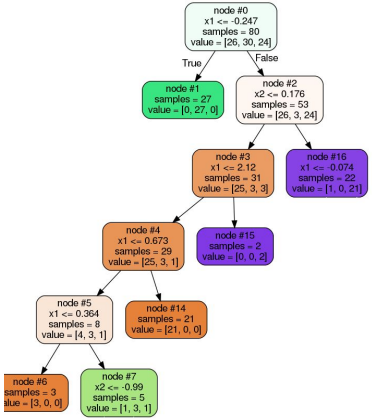
Cost-Complexity Pruning : Series of trees



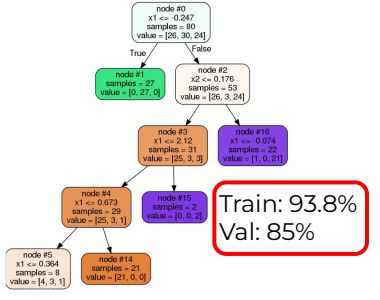
Train: 100%
Val: 75%



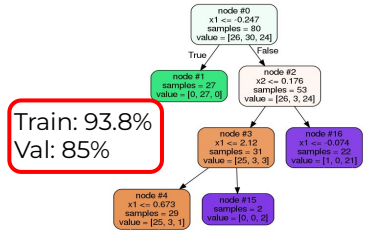
Train: 97.5%
Val: 75%



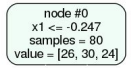
Train: 96.3%
Val: 75%



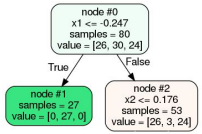
Train: 93.8%
Val: 85%



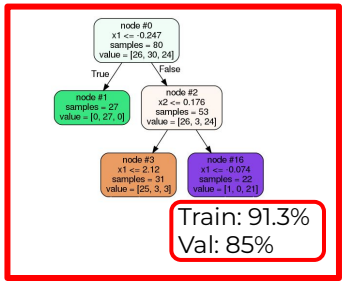
Train: 93.8%
Val: 85%



Train: 37.5%
Val: 25%



Train: 66.3%
Val: 60%



Train: 91.3%
Val: 85%

Fusemachines
AI School

Identify weakest-link

- Weakest link is quantified by the smallest value of α.

$$\alpha = \frac{R(t) - R(T_t)}{N_T - 1}$$

where, $R(t)$ is the error cost of node t.
$R(T_t)$ is the error cost of subtree/branch T rooted at node t.
N_T is the number of leaf nodes in subtree T.

- Error cost of node t is $R(t) = r(t) * p(t)$

$$r(t) = \frac{\text{number of samples misclassified in node t}}{\text{number of all samples in node t}} \quad ; \text{for classification}$$

$$r(t) = \frac{\text{sum of squared error of samples in node t}}{\text{number of all samples in node t}} \quad ; \text{for regression}$$

$$p(t) = \frac{\text{number of all samples in node t}}{\text{total number of training samples}}$$

- Error cost of subtree/branch T is $R(T_t) = \sum_{i=\text{all leaves in subtree(T)}} R(i)$

Computing α

$$R(t) = r(t) * p(t) = \frac{4}{29} * \frac{29}{80} = \frac{1}{20}$$

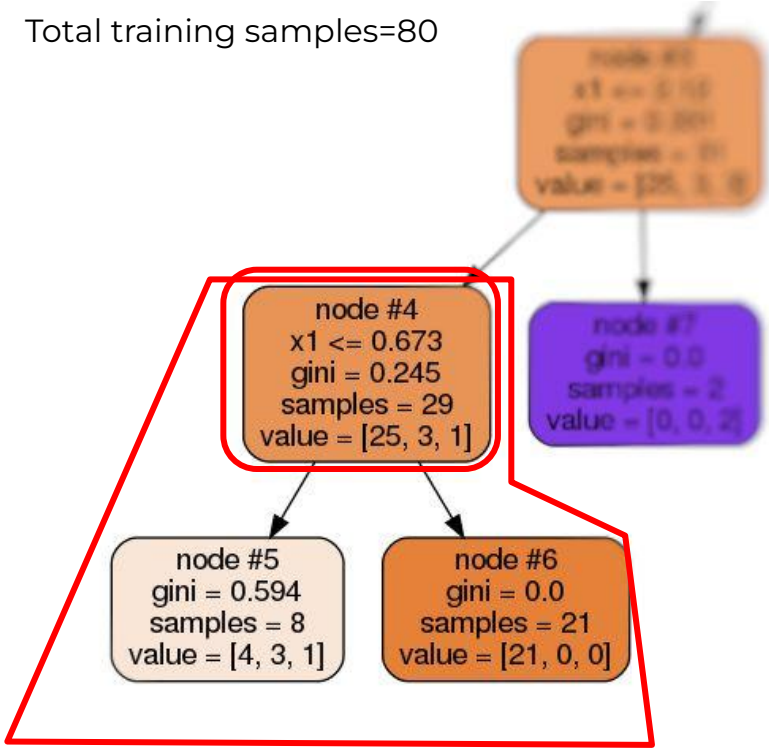
$$R(T_t) = R(\text{node \#5}) + R(\text{node \#6})$$

$$= \frac{4}{8} * \frac{8}{80} + \frac{0}{21} * \frac{21}{80} = \frac{1}{20}$$

$$N_T = 2$$

$$\alpha = \frac{R(t) - R(T_t)}{N_T - 1} = \frac{1/20 - 1/20}{2 - 1} = 0$$

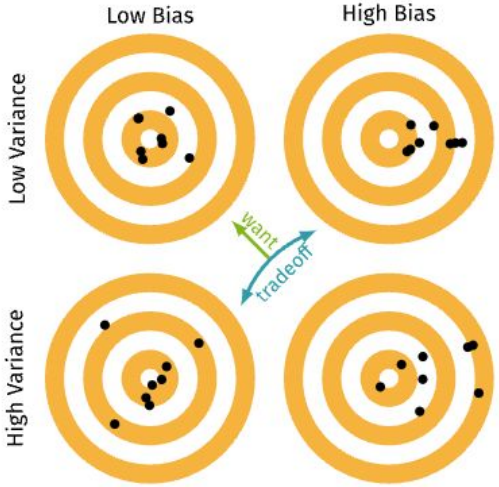
Total training samples=80



Fusemachines
AI School

ERROR DECOMPOSITION

BIAS & VARIANCE



Bias: how far the estimate is from the actual value.

- High-bias: The model can't fit the data properly
- Solution: Fit complex model

Variance: spread of estimates from the mean estimation .

- High-variance: The model overfits.
- Solution: Fit a simpler model

Bias-Variance Trade off

Fusemachines
AI School

Image Source: https://dm.cs.tu-dortmund.de/en/mlbits/foundations-bias-variance/

Error Decomposition of MSE

$$\text{Expected Loss} = (bias)^2 + variance + noise$$

$$\begin{aligned}
\mathbb{E}\left[(\hat{\theta} - \theta^*)^2\right] &= \mathbb{E}\left[\left[(\hat{\theta} - \overline{\theta}) + (\overline{\theta} - \theta^*)\right]^2\right] \\
&= \mathbb{E}\left[\left(\hat{\theta} - \overline{\theta}\right)^2\right] + 2(\overline{\theta} - \theta^*)\mathbb{E}\left[\hat{\theta} - \overline{\theta}\right] + (\overline{\theta} - \theta^*)^2 \\
&= \mathbb{E}\left[\left(\hat{\theta} - \overline{\theta}\right)^2\right] + (\overline{\theta} - \theta^*)^2 \\
&= \mathbb{V}\left[\hat{\theta}\right] + \text{bias}^2(\hat{\theta})
\end{aligned}$$

In words,

$$\boxed{\text{MSE} = \text{variance} + \text{bias}^2}$$

$\hat{\theta}$: prediction

θ^* : true parameter value

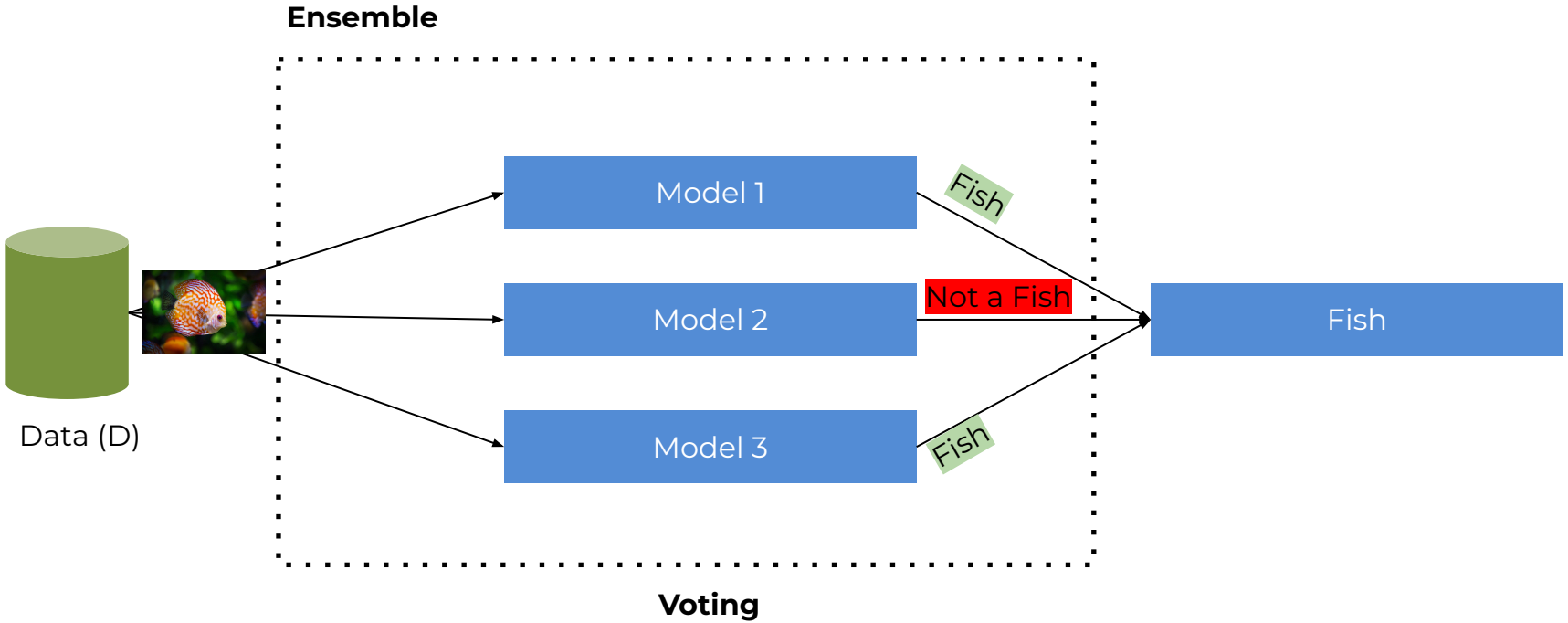
$\overline{\theta} = \mathbb{E}[\hat{\theta}]$: expected value of the estimate

Page 161, Kevin P. Murphy, Probabilistic Machine Learning: An Introduction (2022)

Fusemachines
AI School

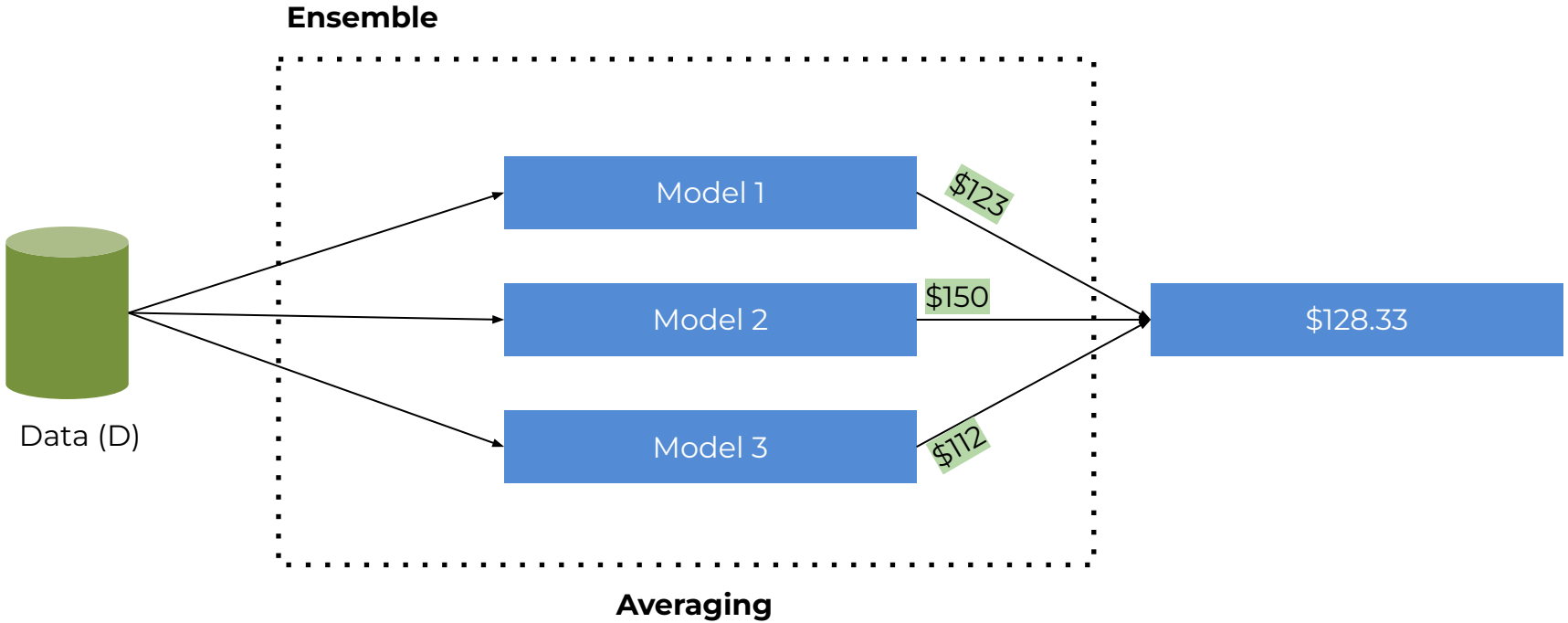
ENSEMBLE
(WISDOM OF CROWD)

Ensemble Example: Classification



In ensembling, we combine decisions from multiple models to improve the overall performance.

Ensemble Example: Regression

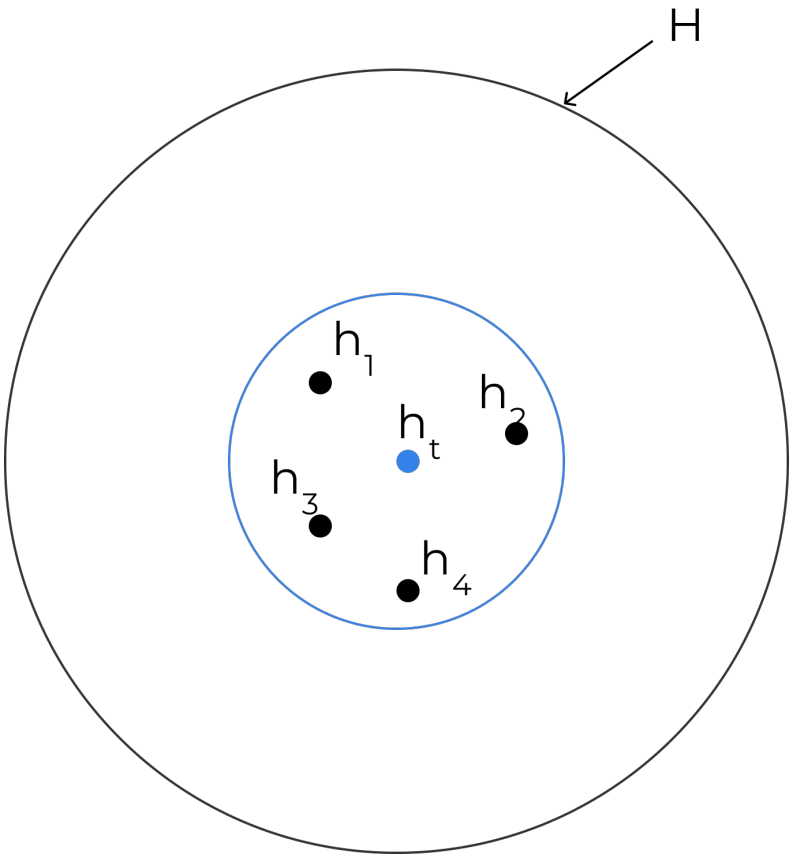


Why is an ensemble better than individual model ?

1. **Statistical Reason:**

- A learning algorithm can be viewed as searching a space of hypotheses - all possible models, to find the best possible hypothesis.
- Each of the individual hypotheses, may not be a good approximation of the true hypothesis .
- However, taking their average can get us closer to it .

2. Computational Reason
3. Representational Reason



Thomas G. Dietterich, Ensemble Methods in Machine Learning (https://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf)

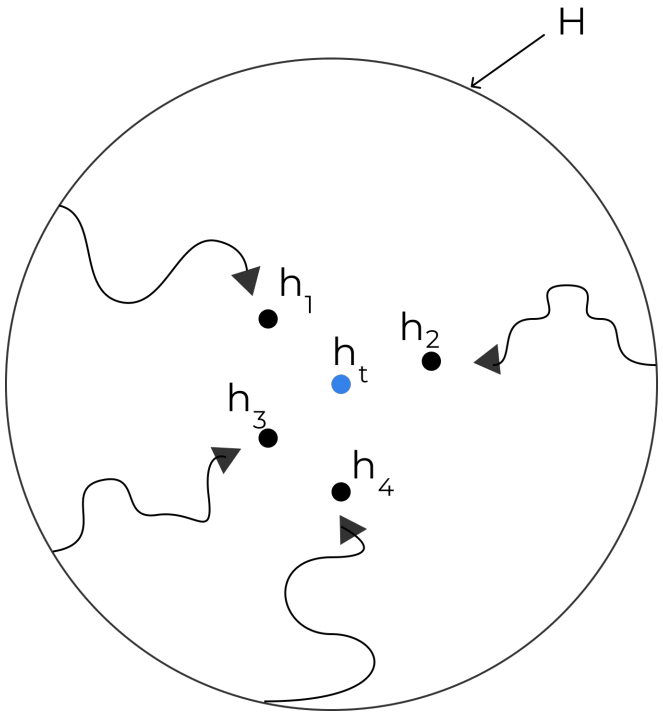
Why is an ensemble better than individual model ?

1. Statistical Reason

2. **Computational Reason:**

- Machine learning algorithms like decision trees, neural networks, etc. work by performing a local search.
- These algorithms may get stuck in local optima even when there is enough training data.
- An ensemble with different models running the local search from different points provides a good approximation of than any individual model.

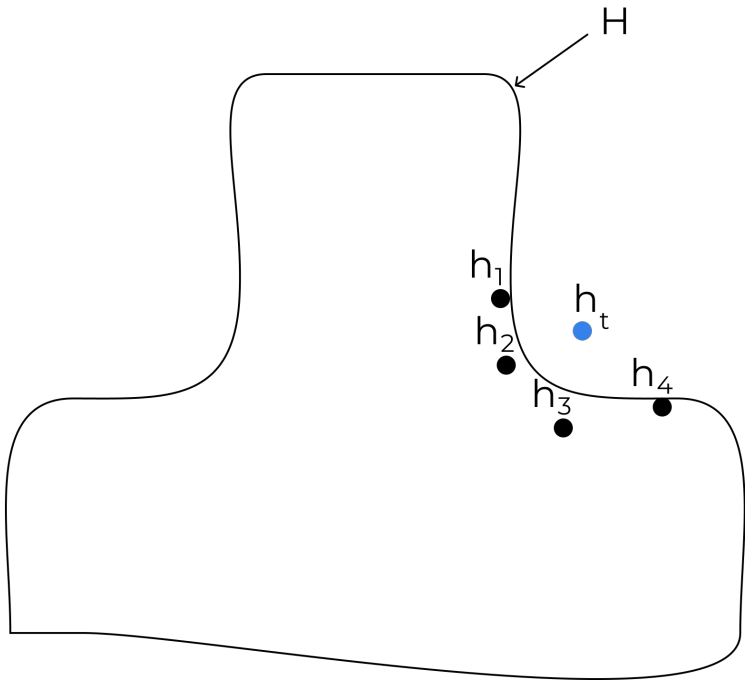
3. Representational Reason



Why is an ensemble better than individual model ?

1. Statistical Reason
2. Computational Reason
3. **Representational Reason:**

- In some applications, the true function cannot be represented by any of the hypotheses.
- But, by combining hypotheses, it may be possible to expand the space of representable functions.

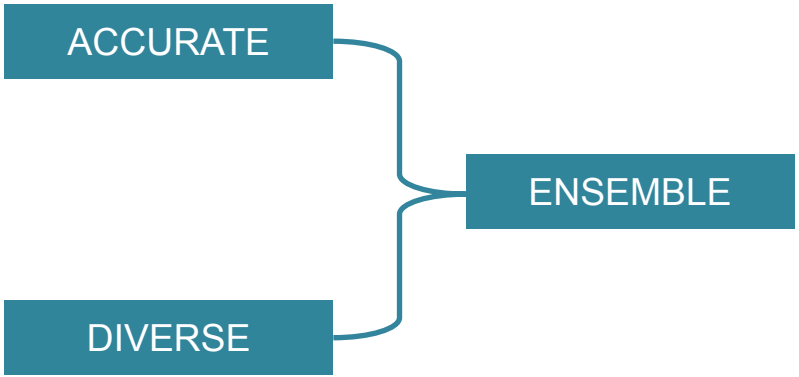


PREREQUISITE TO ENSEMBLING

Can any arbitrary models participate in an ensemble?

↓

NO



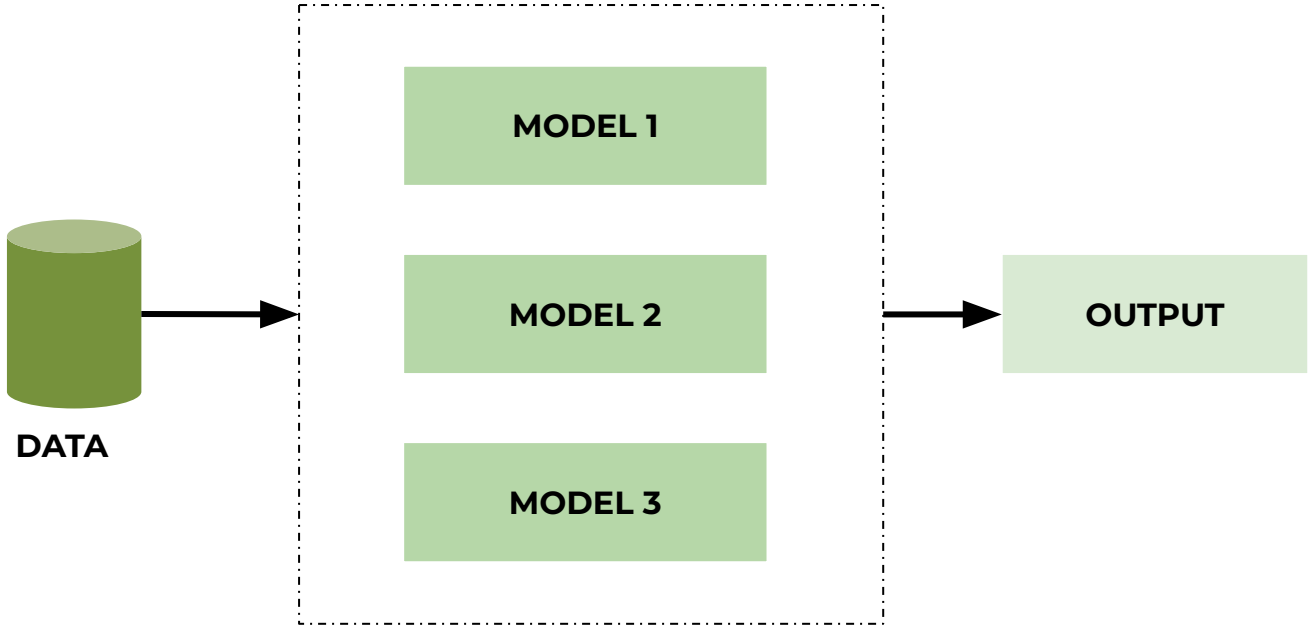
POPULAR ENSEMBLING TECHNIQUES

1. **Bagging/Bootstrap Aggregation**
2. **Boosting**
3. Stacking

BAGGING

Bagging/Bootstrap Aggregation: Train multiple independent models on different subsets of the data and combine their output

- random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once



BAGGING

Bags —------>

Original Data(D)	1	2	3	4
D_1	2	2	1	3
D_2	4	1	1	2
. . .				
D_m	4	3	3	1

Bootstrap samples

BAGGING

Regression

$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_i$$

Classification

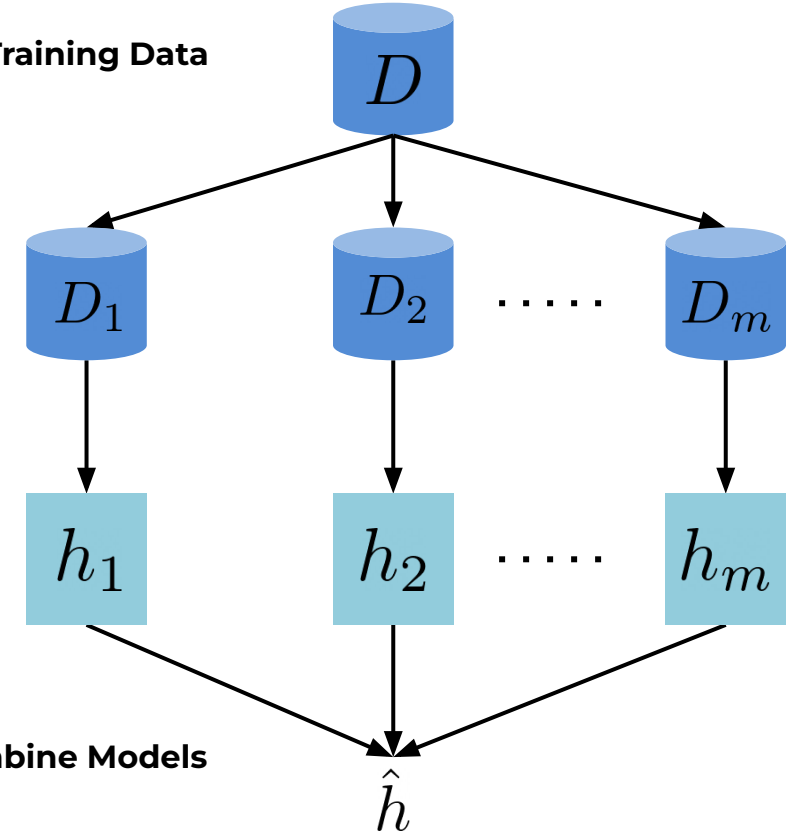
$$\hat{h} = \text{Mode}(h_i, i = 1 \ldots m)$$

(Majority Voting)

Create Bootstrap Samples

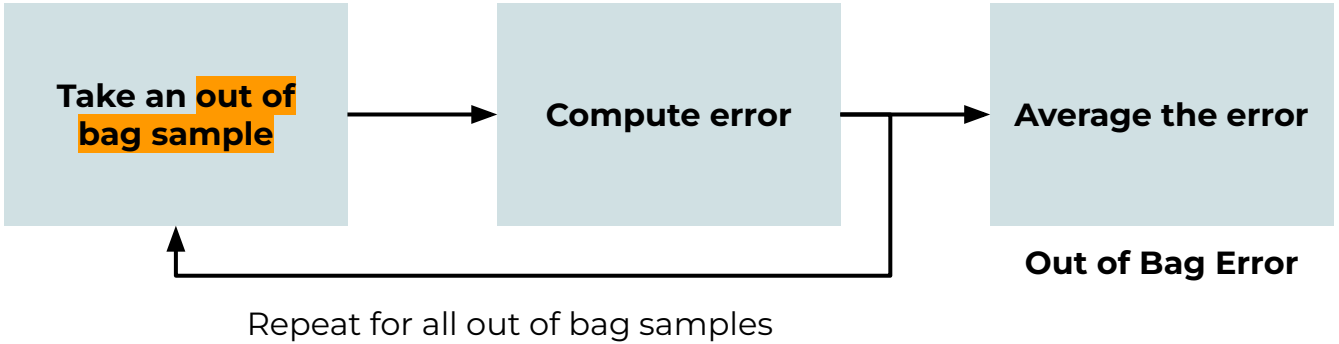
Build Multiple Models

Combine Models



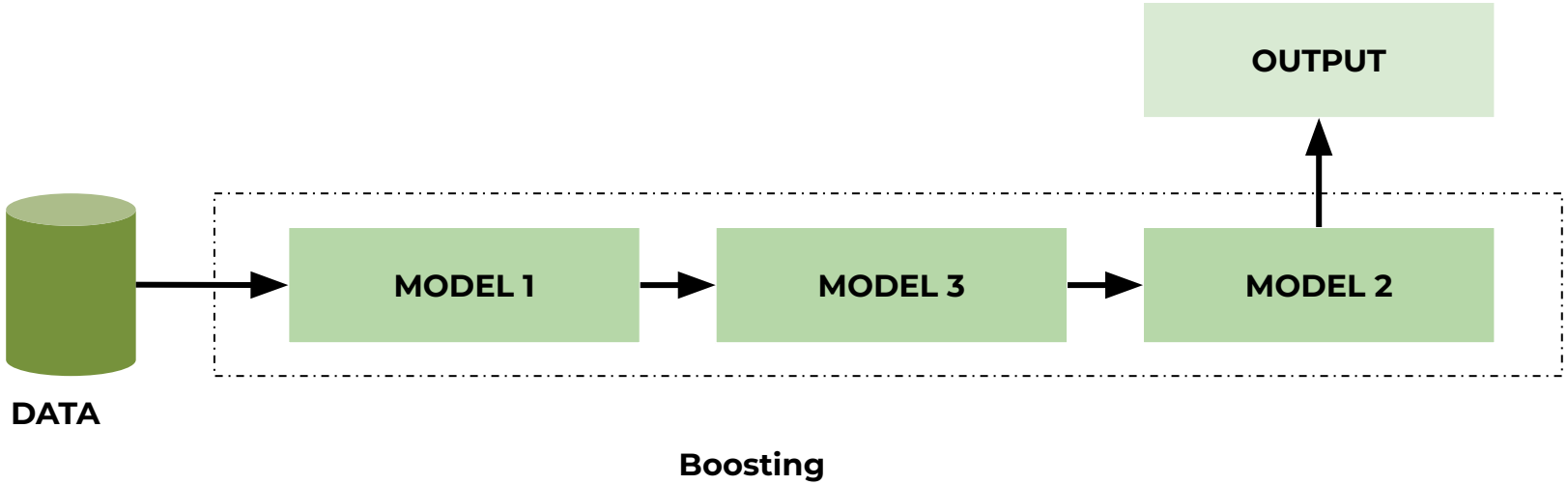
ADVANTAGES OF BAGGING

- Easy to implement: no change in algorithm
- Reduces variance
- Parallel Execution
- Gives an **unbiased estimate of the test error:**
    - ~63% of unique data in each bag. Samples present in the original dataset but absent in bootstrapped data are called out of bag samples (OOB).



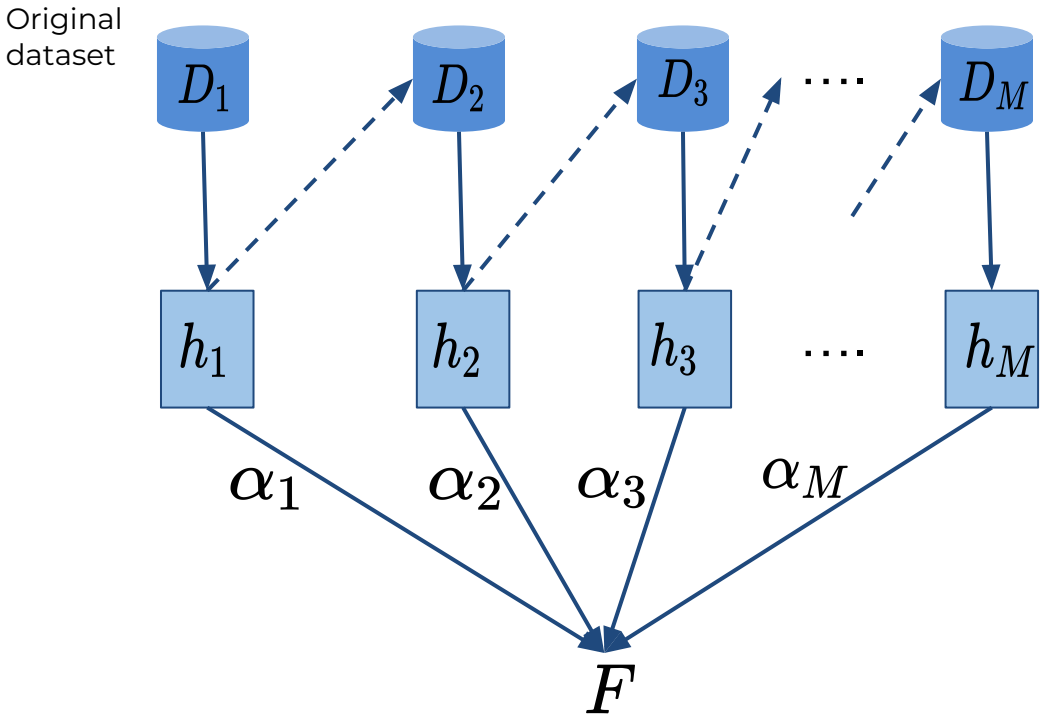
BOOSTING

Boosting: Train multiple models sequentially where each new model tries to rectify the mistakes of previous model

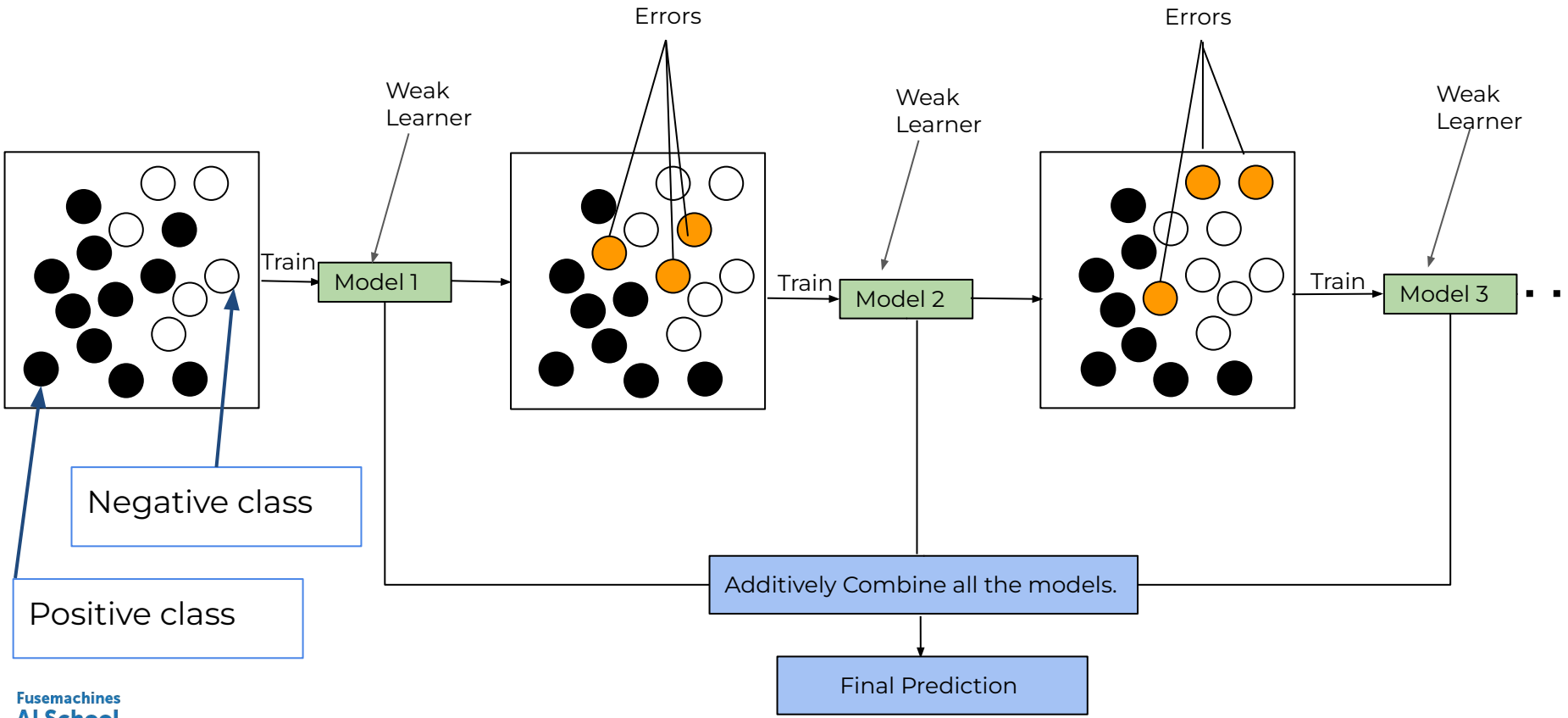


Boosting

Boosting builds multiple models on modified version of training data and aggregate them using **weighted averaging**

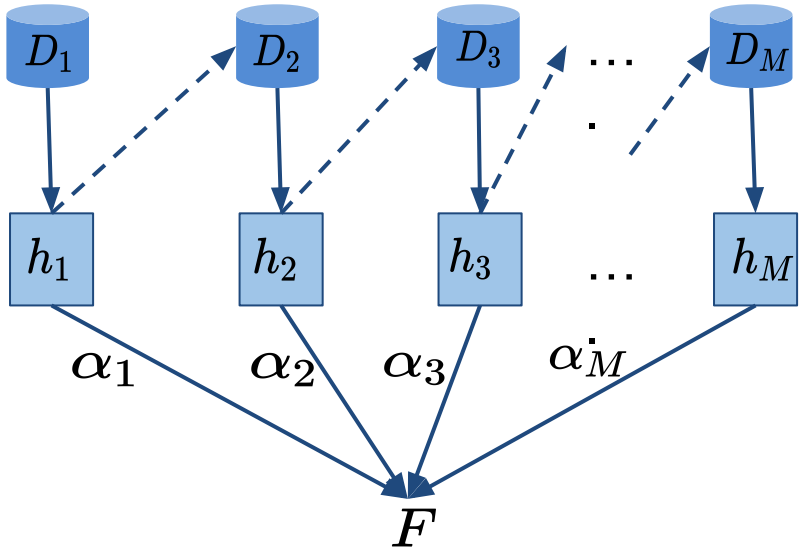


Boosting additively combines multiple weak learners



Working of Boosting

Original dataset



$$F_0(\mathbf{x}) = 0$$

$$F_1(\mathbf{x}) = F_0(\mathbf{x}) + \alpha_1 h_1(\mathbf{x}) = \alpha_1 h_1(\mathbf{x})$$

$$F_2(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x})$$

$$\ldots.$$

$$F_M(\mathbf{x}) = F(\mathbf{x}) = \sum_{t=1}^{M} \alpha_t h_t(\mathbf{x})$$

What is α ?

Ans : Contribution factor

Bagging and Boosting comparison

Bagging

Aggregate multiple **unstable** learners

Goal: **Reduce Variance**

Parallel Process

Simple aggregation

Boosting

Aggregate multiple **weak** learners

Goal: **Reduce Bias**

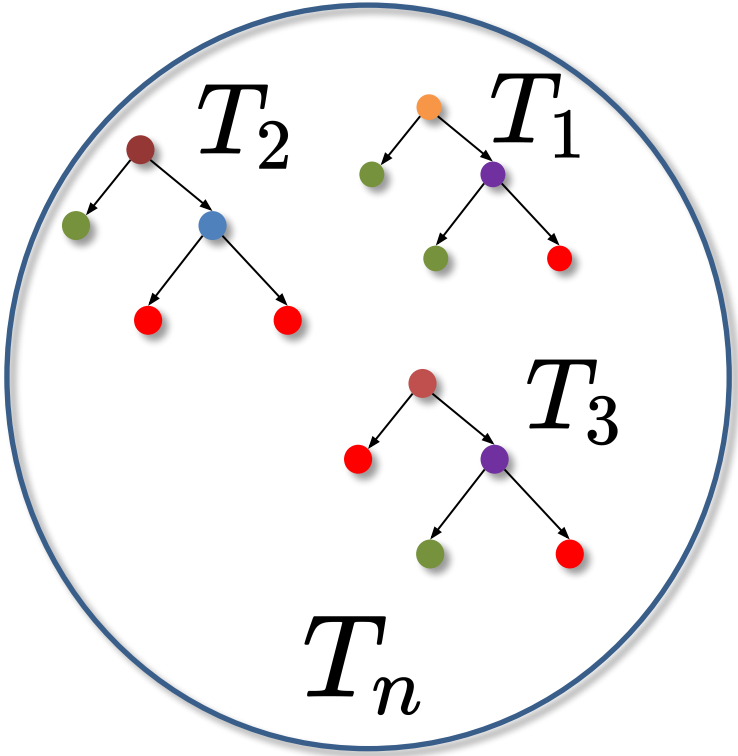
Sequential Process

Weighted Aggregation

Fusemachines
AI School

RANDOM FOREST

Forest = Ensemble of Trees

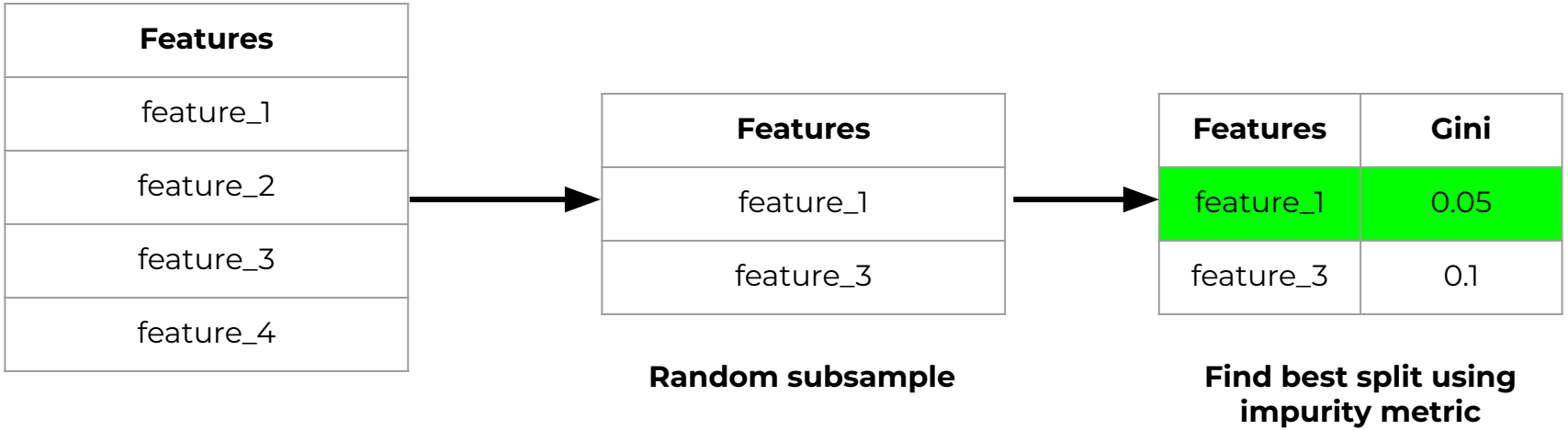


- A depth unlimited decision tree or deep decision tree is a strong learner. It easily overfits the training data.
- The random forest's main goal is to combine multiple overfitted decision trees to reduce **variance or overfitting**.

Leo Breiman https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

Random Forest = Bagging + Random Feature Subsampling

Random feature subsampling: random subset of features -> reduces correlation between the trees → reduces the variance even further



ALGORITHM

Regression

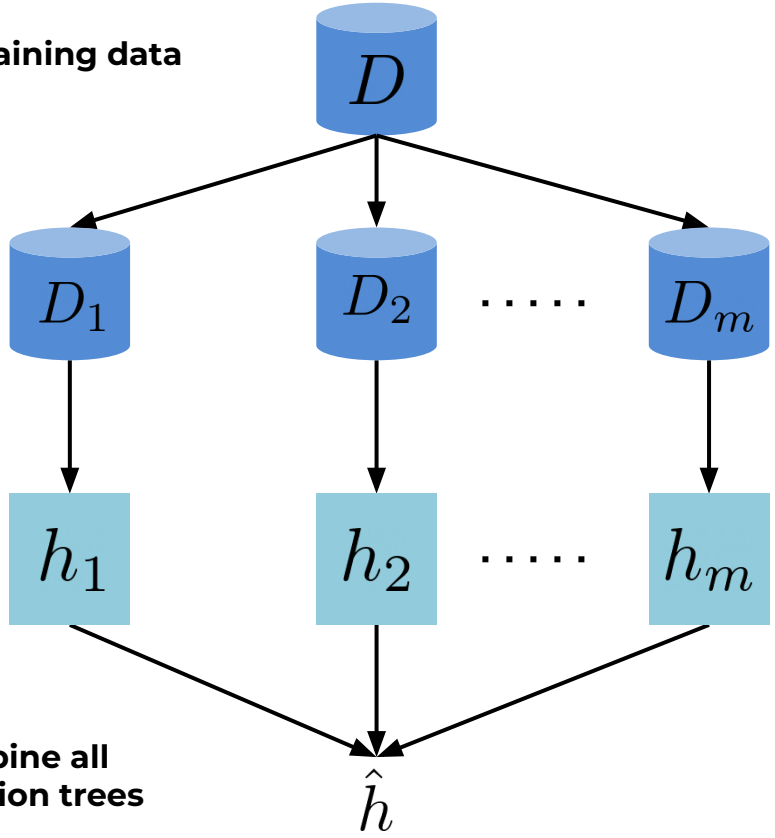
$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_i$$

Classification

$$\hat{h} = \text{Mode}(h_i, i = 1 \ldots m)$$

Create bootstrap samples

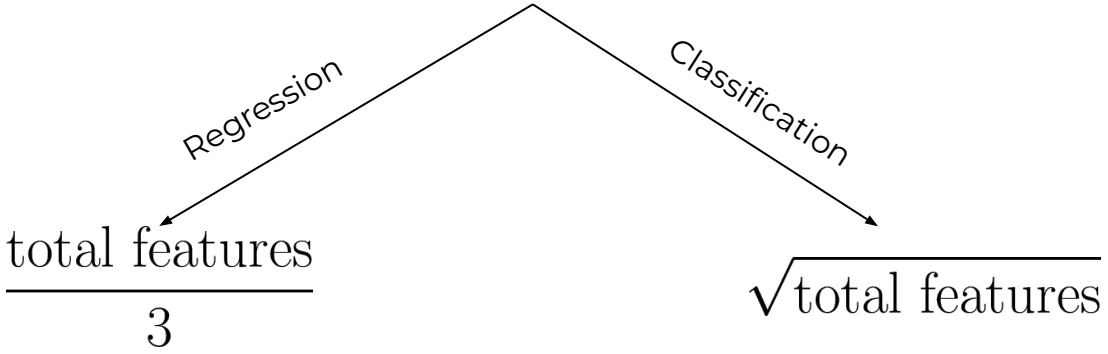
Build multiple decision trees (use random feature subsampling)



ALGORITHM

Two hyperparameters(excluding hyperparameters of decision tree)

- Forest Size = #trees
- Number of features to select during random feature subsampling



Fusemachines
AI School

Boosting based ensemble

(AdaBoost, Gradient Boosted Models)

Introduction

Boosting: Add models iteratively to compensates the **error or shortcoming** of an existing model.

$$F(\mathbf{x}) = \sum_i \alpha_i h_i(\mathbf{x})$$

How to identify the error or shortcomings of a model?

- **Weight of data points:** Adaptive Boosting (AdaBoost)
- **Gradient:** Gradient Boosting

AdaBoost

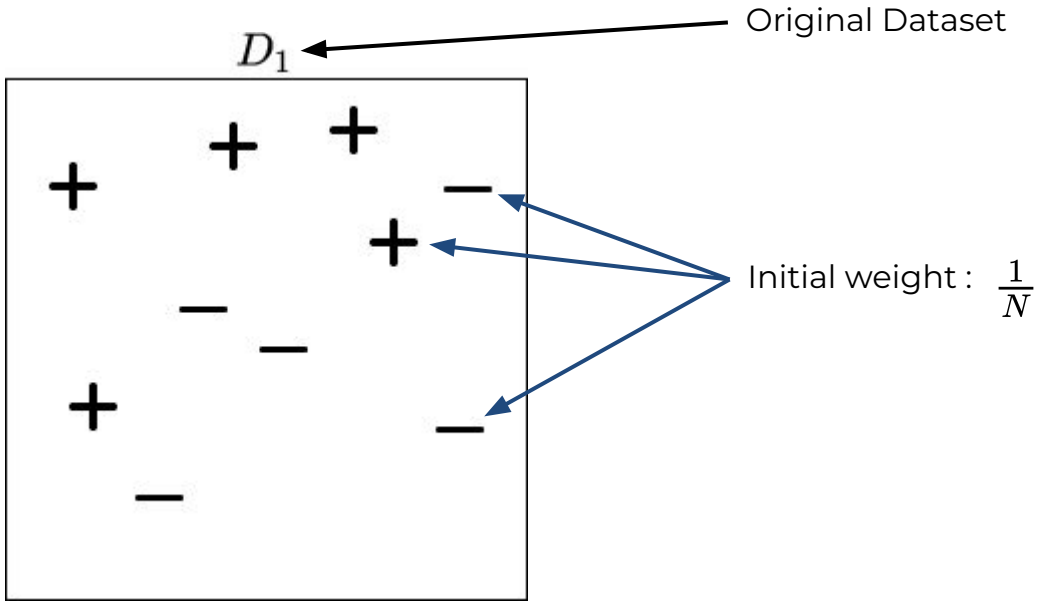
One of the first and most popular boosting algorithm

Data modification involves weight update

Weight of **mis-classified** samples

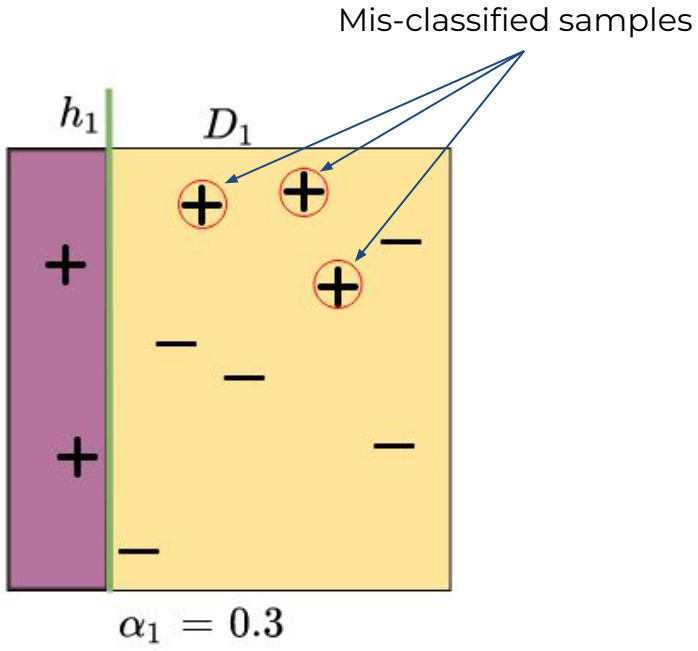
Weight of **correctly classified** samples

AdaBoost Intuition



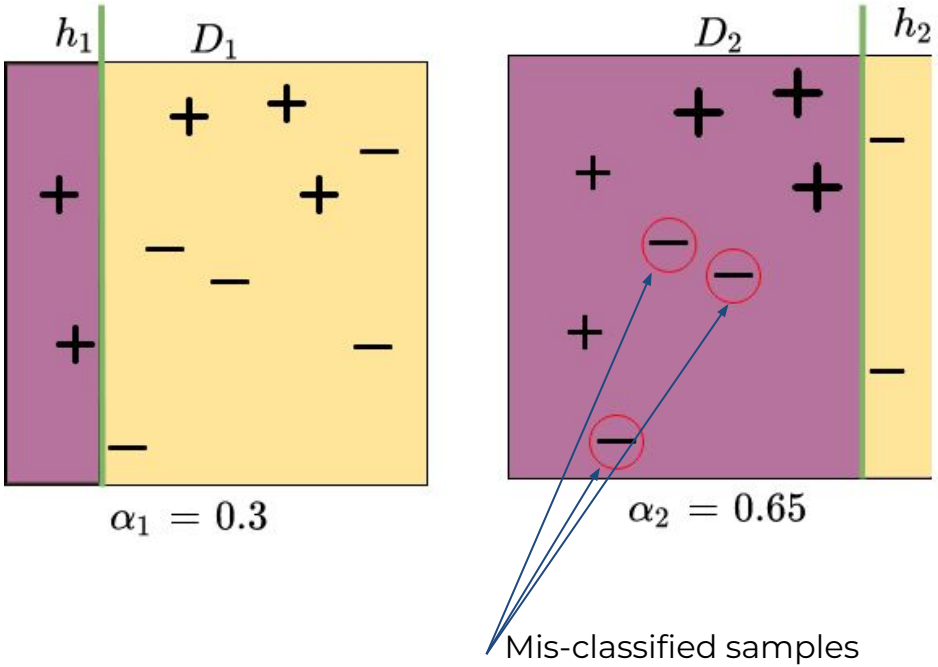
AdaBoost Intuition

First Iteration



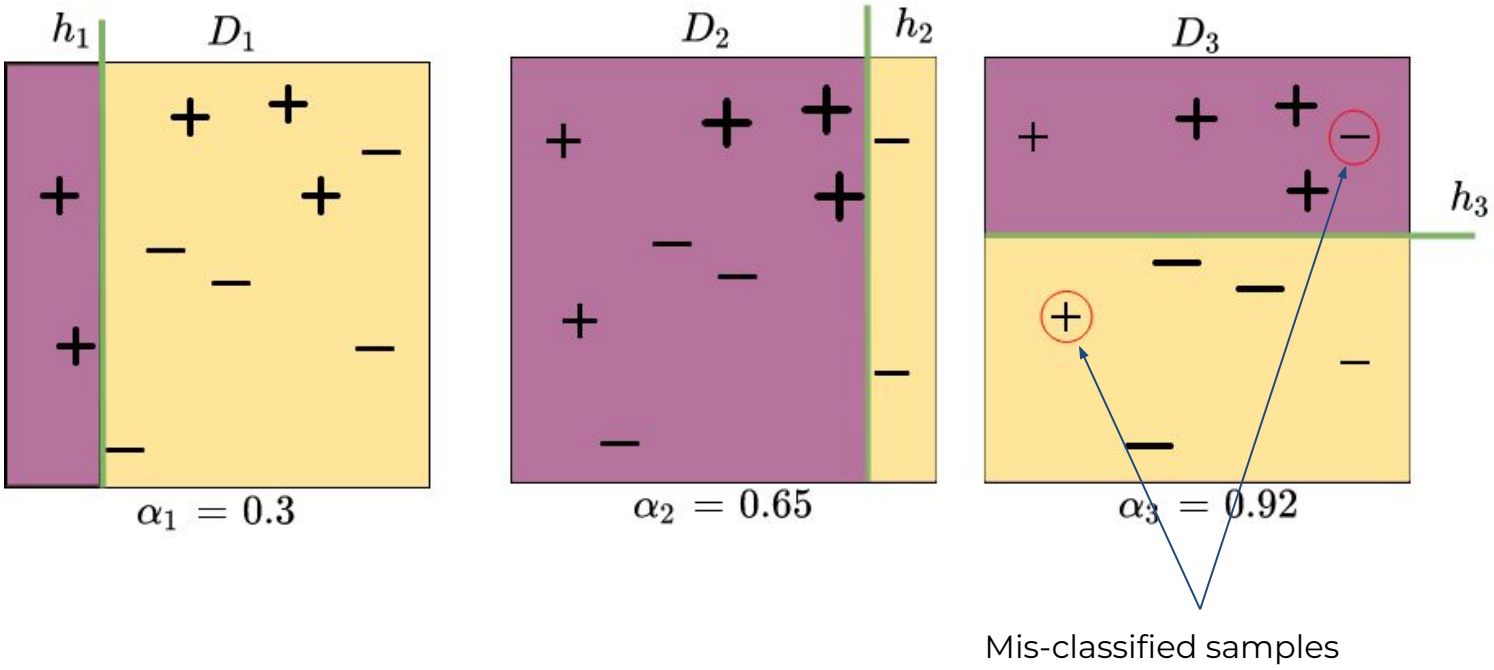
AdaBoost Intuition

Second Iteration

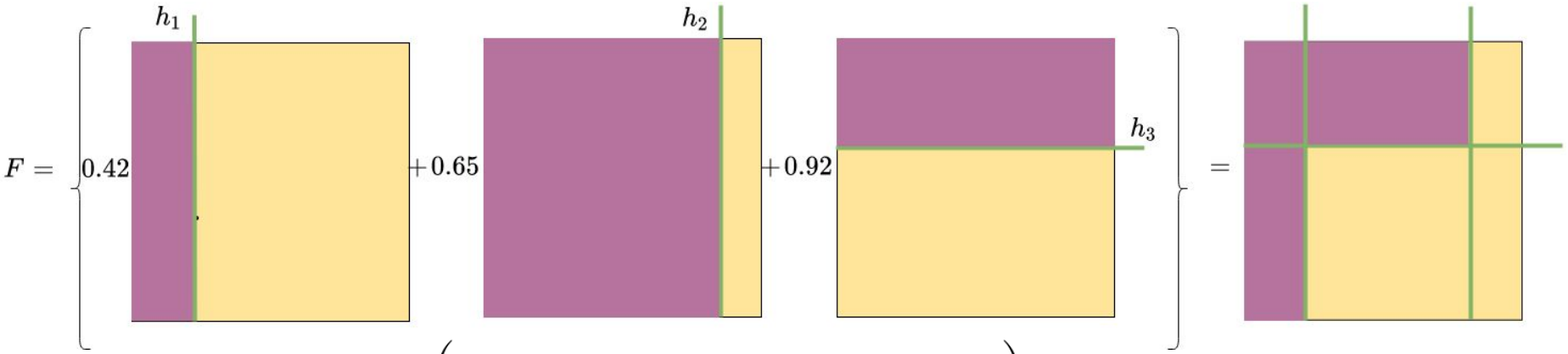


AdaBoost Intuition

Third Iteration



AdaBoost Intuition

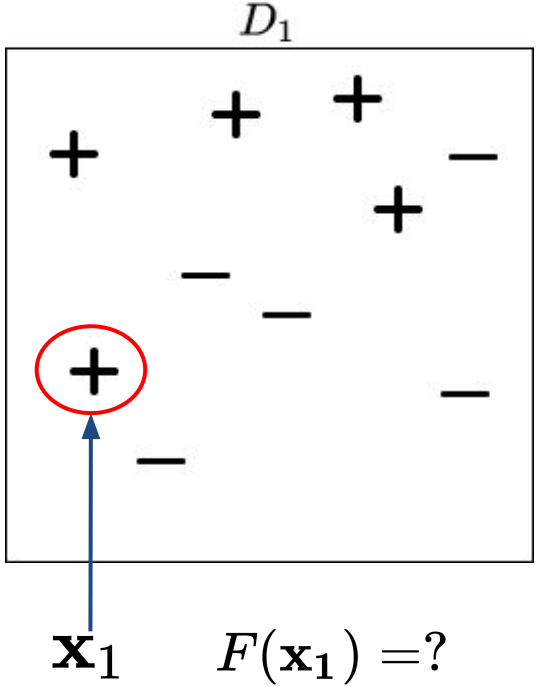


$$F(\mathbf{x}) = \text{sign}\left(0.42 * h_1(\mathbf{x}) + 0.65 * h_2(\mathbf{x}) + 0.92 * h_3(\mathbf{x})\right)$$

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$h_1(\mathbf{x}), h_2(\mathbf{x}), h_3(\mathbf{x}) \in \{1, -1\}$$

Inference in AdaBoost



$$F(\mathbf{x}) = \text{sign}\Big(0.42 * h_1(\mathbf{x}) + 0.65 * h_2(\mathbf{x}) + 0.92 * h_3(\mathbf{x})\Big)$$

$$h_1(\mathbf{x}_1) = 1, h_2(\mathbf{x}_1) = 1, \text{ and } h_3(\mathbf{x}_1) = -1$$

$$\begin{aligned} F(\mathbf{x_1}) &= \text{sign}(0.42 * 1 + 0.65 * 1 + 0.92 * -1) \\ &= \text{sign}(+0.15) \\ &= 1 \end{aligned}$$

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Introduction: Gradient Boosting

Gradient Boosting = **Gradient Descent** + **Boosting**

$$\theta_i := \theta_i - \alpha \frac{\partial J}{\partial \theta_i}$$

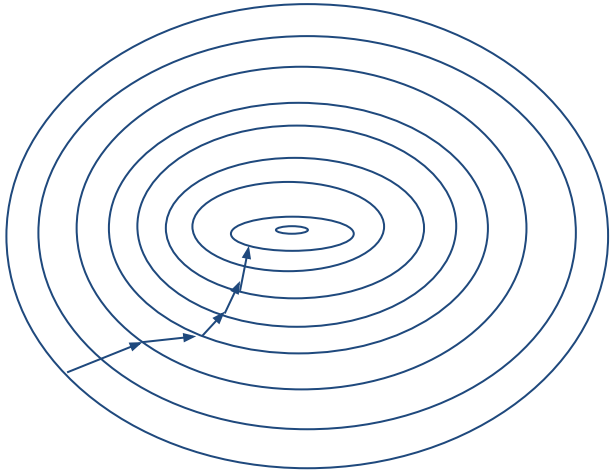


Figure: Gradient Descent

$$\theta_i := \theta_i - \alpha \frac{\partial J}{\partial \theta_i}$$

Vs.

$$F(\mathbf{x}_i) := F(\mathbf{x}_i) - \alpha \frac{\partial J}{\partial F(\mathbf{x}_i)}$$

Numerical Intuition of Gradient Boosting

In gradient boosting, we add another model $h(\mathbf{x})$ to compensate the shortcoming of existing model $F(\mathbf{x})$

$$\underbrace{F(\mathbf{x})}_{\text{existing model}} + \underbrace{h(\mathbf{x})}_{\text{new added model}} = \mathbf{y}$$

$$h(\mathbf{x}) = \mathbf{y} - F(\mathbf{x})$$

This is equivalent to train a model $h(\mathbf{x})$ on data $\left(\mathbf{x}_i, y_i - F(\mathbf{x}_i)\right)_{i=1}^{N}$

The term $\mathbf{y} - F(\mathbf{x})$ is the residue or error

Numerical Intuition of Gradient Boosting

Suppose we fit the model $F(\mathbf{x})$ on dataset $(\mathbf{x}_i, y_i)_{i=1}^{N}$

	y_i	$F(\mathbf{x}_i)$
1	0.8	0.6
2	0.6	0.8
3	1.3	1.4
4	0.5	0.2

Numerical Intuition of Gradient Boosting

	y_i	$F(\mathbf{x}_i)$	**Residue**	$h(\mathbf{x}_i)$
1	0.8	0.6	0.2	0.1
2	0.6	0.8	-0.2	-0.2
3	1.3	1.4	-0.1	0
4	0.5	0.2	0.3	0.2

$$\hat{y}_i = F(\mathbf{x}_i) + h(\mathbf{x}_i)$$

	y_i	$F(\mathbf{x}_i)$	**Residue**	$h(\mathbf{x}_i)$	$\hat{y}_i$
1	0.8	0.6	0.2	0.1	0.7
2	0.6	0.8	-0.2	-0.2	0.6
3	1.3	1.4	-0.1	0	1.4
4	0.5	0.2	0.3	0.2	0.4

New prediction is closer to target, $\hat{y}_i \approx y_i$

$$\hat{y}_i = F(\mathbf{x}_i) + \alpha h(\mathbf{x}_i)$$

Learning rate or shrinkage

How residual is related with the gradient?

Residual of model : $\mathbf{y} - F(\mathbf{x})$

Suppose a square loss function: $J = L(y, F(\mathbf{x})) = \frac{1}{2}(y - F(\mathbf{x}))^2$

Goal : Minimize loss function $J = \sum_i L(y_i, F(\mathbf{x}_i))$ by adjusting $F(\mathbf{x}_1), F(\mathbf{x}_2), \ldots, F(\mathbf{x}_N)$

Taking derivative:

$$\frac{\partial J}{\partial F(\mathbf{x_i})} = \frac{\partial L(y_i, F(\mathbf{x_i}))}{\partial F(\mathbf{x_i})} = F(\mathbf{x_i}) - y_i = -\big(y_i - F(\mathbf{x_i})\big)$$

$$\underbrace{y_i - F(\mathbf{x}_i)}_{\text{residual}} = \underbrace{-\frac{\partial J}{\partial F(\mathbf{x_i})}}_{\text{negative gradient}}$$

Gradient Boosting for Classification

Logits to probability:

For binary classification: **Sigmoid function**

$$\hat{p} = \frac{1}{1+e^{-\hat{y}}} = \frac{e^{\hat{y}}}{1+e^{\hat{y}}}$$

For multi-class classification: **Softmax function**

$$\hat{p}_i = f(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{C} e^{a_j}}$$

a_i is the value of logit for i^{th} class.

Data	Setosa a_1	Virginica a_2	Versicolor a_3
$\mathbf{x}_1$	2.3	0.4	1.2

$f(a_i)$ →

Data	Setosa $\hat{p}_1$	Virginica $\hat{p}_2$	Versicolor $\hat{p}_3$
$\mathbf{x}_1$	0.67	0.10	0.23

Gradient Boosting for Classification

Problem: Recognize the hand-written capital letter.

- Multi-class classification
- 26 classes. A..Z

Model

- 26 score functions (our models): $F_A, F_B, F_C, ..., F_Z$.
- $F_A(x)$ assigns a score for class A
- scores are used to calculate probabilities

$$P_A(x) = \frac{e^{F_A(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

$$P_B(x) = \frac{e^{F_B(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

...

$$P_Z(x) = \frac{e^{F_Z(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

- predicted label = class that has the highest probability

Remember, we are fitting regression trees even for classification problems. Therefore, 1 tree corresponds to 1 class.

Slides 81-89: Cheng Li A Gentle Introduction to Gradient Boosting
https://www.khoury.northeastern.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Gradient Boosting for Classification

Loss Function for each data point

Step 1 turn the label y_i into a (true) probability distribution $Y_c(x_i)$
For example: y_5=G,
$Y_A(x_5) = 0, Y_B(x_5) = 0, ..., Y_G(x_5) = 1, ..., Y_Z(x_5) = 0$

Step 2 calculate the predicted probability distribution $P_c(x_i)$ based on the current model $F_A, F_B, ..., F_Z$.
$P_A(x_5) = 0.03, P_B(x_5) = 0.05, ..., P_G(x_5) = 0.3, ..., P_Z(x_5) = 0.05$

Step 3 calculate the difference between the true probability distribution and the predicted probability distribution.

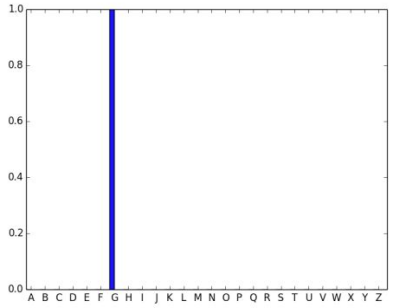


Figure: true probability distribution

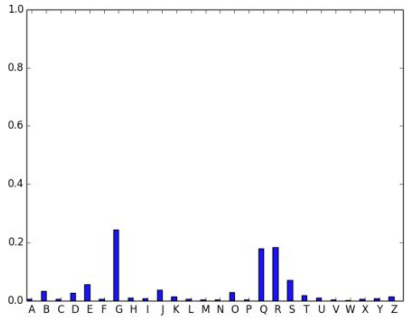


Figure: predicted probability distribution based on current model

Gradient Boosting for Classification

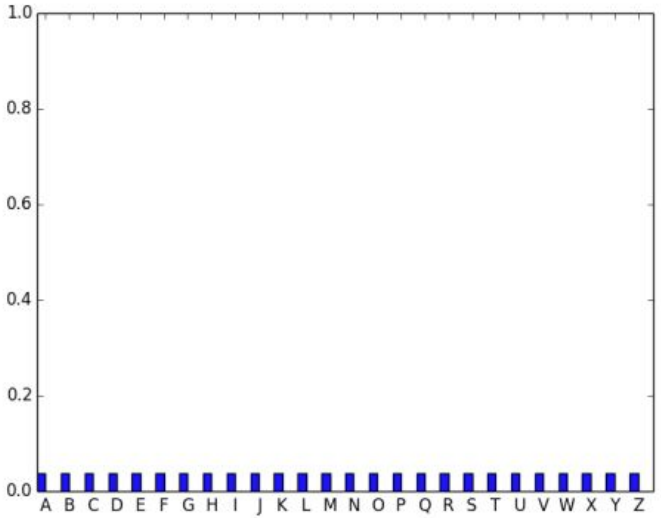


Figure: predicted probability distribution at round 0

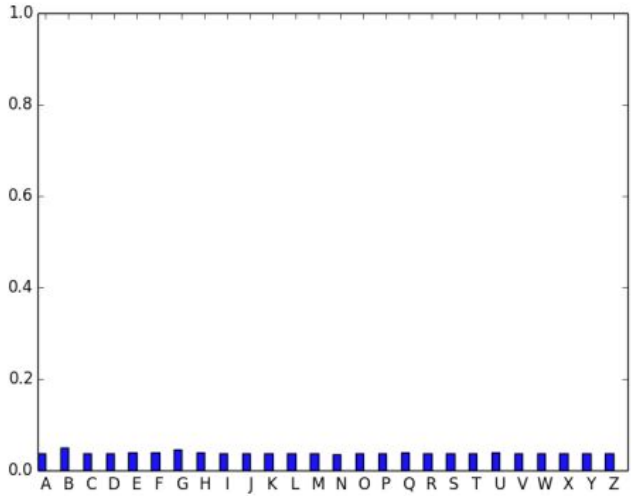


Figure: predicted probability distribution at round 1

Gradient Boosting for Classification

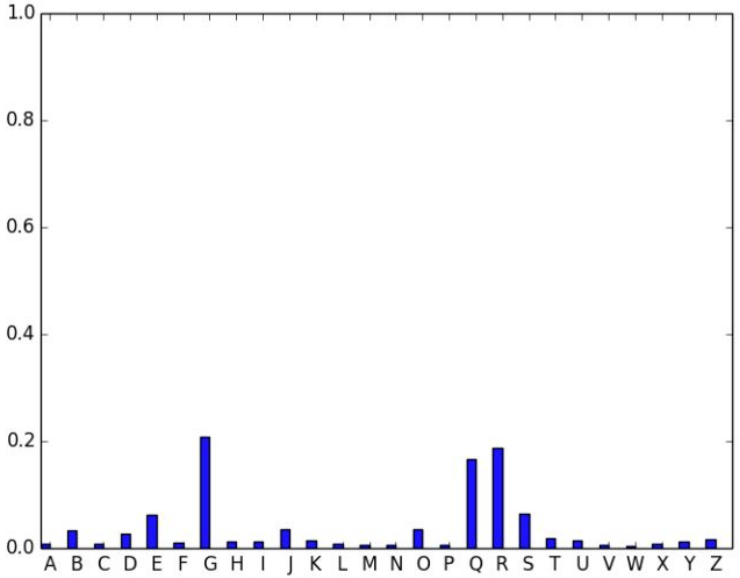


Figure: predicted probability distribution at round 30

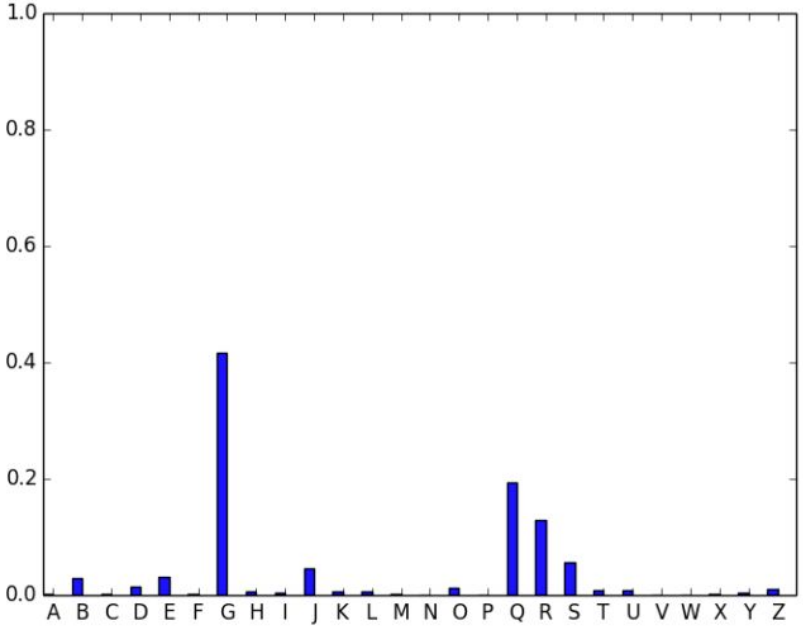


Figure: predicted probability distribution at round 100

Gradient Boosting for Classification

start with initial models $F_A, F_B, F_C, ..., F_Z$
iterate until converge:
calculate negative gradients for class A: $-g_A(x_i) = Y_A(x_i) - P_A(x_i)$
calculate negative gradients for class B: $-g_B(x_i) = Y_B(x_i) - P_B(x_i)$
...
calculate negative gradients for class Z:$-g_Z(x_i) = Y_Z(x_i) - P_Z(x_i)$
fit a regression tree h_A to negative gradients $-g_A(x_i)$
fit a regression tree h_B to negative gradients $-g_B(x_i)$
...
fit a regression tree h_Z to negative gradients $-g_Z(x_i)$
$F_A := F_A + \rho_A h_A$
$F_B := F_A + \rho_B h_B$
...
$F_Z := F_A + \rho_Z h_Z$

Gradient Boosting for Classification

round 0

i	y	Y_A	Y_B	Y_C	Y_D	Y_E	Y_F	Y_G	Y_H	Y_I	Y_J	Y_K	Y_L	Y_M	Y_N	Y_O	Y_P	Y_Q	Y_R	Y_S	Y_T	Y_U	Y_V	Y_W	Y_X	Y_Y	Y_Z
1	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	N	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	G	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	F_A	F_B	F_C	F_D	F_E	F_F	F_G	F_H	F_I	F_J	F_K	F_L	F_M	F_N	F_O	F_P	F_Q	F_R	F_S	F_T	F_U	F_V	F_W	F_X	F_Y	F_Z
1	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	P_A	P_B	P_C	P_D	P_E	P_F	P_G	P_H	P_I	P_J	P_K	P_L	P_M	P_N	P_O	P_P	P_Q	P_R	P_S	P_T	P_U	P_V	P_W	P_X	P_Y	P_Z
1	T	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
2	I	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
3	D	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
4	N	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
5	G	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	$Y_A - P_A$	$Y_B - P_B$	$Y_C - P_C$	$Y_D - P_D$	$Y_E - P_E$	$Y_F - P_F$	$Y_G - P_G$	$Y_H - P_H$	$Y_I - P_I$	$Y_J - P_J$	$Y_K - P_K$	$Y_L - P_L$	$Y_M - P_M$	$Y_N - P_N$	$Y_O - P_O$	$Y_P - P_P$	$Y_Q - P_Q$	$Y_R - P_R$	$Y_S - P_S$	$Y_T - P_T$	$Y_U - P_U$	$Y_V - P_V$	$Y_W - P_W$	$Y_X - P_X$	$Y_Y - P_Y$	$Y_Z - P_Z$
1	T	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
2	I	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
3	D	-0.04	-0.04	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
4	N	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
5	G	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Gradient Boosting for Classification

round 1

i	y	Y_A	Y_B	Y_C	Y_D	Y_E	Y_F	Y_G	Y_H	Y_I	Y_J	Y_K	Y_L	Y_M	Y_N	Y_O	Y_P	Y_Q	Y_R	Y_S	Y_T	Y_U	Y_V	Y_W	Y_X	Y_Y	Y_Z
1	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	N	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	G	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	F_A	F_B	F_C	F_D	F_E	F_F	F_G	F_H	F_I	F_J	F_K	F_L	F_M	F_N	F_O	F_P	F_Q	F_R	F_S	F_T	F_U	F_V	F_W	F_X	F_Y	F_Z
1	T	-0.08	-0.07	-0.06	-0.07	-0.02	-0.02	-0.08	-0.02	-0.03	-0.03	-0.06	-0.04	-0.08	-0.08	-0.07	-0.07	-0.02	-0.04	-0.04	0.59	-0.01	-0.07	-0.07	-0.05	-0.06	-0.07
2	I	-0.08	0.23	-0.06	-0.07	-0.02	-0.02	0.16	-0.02	-0.03	-0.03	-0.06	-0.04	-0.08	-0.08	-0.07	-0.07	-0.02	-0.04	-0.04	-0.07	-0.01	-0.07	-0.07	-0.05	-0.06	-0.07
3	D	-0.08	0.23	-0.06	-0.07	-0.02	-0.02	-0.08	-0.02	-0.03	-0.03	-0.06	-0.04	-0.08	-0.08	-0.07	-0.07	-0.02	-0.04	-0.04	-0.07	-0.01	-0.07	-0.07	-0.05	-0.06	-0.07
4	N	-0.08	-0.07	-0.06	-0.07	-0.02	-0.02	0.16	-0.02	-0.03	-0.03	0.26	-0.04	-0.08	0.3	-0.07	-0.07	-0.02	-0.04	-0.04	-0.07	-0.01	-0.07	-0.07	-0.05	-0.06	-0.07
5	G	-0.08	0.23	-0.06	-0.07	-0.02	-0.02	0.16	-0.02	-0.03	-0.03	-0.06	-0.04	-0.08	-0.08	-0.07	-0.07	-0.02	-0.04	-0.04	-0.07	-0.01	-0.07	-0.07	-0.05	-0.06	-0.07
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	P_A	P_B	P_C	P_D	P_E	P_F	P_G	P_H	P_I	P_J	P_K	P_L	P_M	P_N	P_O	P_P	P_Q	P_R	P_S	P_T	P_U	P_V	P_W	P_X	P_Y	P_Z
1	T	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.07	0.04	0.04	0.04	0.04	0.04	0.04
2	I	0.04	0.05	0.04	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
3	D	0.04	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
4	N	0.04	0.04	0.04	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.04	0.04	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
5	G	0.04	0.05	0.04	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	$Y_A - P_A$	$Y_B - P_B$	$Y_C - P_C$	$Y_D - P_D$	$Y_E - P_E$	$Y_F - P_F$	$Y_G - P_G$	$Y_H - P_H$	$Y_I - P_I$	$Y_J - P_J$	$Y_K - P_K$	$Y_L - P_L$	$Y_M - P_M$	$Y_N - P_N$	$Y_O - P_O$	$Y_P - P_P$	$Y_Q - P_Q$	$Y_R - P_R$	$Y_S - P_S$	$Y_T - P_T$	$Y_U - P_U$	$Y_V - P_V$	$Y_W - P_W$	$Y_X - P_X$	$Y_Y - P_Y$	$Y_Z - P_Z$
1	T	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	0.93	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
2	I	-0.04	-0.05	-0.04	-0.04	-0.04	-0.04	-0.05	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
3	D	-0.04	-0.05	-0.04	0.96	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
4	N	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.05	-0.04	-0.04	-0.04	-0.05	-0.04	-0.04	0.95	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
5	G	-0.04	-0.05	-0.04	-0.04	-0.04	-0.04	0.95	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Gradient Boosting for Classification

round 100

i	y	Y_A	Y_B	Y_C	Y_D	Y_E	Y_F	Y_G	Y_H	Y_I	Y_J	Y_K	Y_L	Y_M	Y_N	Y_O	Y_P	Y_Q	Y_R	Y_S	Y_T	Y_U	Y_V	Y_W	Y_X	Y_Y	Y_Z
1	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	N	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	G	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	F_A	F_B	F_C	F_D	F_E	F_F	F_G	F_H	F_I	F_J	F_K	F_L	F_M	F_N	F_O	F_P	F_Q	F_R	F_S	F_T	F_U	F_V	F_W	F_X	F_Y	F_Z
1	T	-3.26	-2.7	-2.2	-2.22	-2.48	-0.31	-2.77	-1.19	2.77	0.1	-1.49	-1.02	-1.64	-0.8	-2.4	-3.57	-0.9	-2.45	-0.2	4.61	0.5	-0.71	-1.21	-0.24	0.49	-1.66
2	I	-1.64	-1.09	-2.29	-1.8	0.45	-0.43	2.14	-1.56	1.19	1.09	-1.5	-0.5	-3.64	-3.98	-0.39	-2.3	1.42	-0.59	0.27	-2.88	-1.96	-1.67	-4.38	-2.06	-2.95	-1.76
3	D	-2.45	0.18	-3.01	0.18	-2.79	-1.7	-2.21	0.43	-1.12	0.32	0.67	-2.16	-2.91	-2.76	-1.92	-3.04	-1.47	-0.48	-1.48	-1.25	-2.25	-3.23	-4.38	0.17	-2.95	-2.65
4	N	-3.95	-3.38	-0.22	-0.94	-1.33	-1.38	-1.22	-0.12	-2.33	-3.13	0.58	-0.65	-0.25	2.96	-2.84	-1.82	0.19	0.55	-1.22	-1.25	0.45	-1.8	0.11	-0.69	-1.6	-3.78
5	G	-3.14	-0.04	-2.37	-0.78	0.02	-2.68	2.6	-1.48	-1.93	0.42	-1.44	-1.45	-3.36	-3.98	-0.94	-3.42	1.84	1.44	0.62	-1.25	-1.33	-4.41	-4.71	-2.62	-2.15	-1.09
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

i	y	P_A	P_B	P_C	P_D	P_E	P_F	P_G	P_H	P_I	P_J	P_K	P_L	P_M	P_N	P_O	P_P	P_Q	P_R	P_S	P_T	P_U	P_V	P_W	P_X	P_Y	P_Z
1	T	0	0	0	0	0	0.01	0	0	0.13	0.01	0	0	0	0	0	0	0	0	0.01	0.79	0.01	0	0	0.01	0.01	0
2	I	0.01	0.01	0	0.01	0.06	0.02	0.32	0.01	0.12	0.11	0.01	0.02	0	0	0.03	0	0.16	0.02	0.05	0	0.01	0.01	0	0	0	0.01
3	D	0.01	0.11	0	0.11	0.01	0.02	0.01	0.14	0.03	0.12	0.17	0.01	0	0.01	0.01	0	0.02	0.05	0.02	0.03	0.01	0	0	0.11	0	0.01
4	N	0	0	0.02	0.01	0.01	0.01	0.01	0.03	0	0	0.05	0.02	0.02	0.59	0	0	0.04	0.05	0.01	0.01	0.05	0.01	0.03	0.02	0.01	0
5	G	0	0.03	0	0.01	0.03	0	0.42	0.01	0	0.05	0.01	0.01	0	0	0.01	0	0.19	0.13	0.06	0.01	0.01	0	0	0	0	0.01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

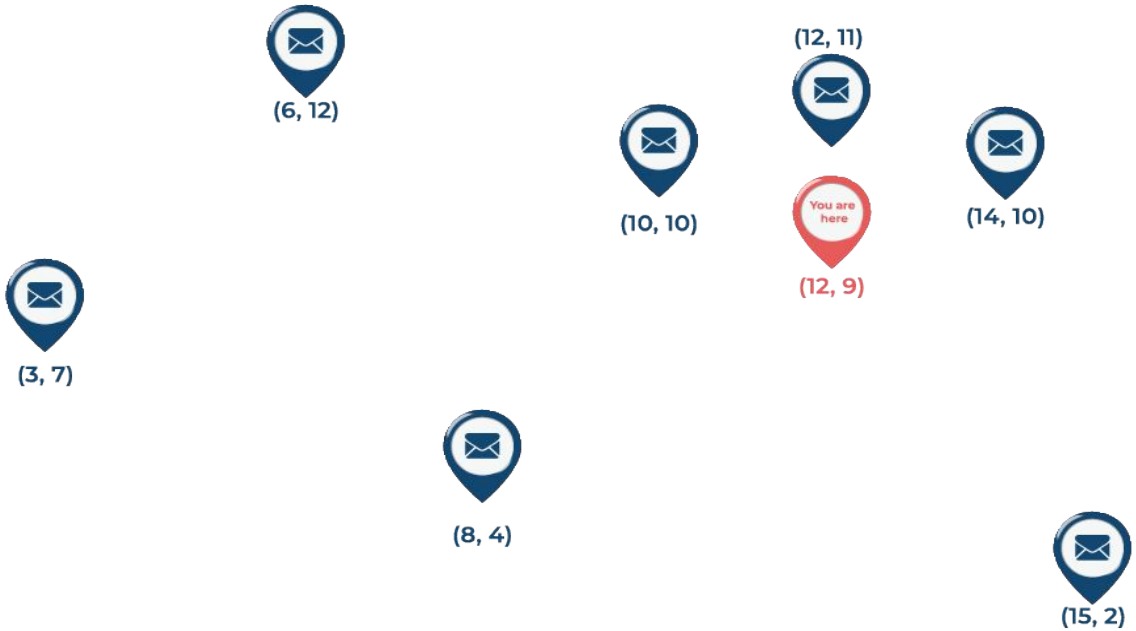
i	y	$Y_A - P_A$	$Y_B - P_B$	$Y_C - P_C$	$Y_D - P_D$	$Y_E - P_E$	$Y_F - P_F$	$Y_G - P_G$	$Y_H - P_H$	$Y_I - P_I$	$Y_J - P_J$	$Y_K - P_K$	$Y_L - P_L$	$Y_M - P_M$	$Y_N - P_N$	$Y_O - P_O$	$Y_P - P_P$	$Y_Q - P_Q$	$Y_R - P_R$	$Y_S - P_S$	$Y_T - P_T$	$Y_U - P_U$	$Y_V - P_V$	$Y_W - P_W$	$Y_X - P_X$	$Y_Y - P_Y$	$Y_Z - P_Z$
1	T	-0	-0	-0	-0	-0	-0.01	-0	-0	-0.13	-0.01	-0	-0	-0	-0	-0	-0	-0	-0	-0.01	0.21	-0.01	-0	-0	-0.01	-0.01	-0
2	I	-0.01	-0.01	-0	-0.01	-0.06	-0.02	-0.32	-0.01	0.88	-0.11	-0.01	-0.02	-0	-0	-0.03	-0	-0.16	-0.02	-0.05	-0	-0.01	-0.01	-0	-0	-0	-0.01
3	D	-0.01	-0.11	-0	0.89	-0.01	-0.02	-0.01	-0.14	-0.03	-0.12	-0.17	-0.01	-0	-0.01	-0.01	-0	-0.02	-0.05	-0.02	-0.03	-0.01	-0	-0	-0.11	-0	-0.01
4	N	-0	-0	-0.02	-0.01	-0.01	-0.01	-0.01	-0.03	-0	-0	-0.05	-0.02	-0.02	0.41	-0	-0	-0.04	-0.05	-0.01	-0.01	-0.05	-0.01	-0.03	-0.02	-0.01	-0
5	G	-0	-0.03	-0	-0.01	-0.03	-0	0.58	-0.01	-0	-0.05	-0.01	-0.01	-0	-0	-0.01	-0	-0.19	-0.13	-0.06	-0.01	-0.01	-0	-0	-0	-0	-0.01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Boosted Decision Trees

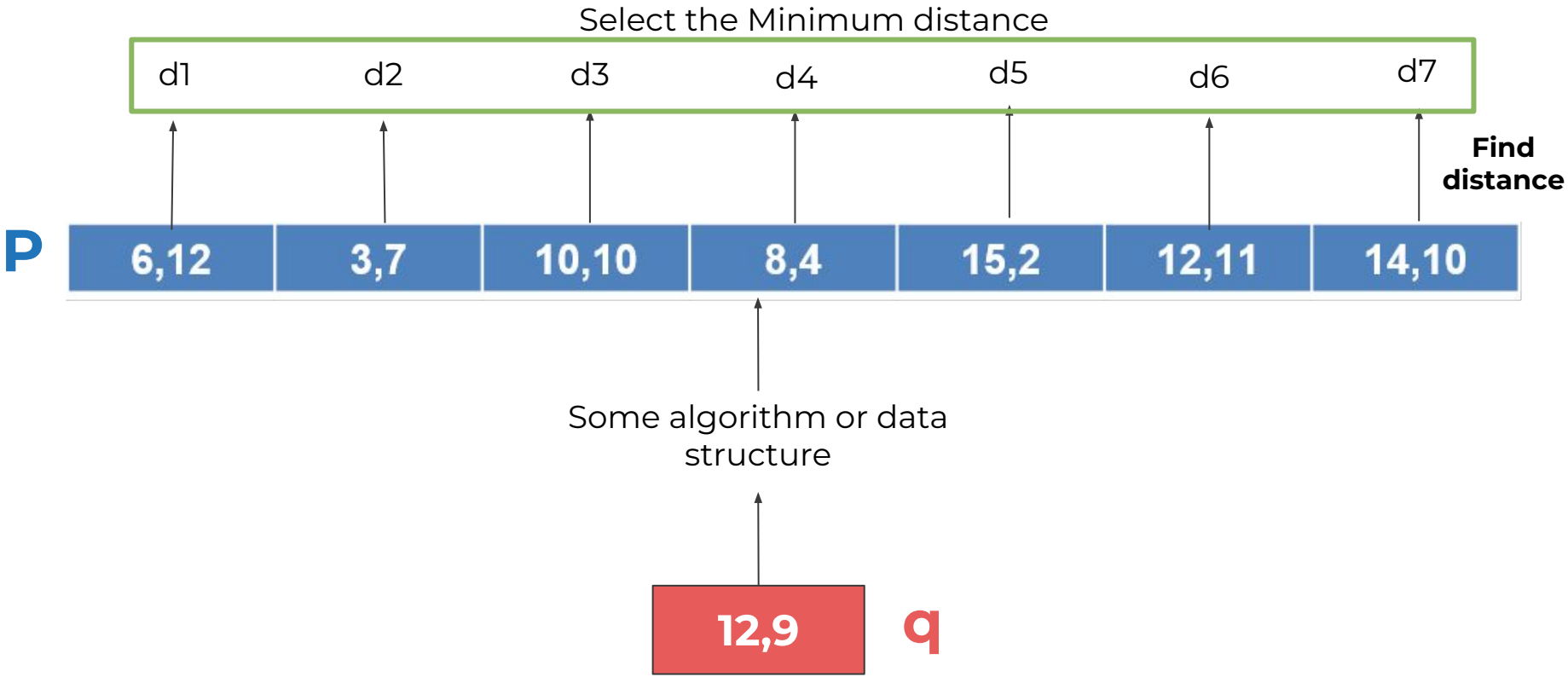
Fusemachines
AI School

K-Nearest Neighbors (KNN)

Which post office would you choose to send a letter to your friend?



Nearest neighbor search

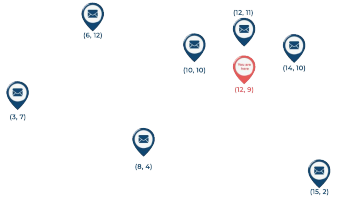


Source: Shakhnarovich, G., Darrell, T., & Indyk, P. (2005). Nearest-neighbor methods in learning and vision. In *Neural Information Processing*.

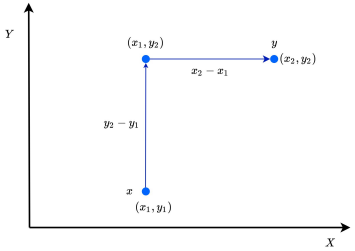
Brute force algorithm

Source: Shakhnarovich, G., Darrell, T., & Indyk, P. (2005). Nearest-neighbor methods in learning and vision. In *Neural Information Processing*.

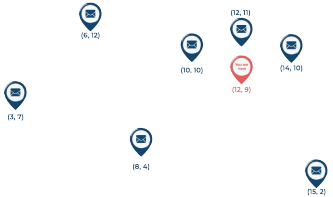
The **Learning objective**



1. **Distance and its properties**



2. **Types of distance metrics**



3. **Solve post office problem**

Definition of Distance or dissimilarity

$$X = \{1, 2, 3, \ldots . n\}$$

A function $d : X \times X \longrightarrow \mathbb{R}$ is called a distance on X if, for all $x, y \in X$

There holds:

1. $d(x,y) \geq 0$ (non-negativity)
2. $d(x,y) = 0$ if and only if $x = y$ (identity of indiscernibles)
3. $d(x,y) = d(y,x)$ (symmetry)

Distance metric: Distance is a result of the distance function or distance metric

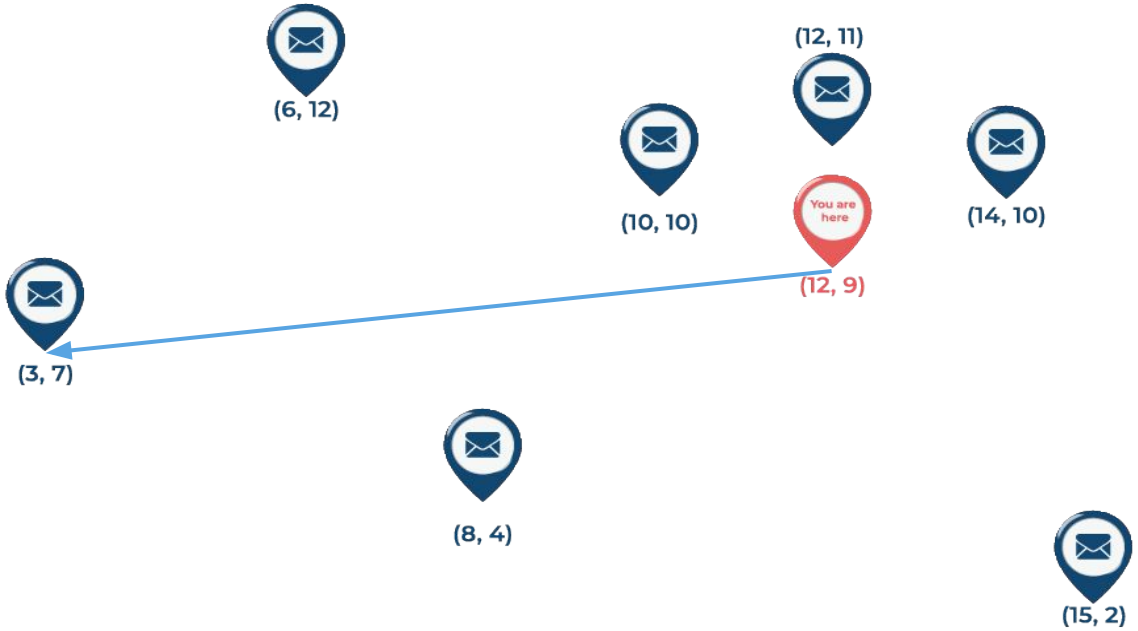
$$X = \{1, 2, 3, \ldots . n\}$$

A function $d : X \times X \longrightarrow \mathbb{R}$ is called a metric on X if, for all $x, y, z \in X$

There holds:

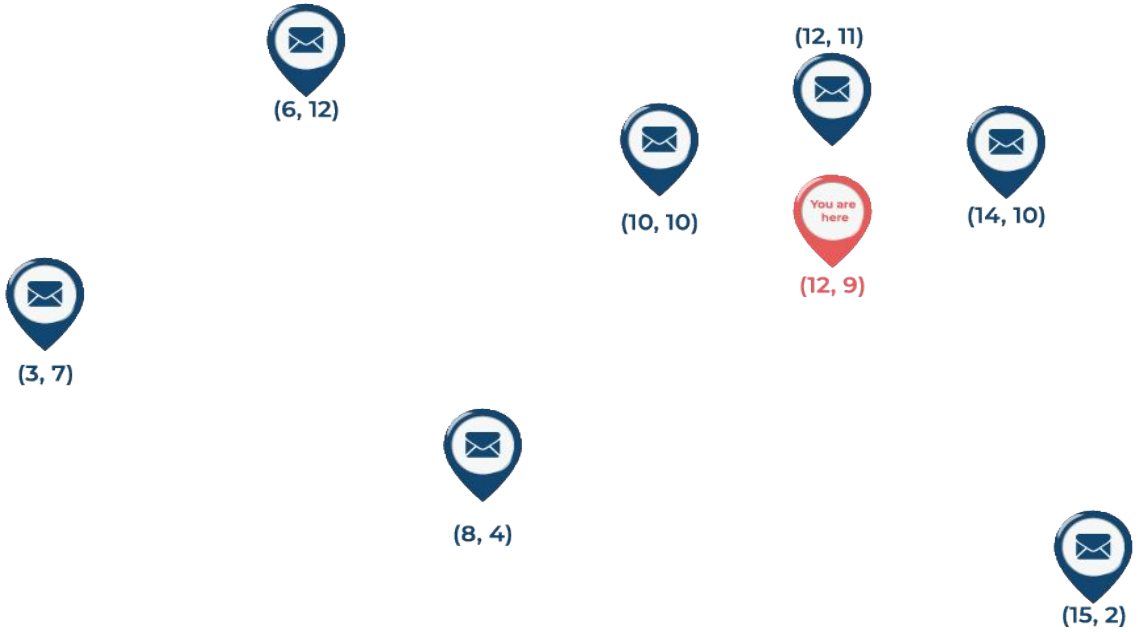
1. $d(x,y) \geq 0$ (non-negativity)
2. $d(x,y) = 0$ if and only if $x = y$ (identity of indiscernibles)
3. $d(x,y) = d(y,x)$ (symmetry)
4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality)

Non-negative: If you travel from (12,9) to (3,7) the distance covered is positive



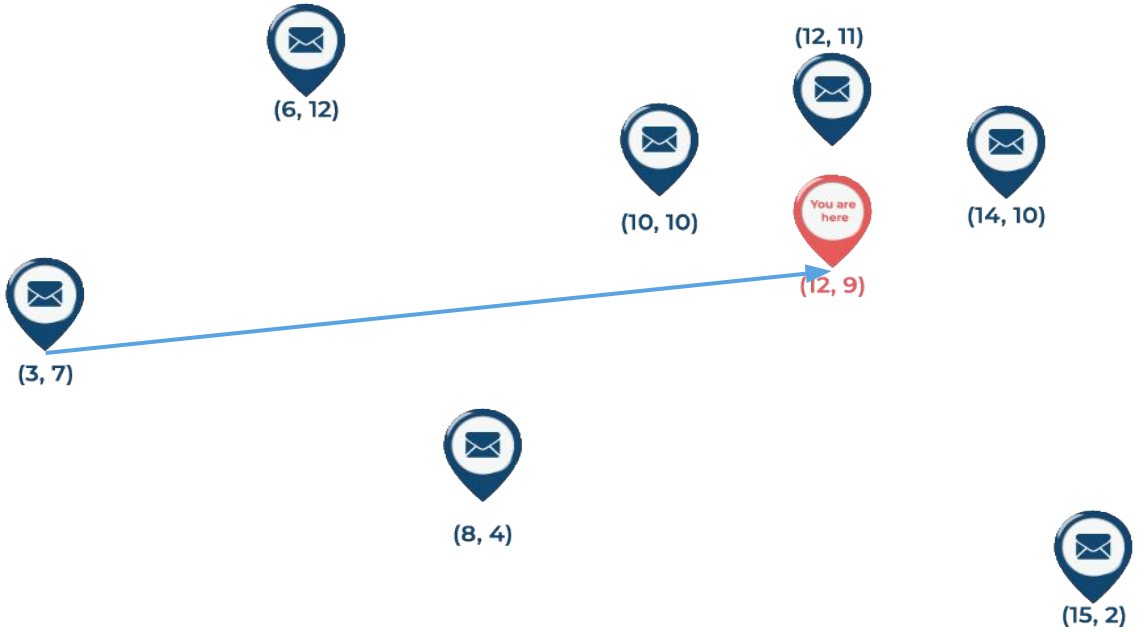
The distance between the two points is always positive.

Identity: If you don't travel from (12,9) the distance covered is zero



The distance between two identical point is always zero

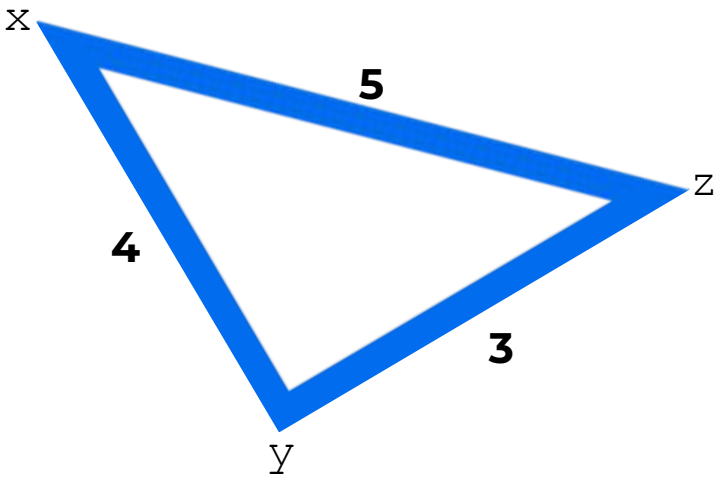
Symmetry: If you travel from (3,7) to (12,9) the distance covered is same as from (12,9) to (3,7)



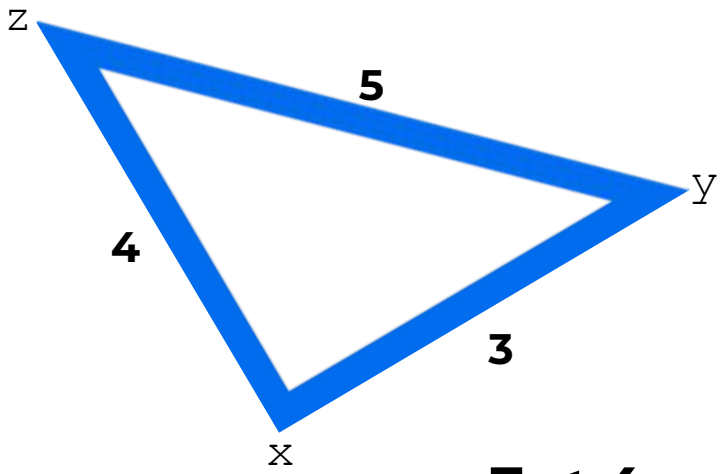
Distance must remain equal if you calculate it from either point.

Triangle Inequality: The sum of length of any two side of triangle is greater than the length of third side

$$d(x,y) \leq d(x,z) + d(z,y)$$



4 ≤ 5 + 3



3 ≤ 4 + 5

Distance metric types

Minkowski Distance

Minkowski distance for x,y ∈ R^n is given as

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/r}$$

p = r	Distance Metric
1	Manhattan
2	Euclidean
∞	Chebyshev

Manhattan distance between **x, y** in *n*-dimensional space is the sum of the distances in each dimension.

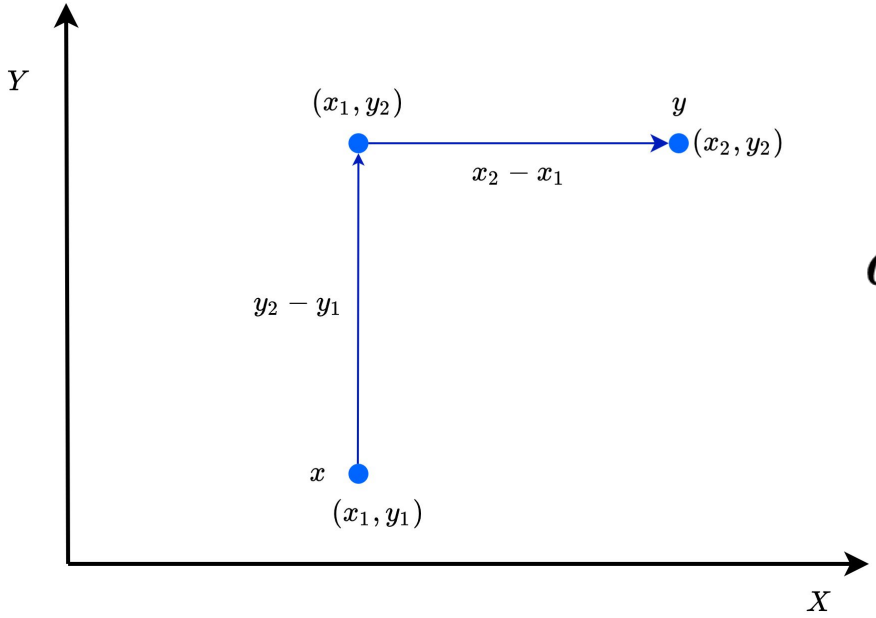
$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/r}$$

p = r = 1

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^1\right)^{1/1}$$

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

Manhattan distance:



$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} | x_i - y_i |$$

$$d(x, y) = |x_2 - x_1| + |y_2 - y_1|$$

Suppose x = (3,1) and y = (12,9)

d(x,y) = |12-3| + |9-1|
d(x,y) = 17

Euclidean distance between two points in Euclidean space is the length of a line segment between the two points

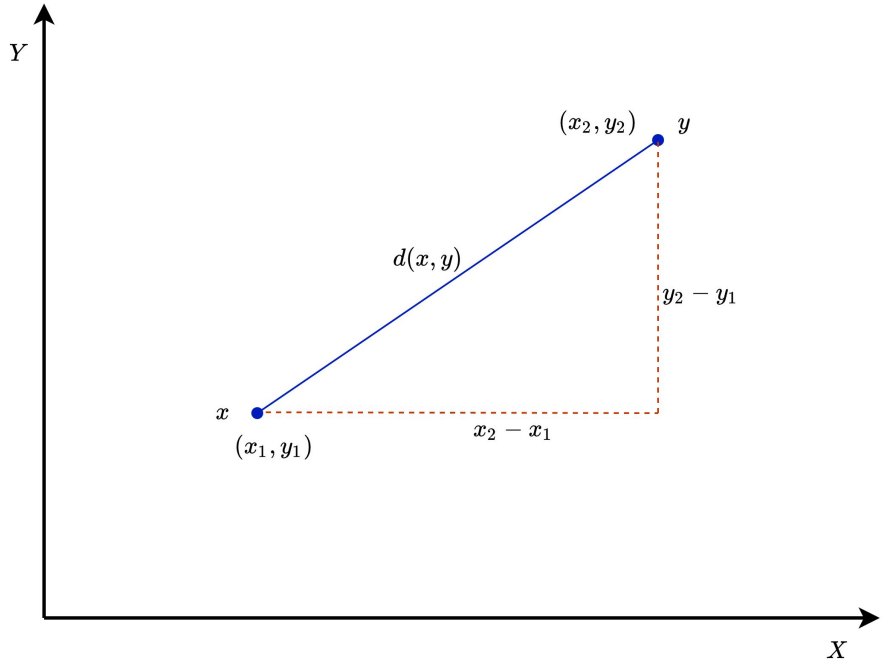
$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/r}$$

p = r = 2

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{1/2}$$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean distance



$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

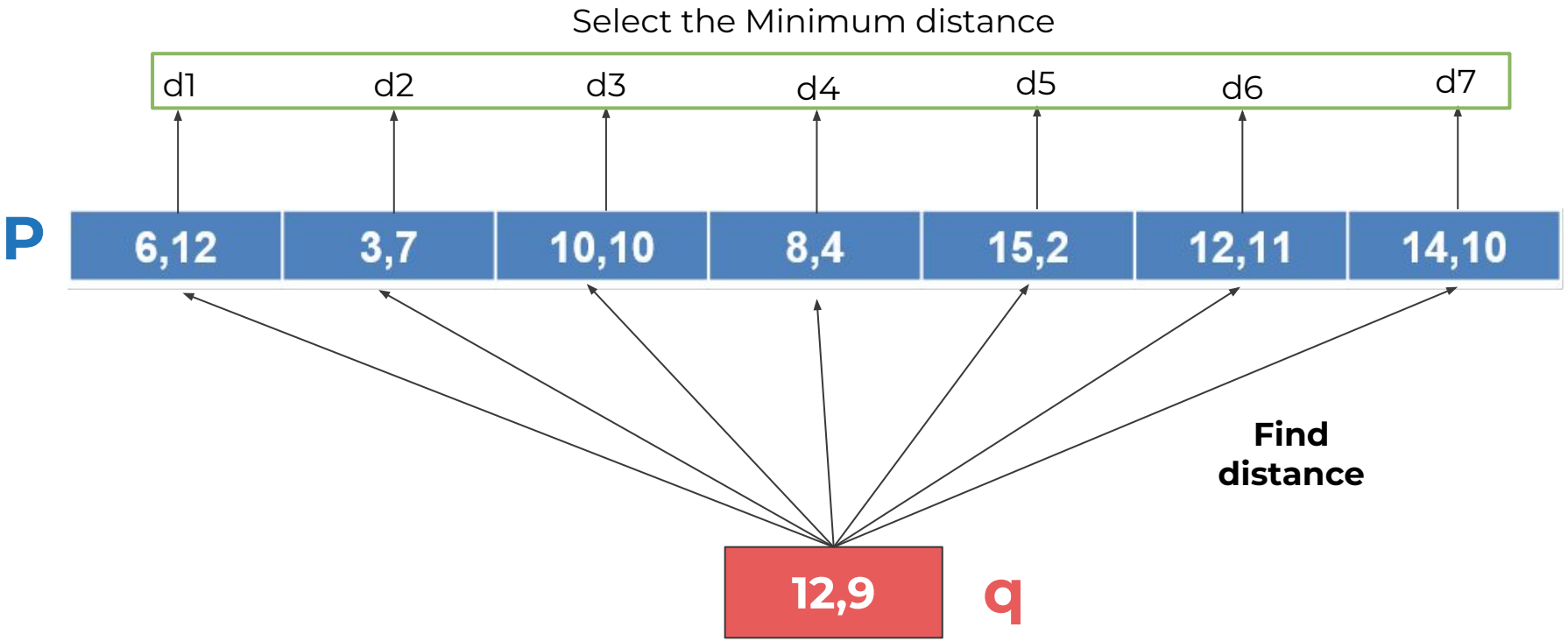
Suppose x = (3,1) and y = (12,9)

$$d(x,y) = \sqrt{(12-3)^2 + (9-1)^2}$$

$$d(x,y) = \sqrt{145}$$

Native-Bruteforce method: Nearest neighbor search on Post office problem

Nearest neighbor search and brute-force algorithm



Source: Shakhnarovich, G., Darrell, T., & Indyk, P. (2005). Nearest-neighbor methods in learning and vision. In *Neural Information Processing*.

Calculate the distance between your position and each post office location using Manhattan distance

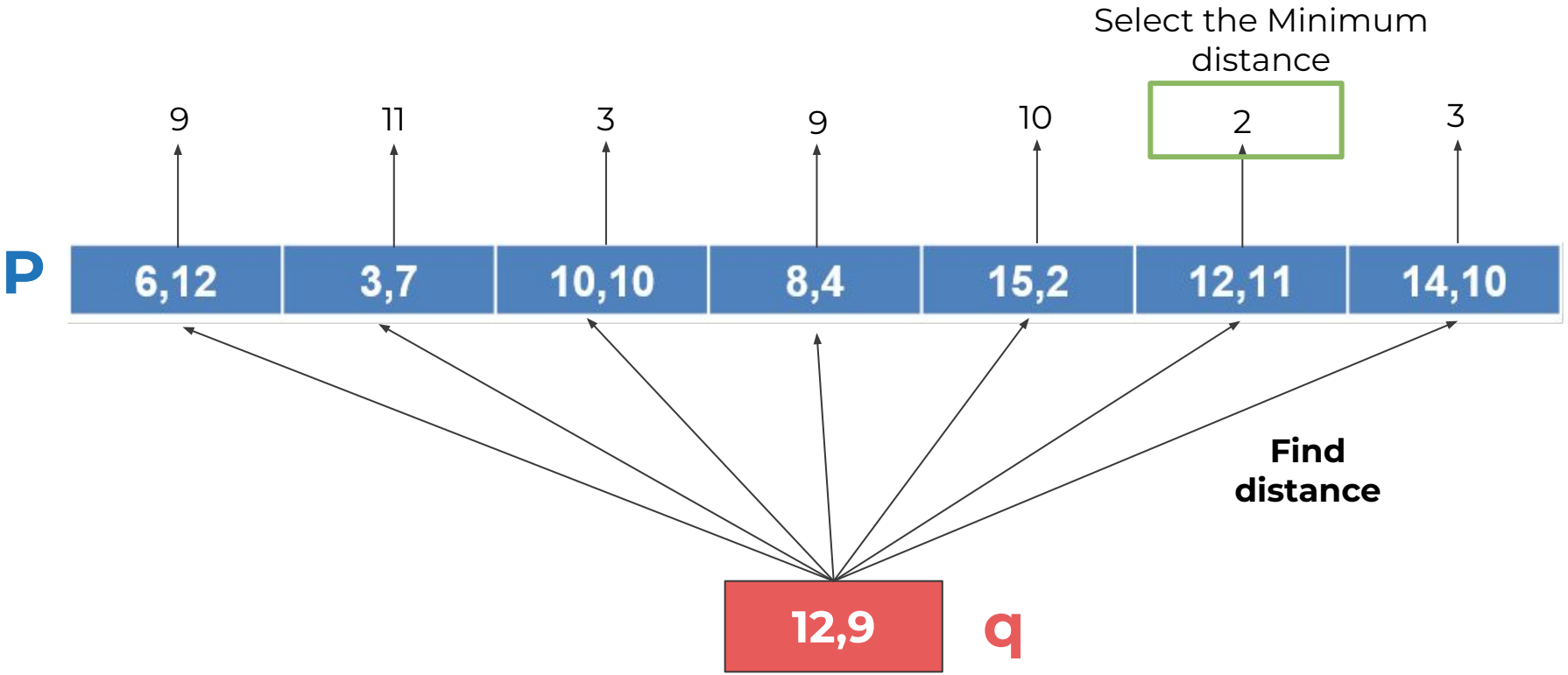
6,12	3,7	10,10	8,4	15,2	12,11	14,10

$$d(x, y) = |x_2 - x_1| + |y_2 - y_1|$$

Your position	Post office	Distance
(12,9)	(6,12)	9
(12,9)	(3,7)	11
(12,9)	(10,10)	3
(12,9)	(8,4)	9
(12,9)	(15,2)	10
(12,9)	(12,11)	2
(12,9)	(14,10)	3

Select the Minimum distance

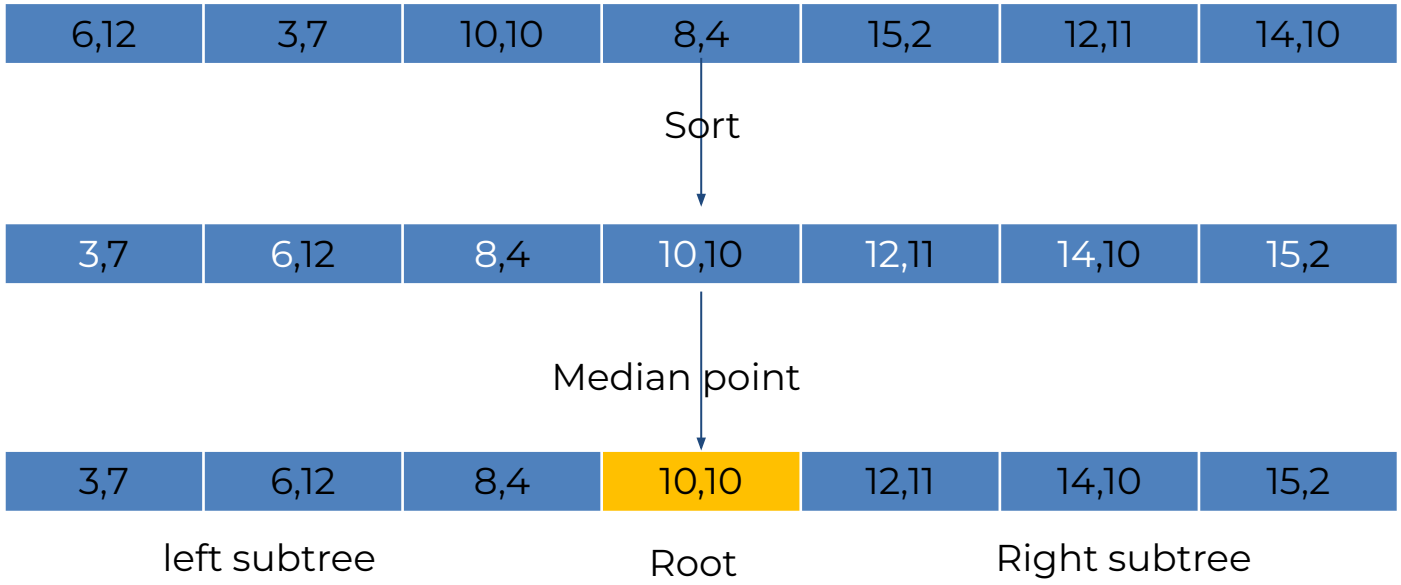
Complexity O(nd): brute-force computation of distances between all pairs of points in the dataset



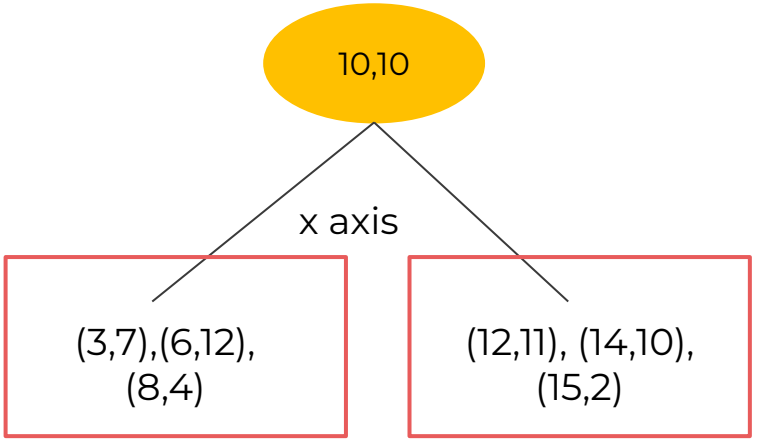
If number of samples (n) in a list grows, the brute-force approach quickly becomes infeasible

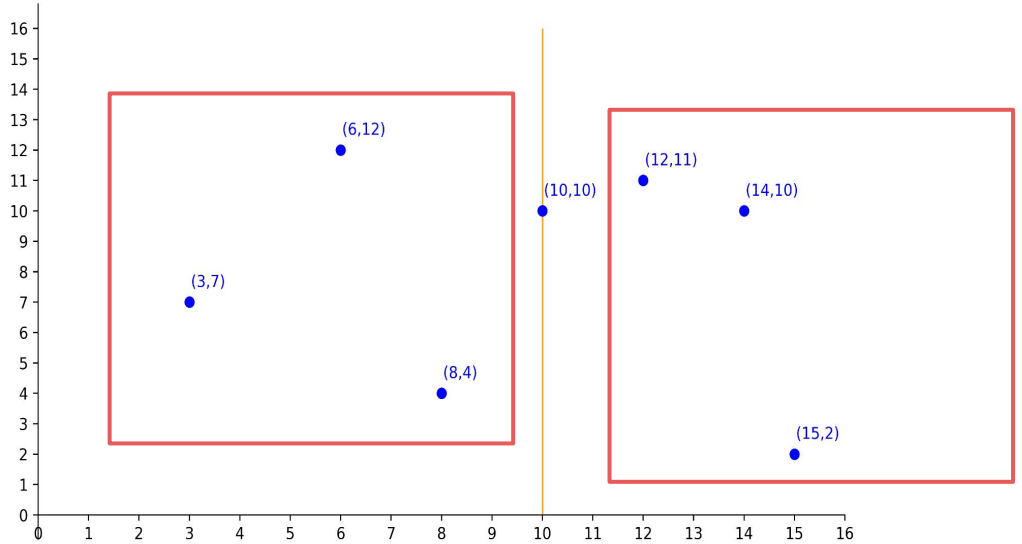
The KD Tree construction algorithm

Sort the data points using *x* coordinate value; find and mark the median point as the root node

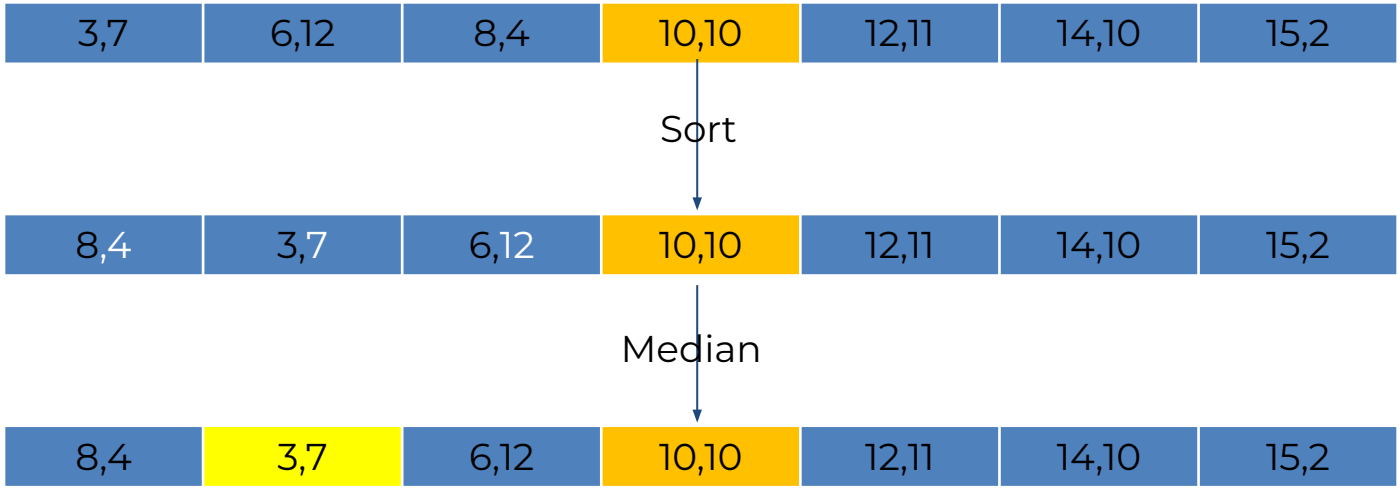


3,7	6,12	8,4	10,10	12,11	14,10	15,2

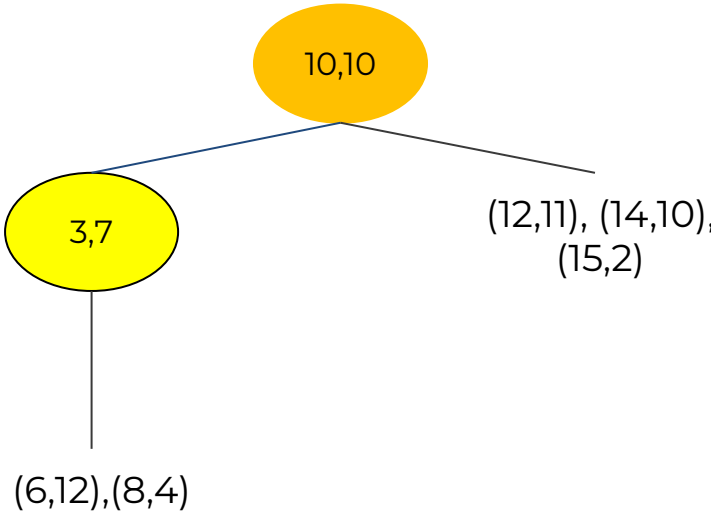


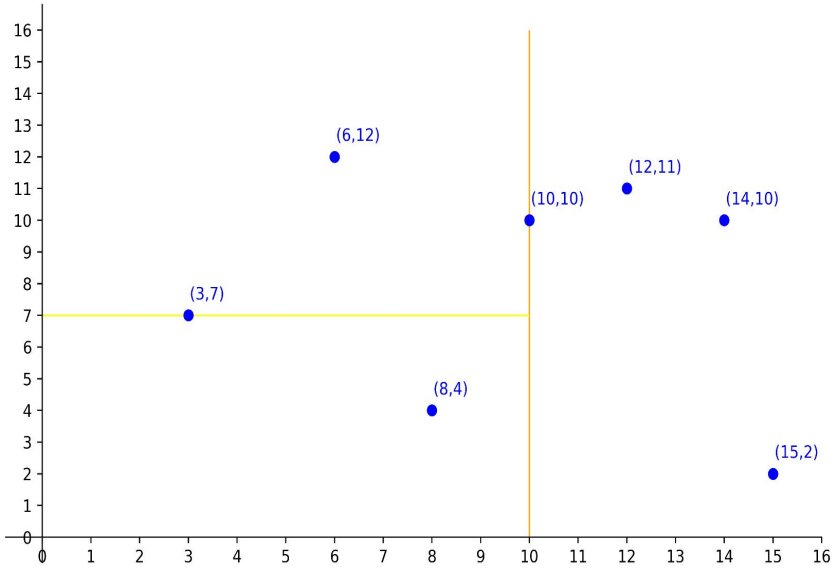


On the left subtree: Sort the data points using *y* coordinate value; find median node and mark the node



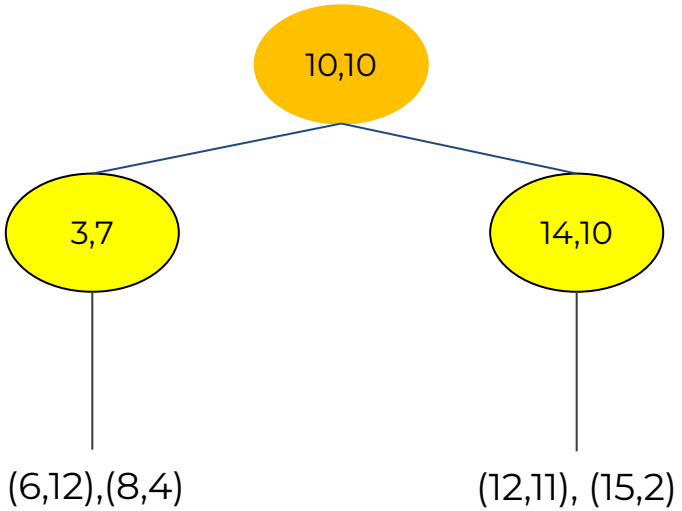
8,4	3,7	6,12	10,10	12,11	14,10	15,2

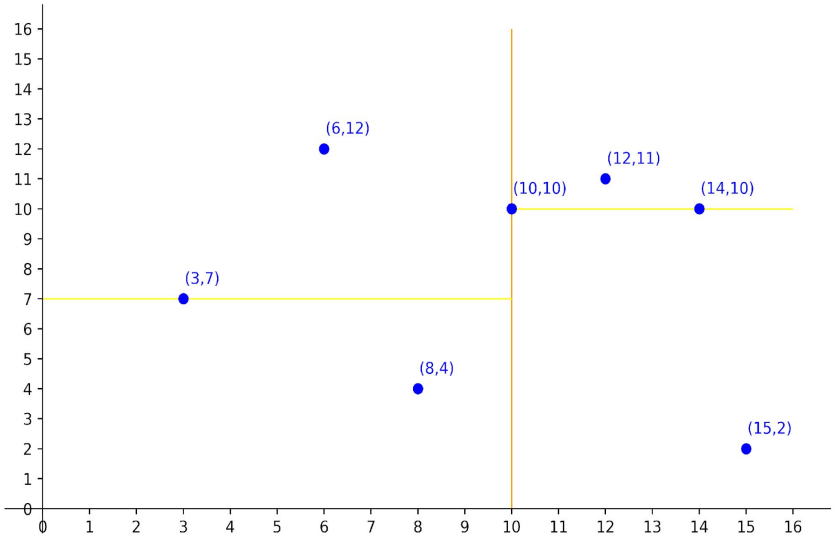




On the right subtree: Sort the data points using *y* coordinate value; find median node and mark the node

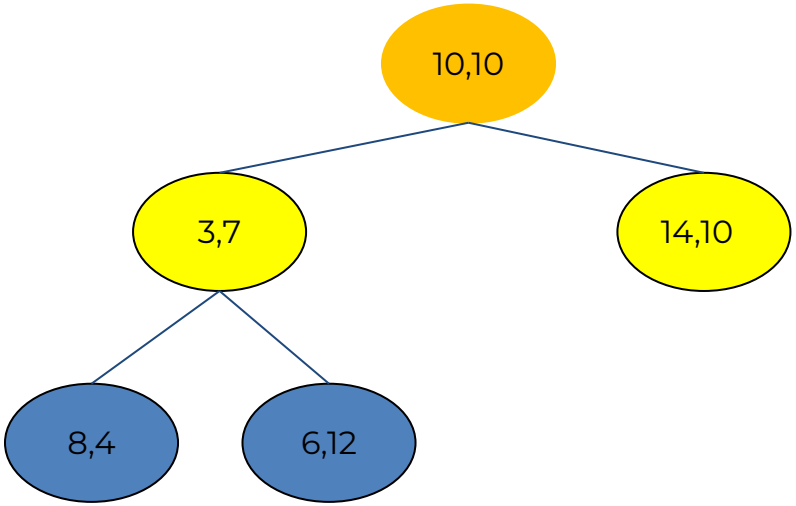
8,4	3,7	6,12	10,10	15,2	14,10	12,11

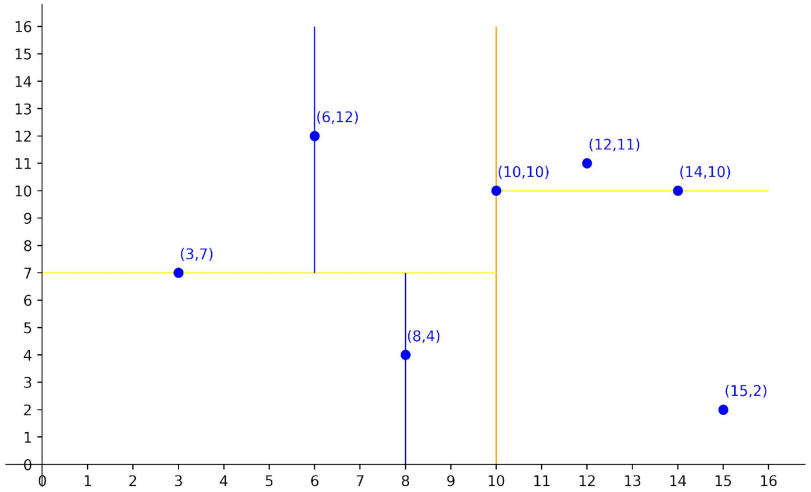




Note: We can stop the algorithm here for this problem; however, the depth of the tree is the programmer's decision

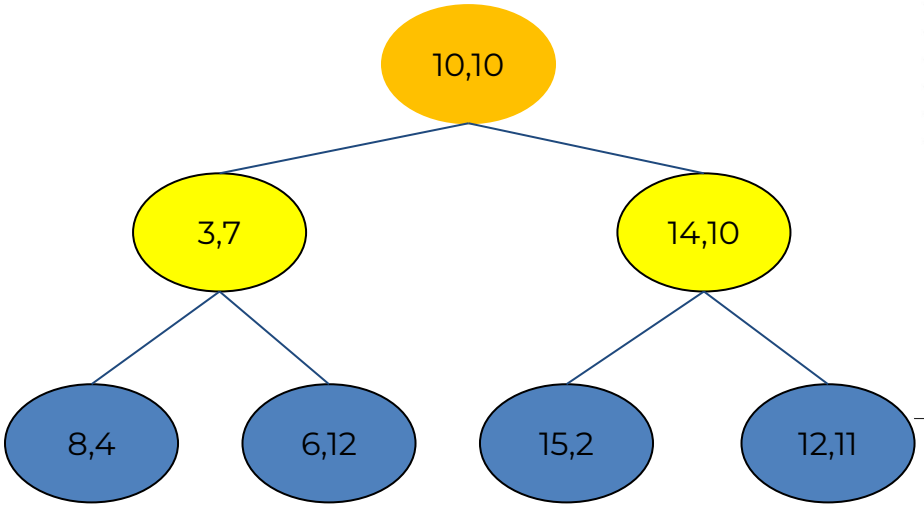
Repeat the process until desired depth (d=2)

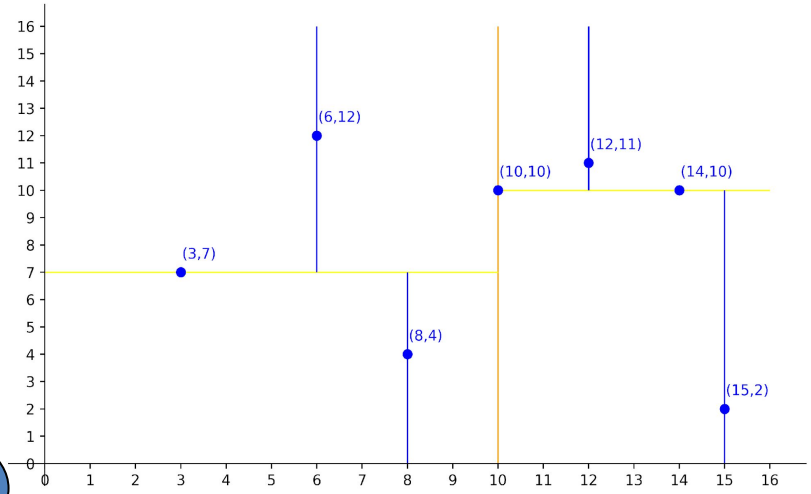




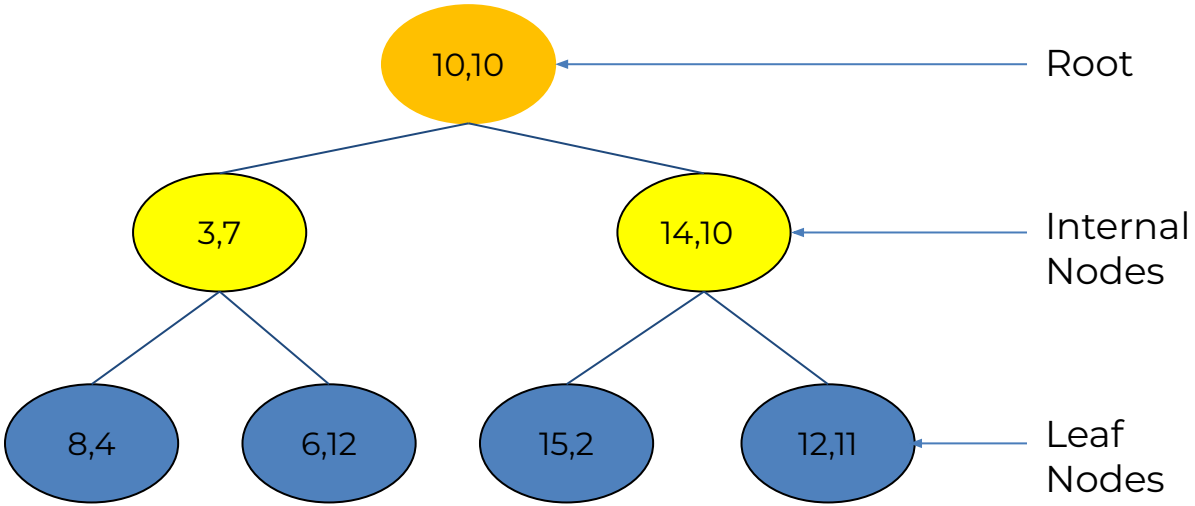
Note: If we have three-dimensional data, the next step would be to split the z-axis and find the median point as a subtree root.

Repeat the process until desired depth (d=2)



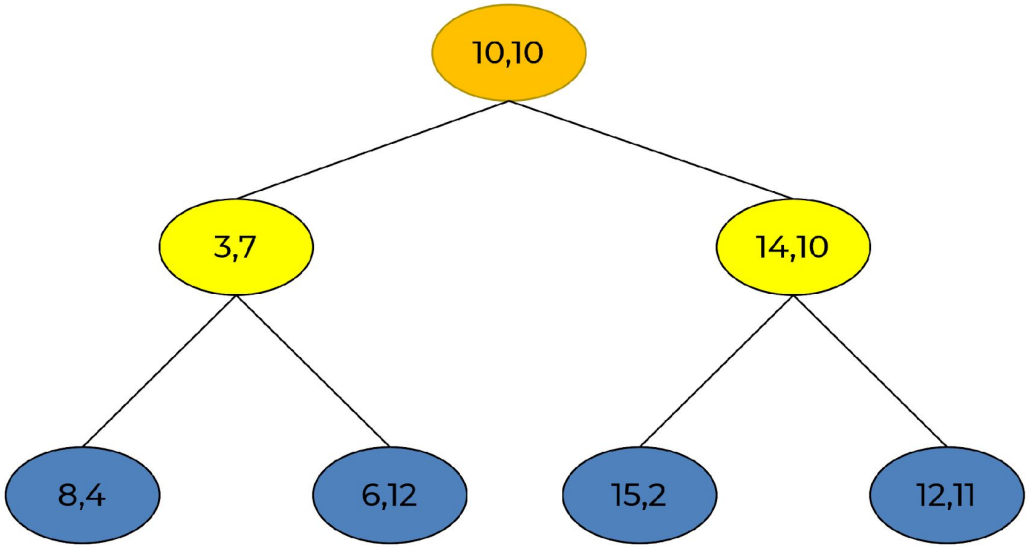


KD Trees: Data structure store set of points on k-dimensional space



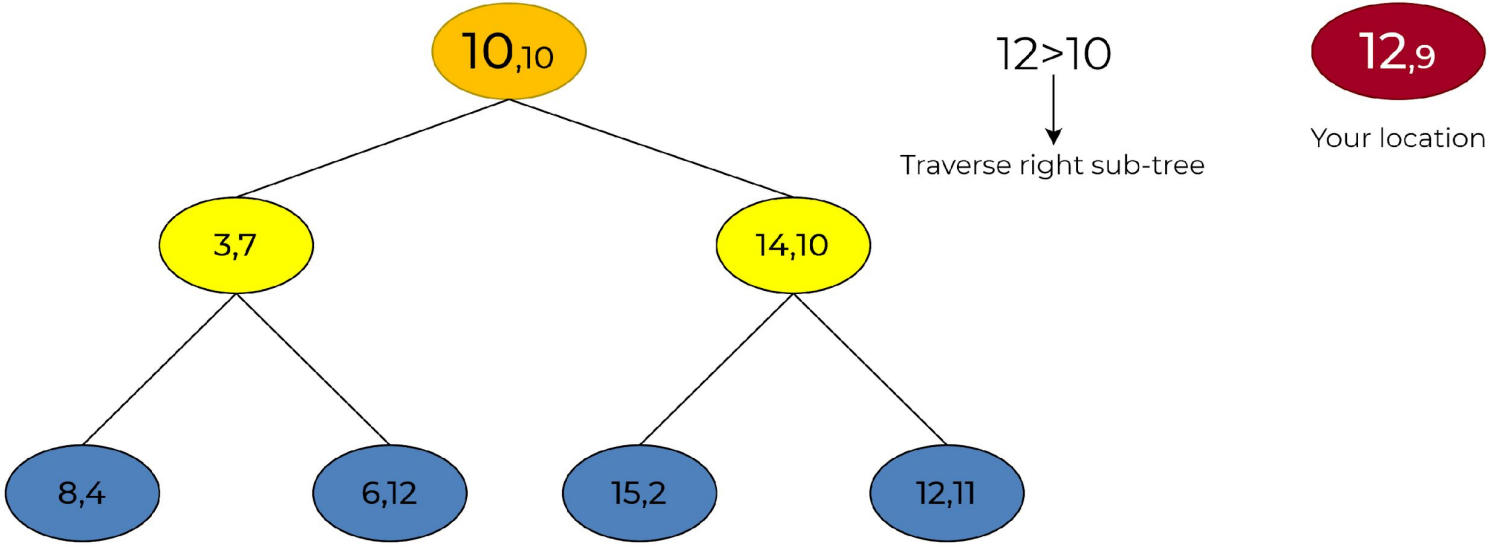
KD Tree: Nearest Neighbor search on the post office problem

Our location is key or the query, and we will try to find the nearest post office using the given KD tree

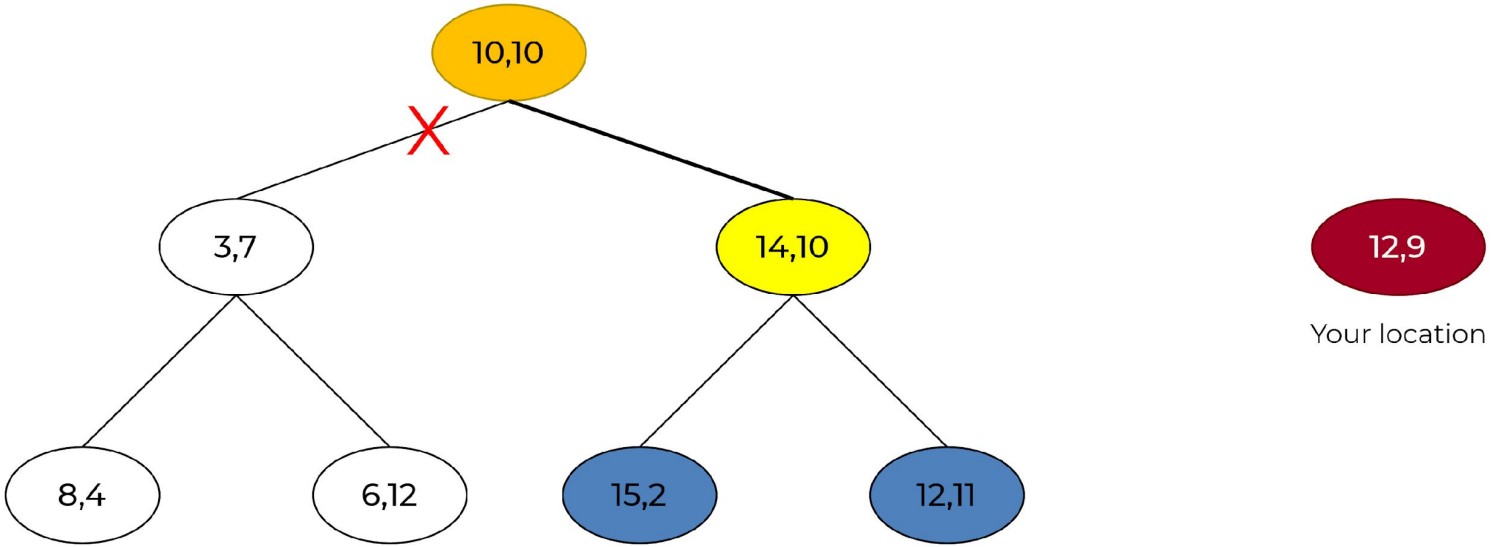


Your location

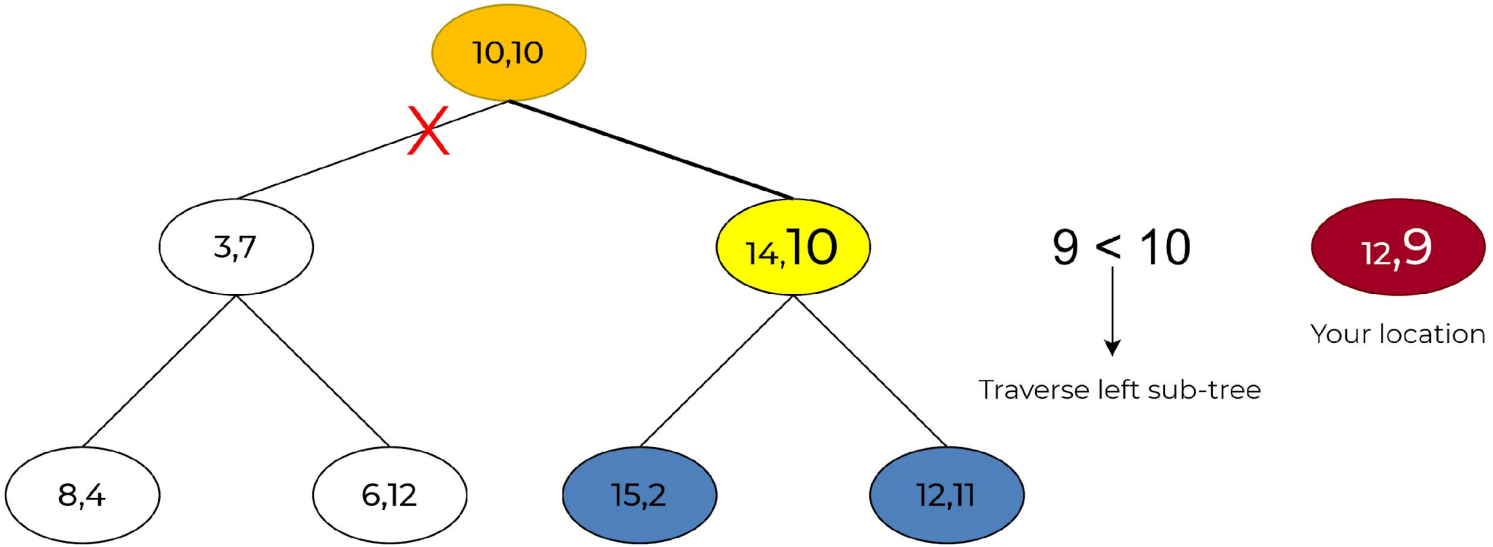
Compare the x coordinate value of the root node with the x coordinate value of the query



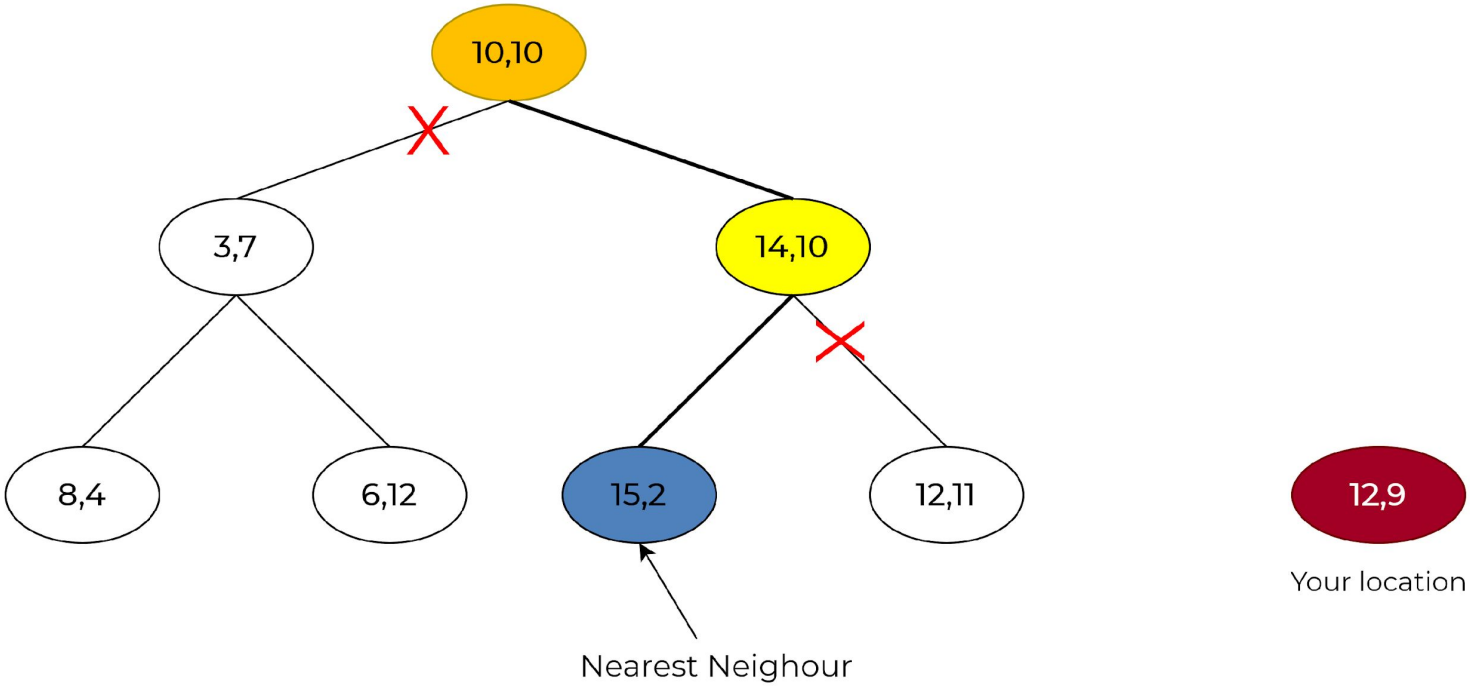
Traverse to the right subtree



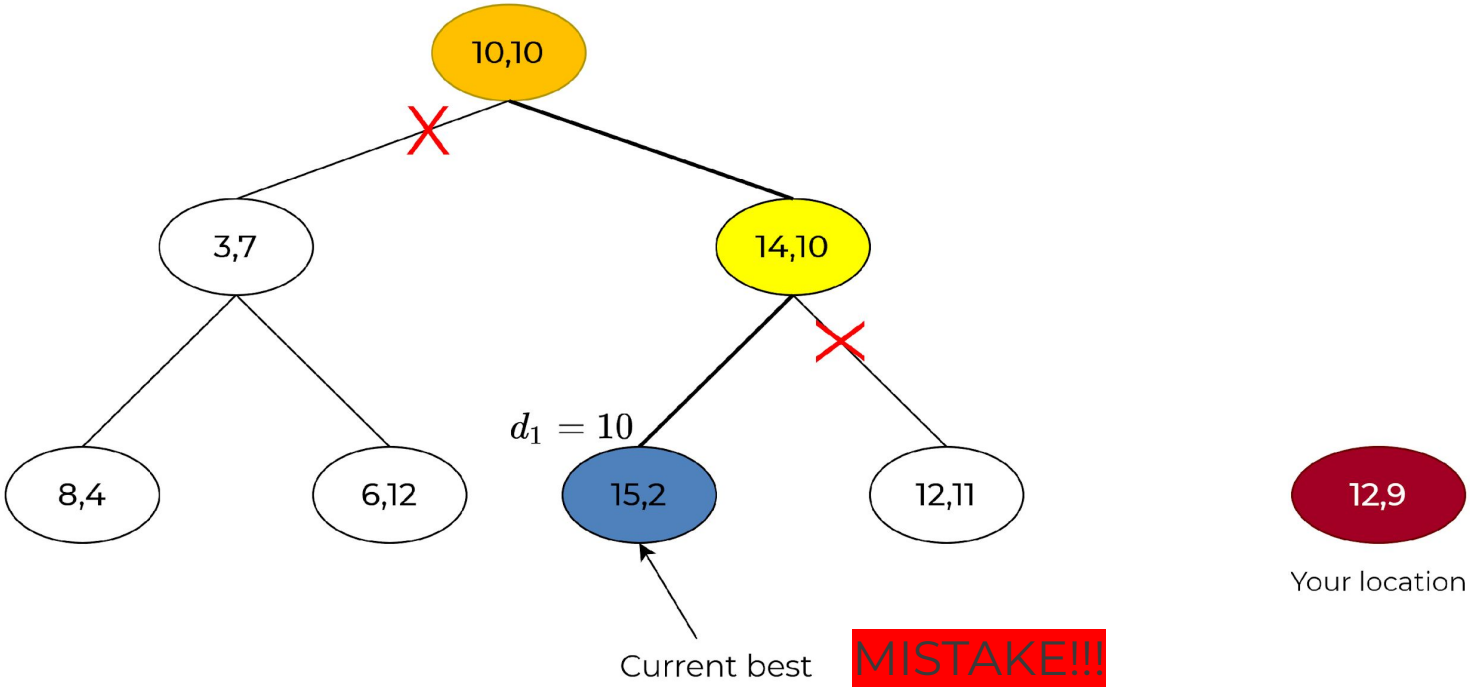
Compare the y coordinate value of the right subtree node with the y coordinate value of the query



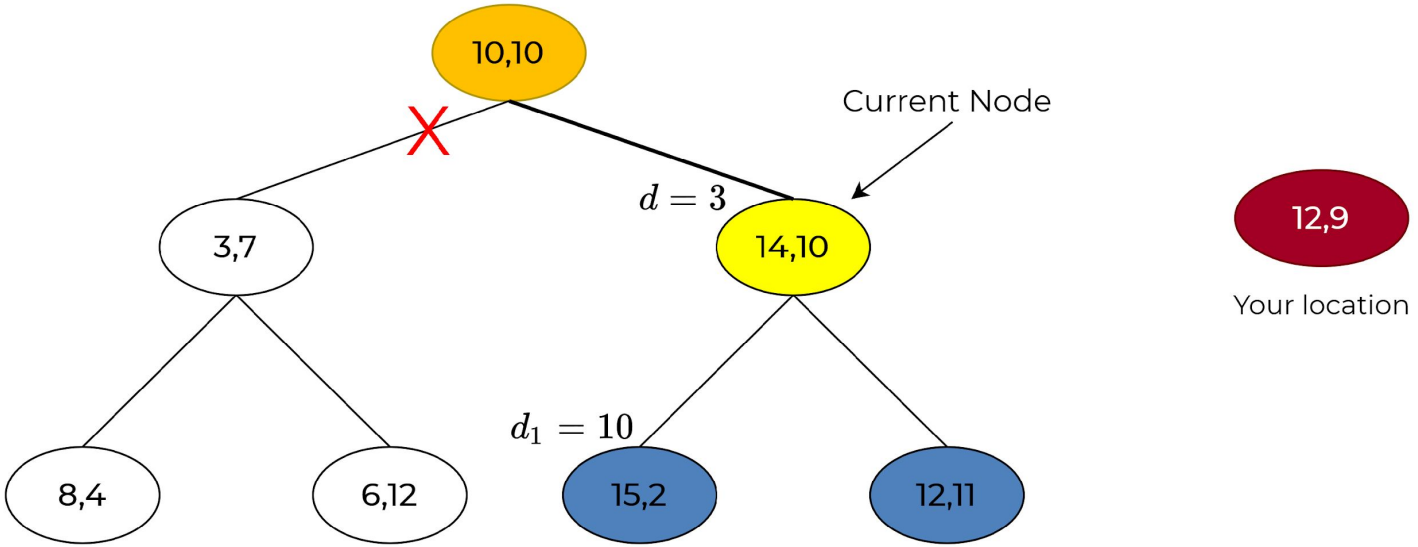
Traverse to the left subtree (leaf node)



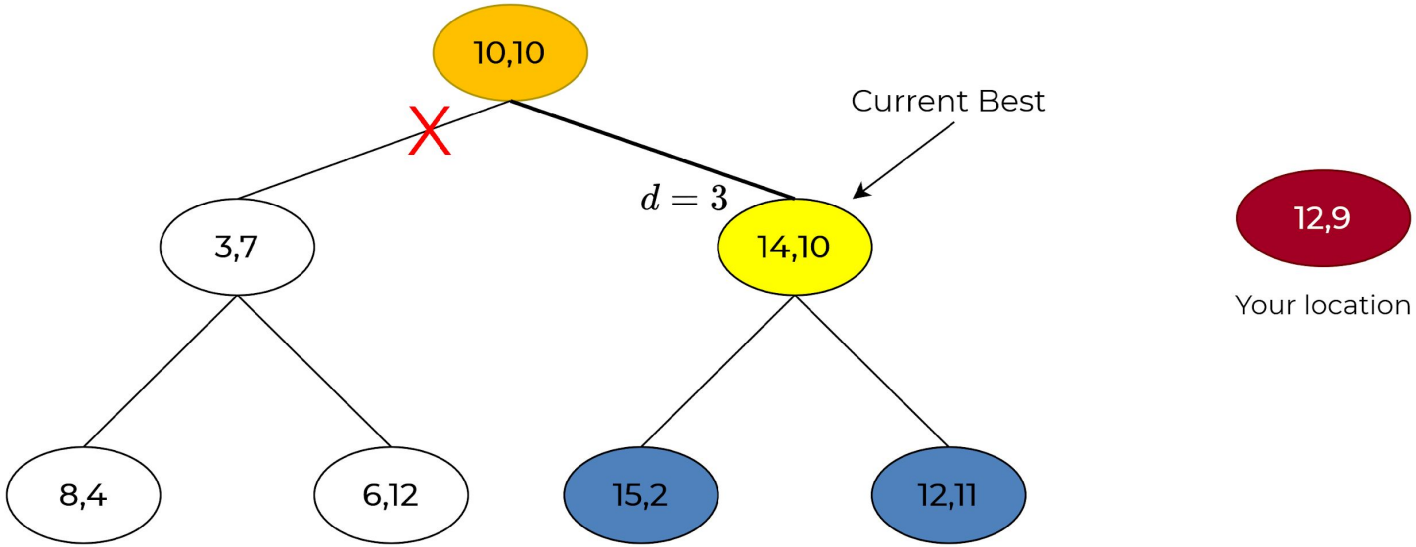
Select leaf node as current best and find the distance between the query and current best



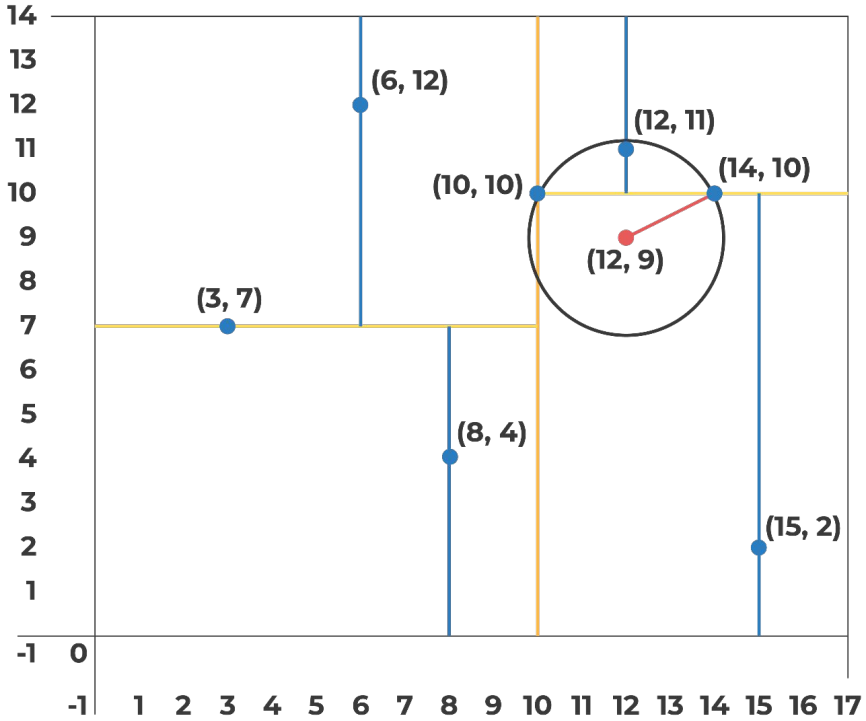
Unwind recursion: The pointer goes to the parent of the current best node; compute the distance



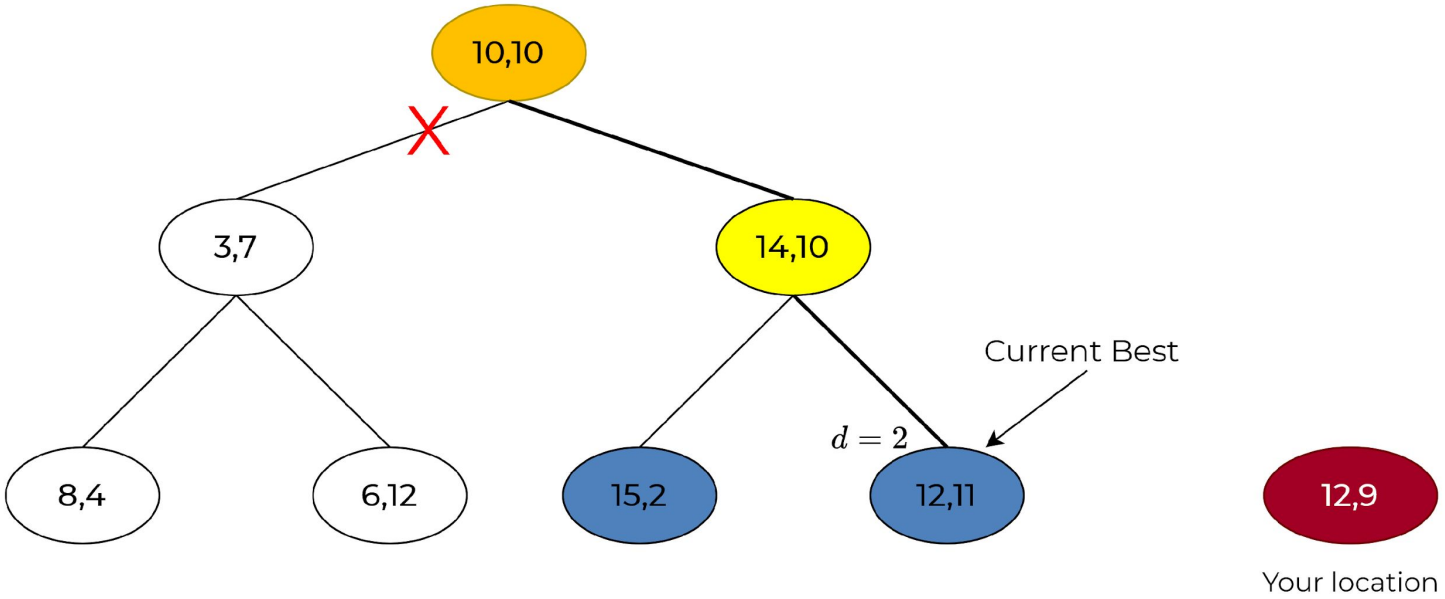
$d < d_1$ change the current node to current best node



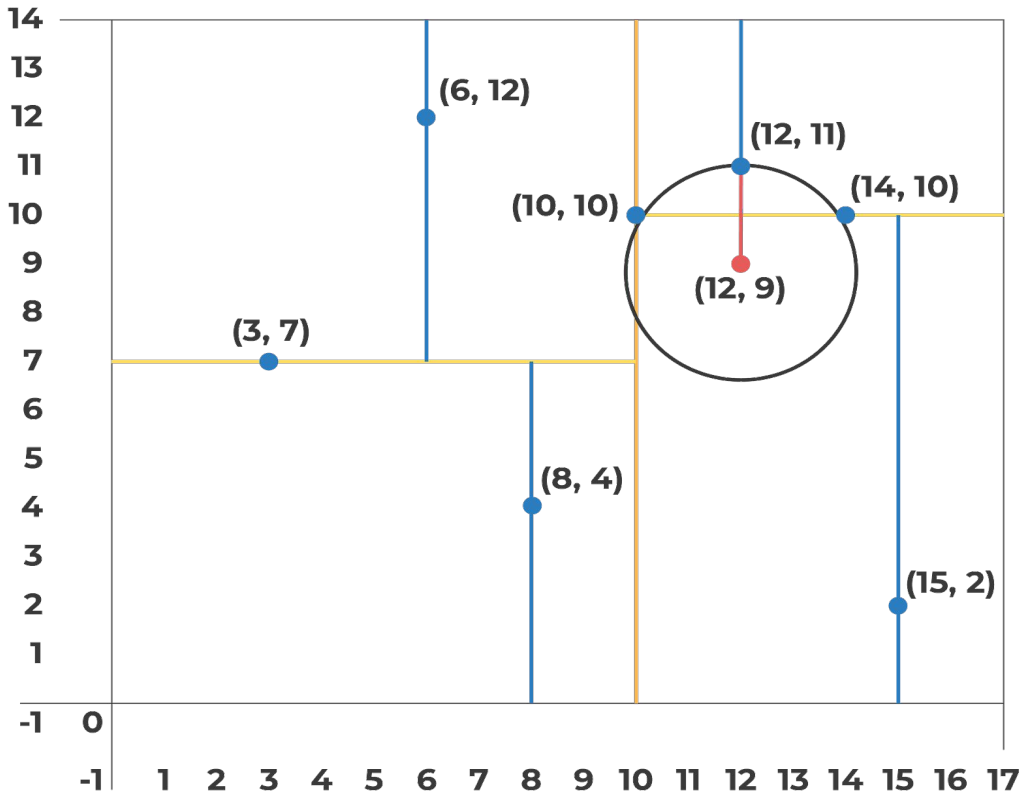
Make a query point (12,9) center of the Circle and radius of a circle = current best distance

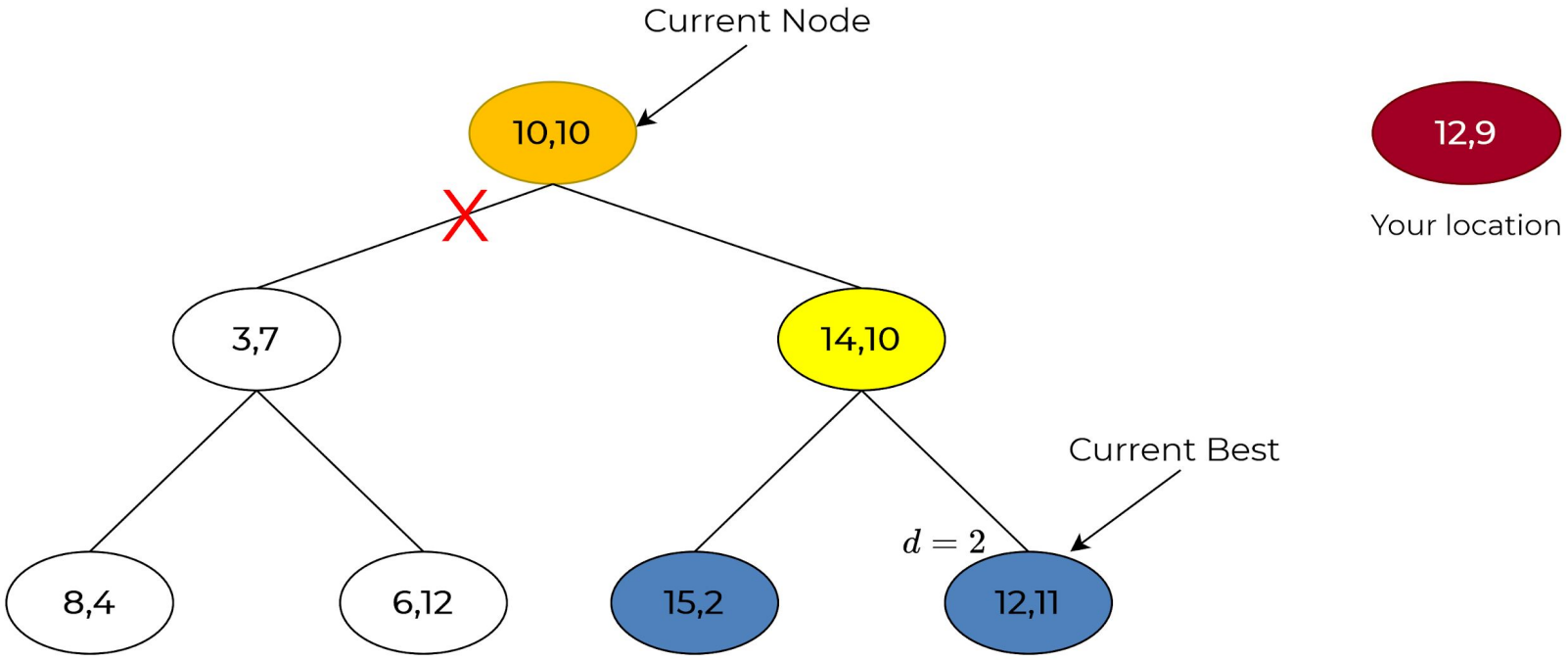


The Pointer goes to the leaf node; Calculate the distance between query and leaf node



Update the circle drawing a radius from (12,9) to (12,11)




Current Node
10,10
12,9
Your location
3,7
14,10
Current Best
$d = 2$
8,4
6,12
15,2
12,11

Fusemachines
AI School

K-NN for Classification & Regression

Model-based learning

Dataset

$$D = \{\{\mathbf{x}_i \in \mathbb{R}^m, y_i \in \{0, 1\}\}, i = 1, 2, \cdots, n\}$$

↓

Fit

$$h : \mathbf{x} \rightarrow y \approx f : \mathbf{x} \rightarrow y$$

↓

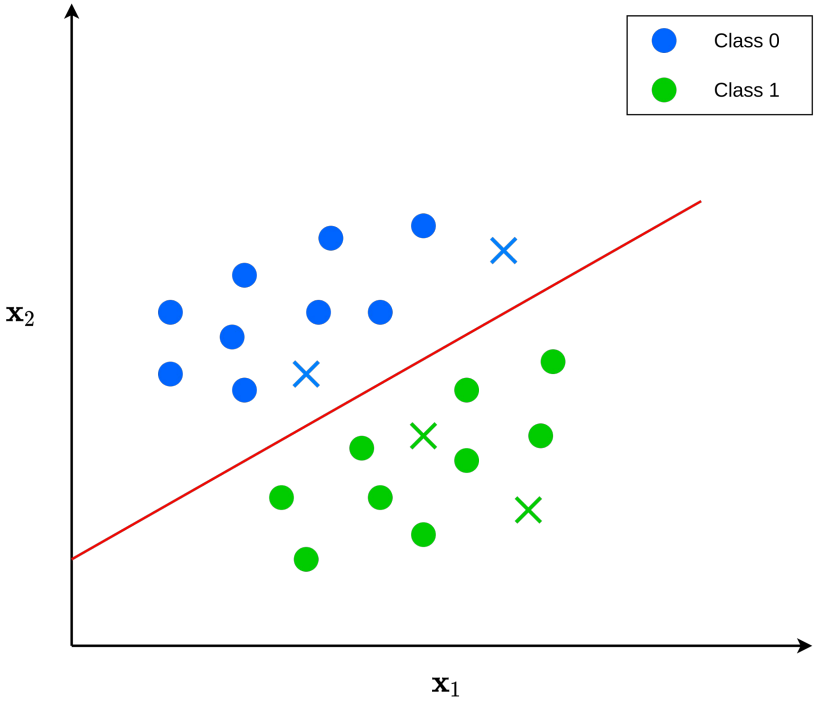
Unseen samples

$$D_t = \{\mathbf{x}'_i, i = 1, \cdots, z\}$$

↓

Predict

$$h(\mathbf{x}'_i) = \hat{y}_i$$



Working of Instance-based learning

Dataset

$$D = \{\{\mathbf{x}_i \in \mathbb{R}^m, y_i \in \{0, 1\}\}, i = 1, 2, \cdots, n\}$$

↓

Fit

Store D (either directly or using data structure)

↓

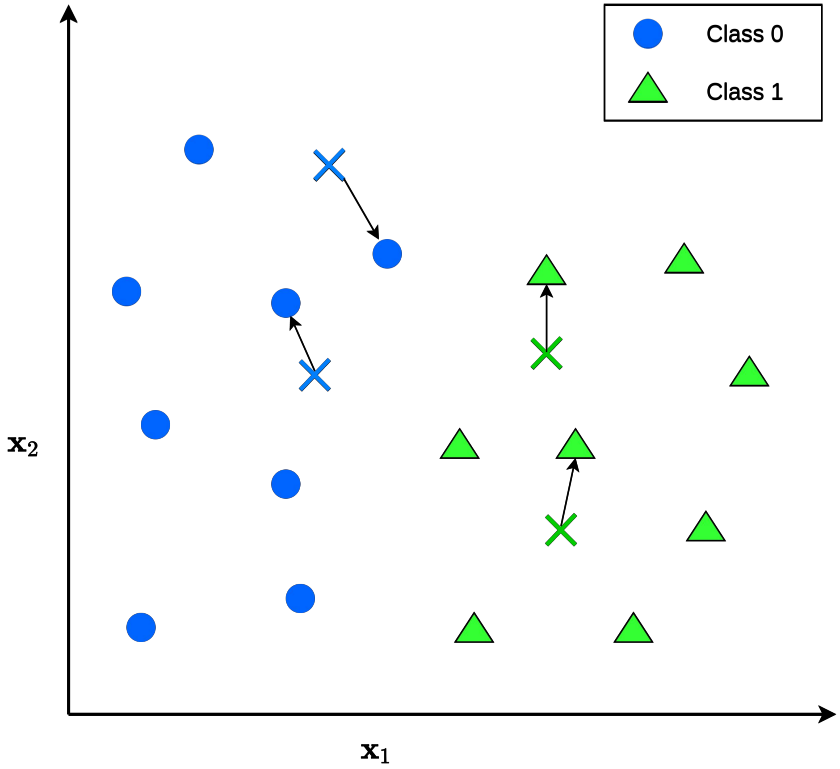
Unseen samples

$$D_t = \{\mathbf{x}_i, i = 1, \cdots, z\}$$

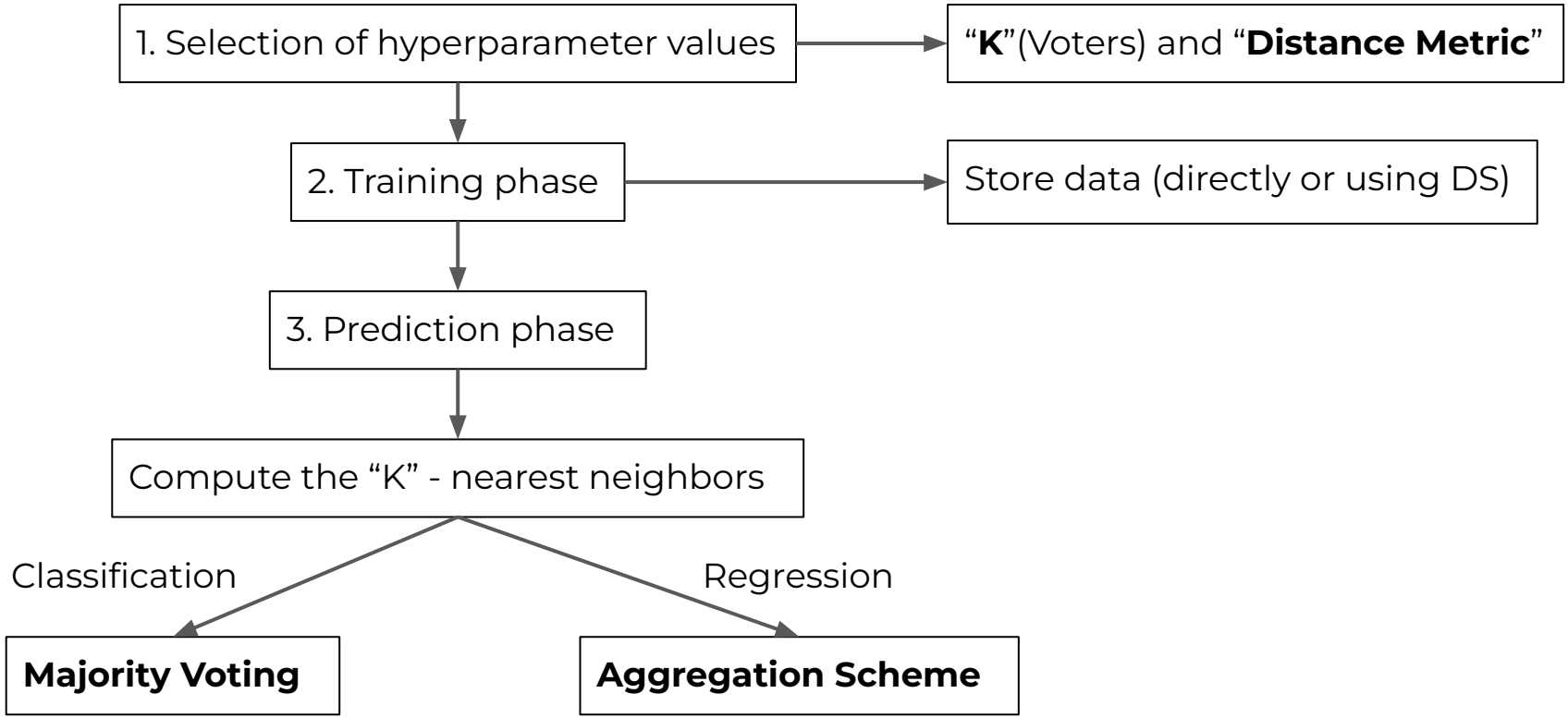
↓

Predict

$$h(\mathbf{x}'_i) = \text{lookup}\,(\mathbf{x}) = \hat{y}_i$$



Working of K-NN



Working of K-NN classifier

Given, Training Dataset

$$D = \{\{\mathbf{x}_i \in \mathbb{R}^m, y_i \in \{0, 1, 2\}\}, i = 1, 2, \cdots, n\}$$

↓

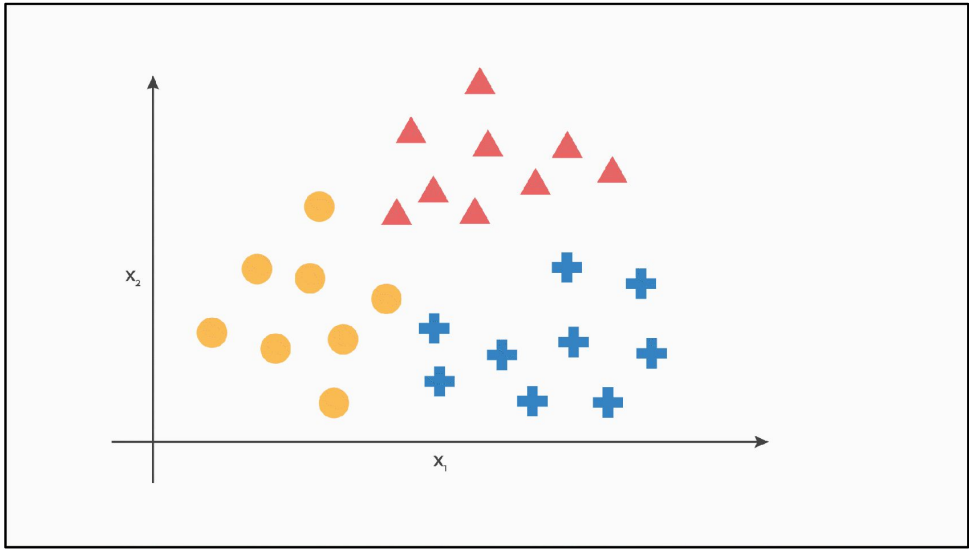
Set hyperparameters

K = 5 and **Euclidean** distance metric

↓

Training Phase

Store D (either directly or using data structure)



Working of K-NN classifier

Prediction Phase

For single unseen test sample, $\mathbf{x}_t$

↓

Compute the set of "**K**"(5)-**nearest neighbors**, $S_{\mathbf{x}_t}$

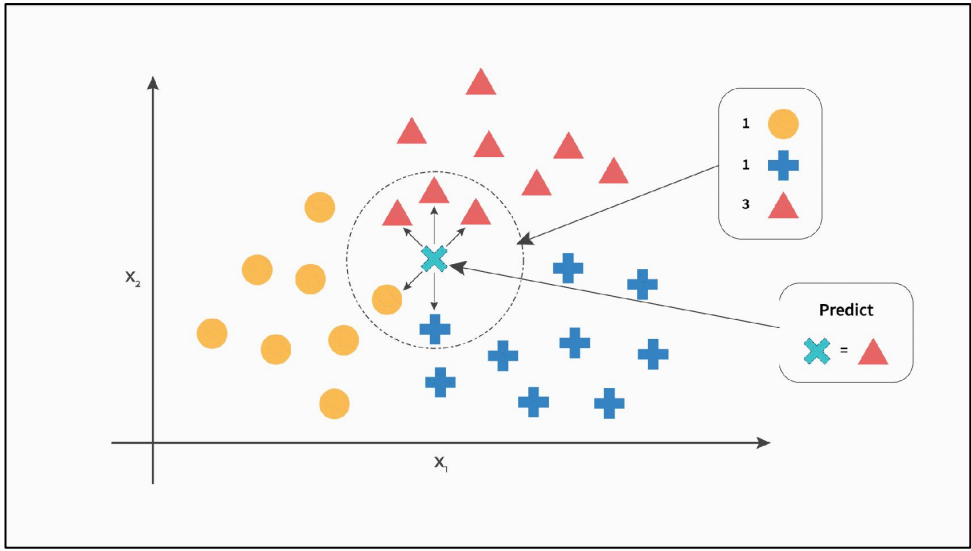
↓

Predict the target value of unseen sample, $h(\mathbf{x}_t)$

↓

Majority voting

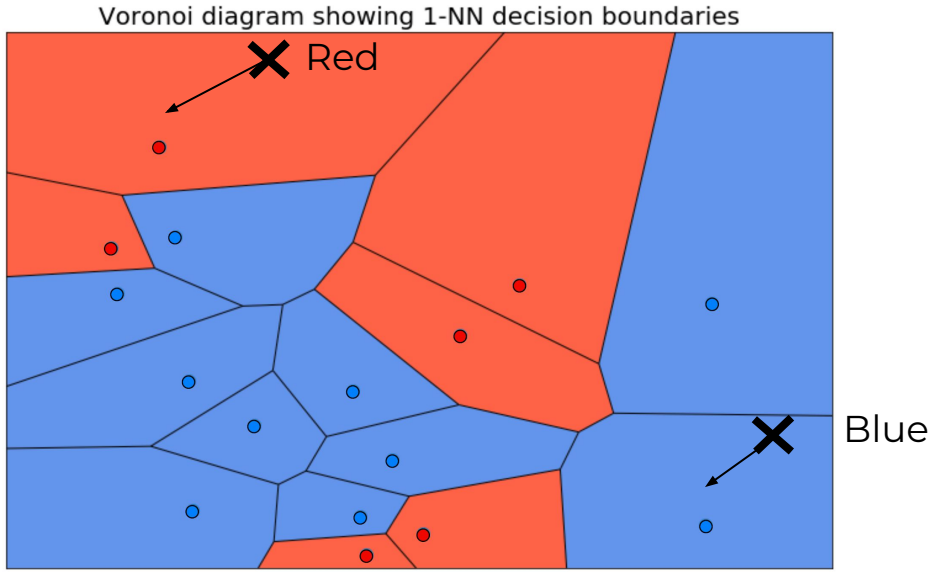
$$h(\mathbf{x}_t) = \text{mode}\left(\{y'' : (\mathbf{x}'', y'') \in S_{\mathbf{x}_t}\}\right)$$



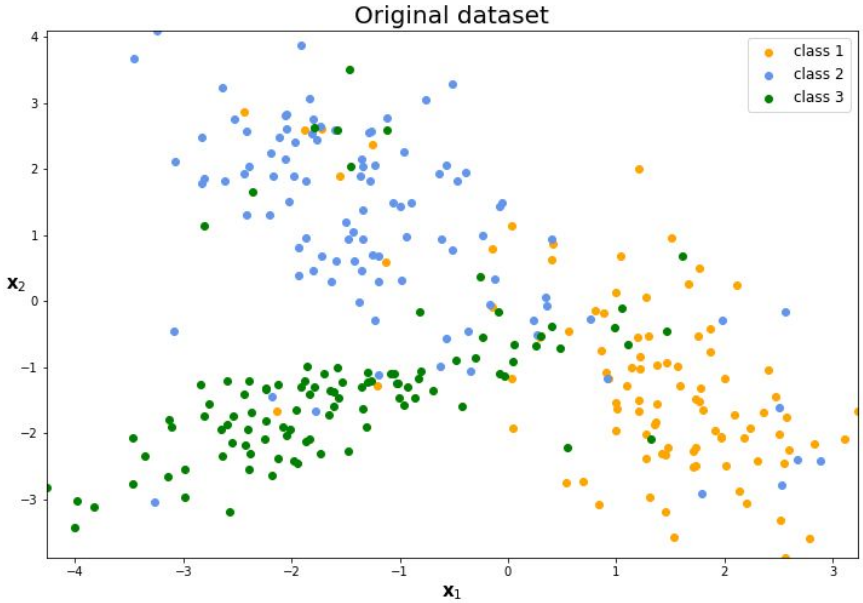
Computation complexity: **O(knm)**

Decision boundary of 1-NN classifier

Voronoi diagram - a partition of a plane into regions close to each of a points of dataset.

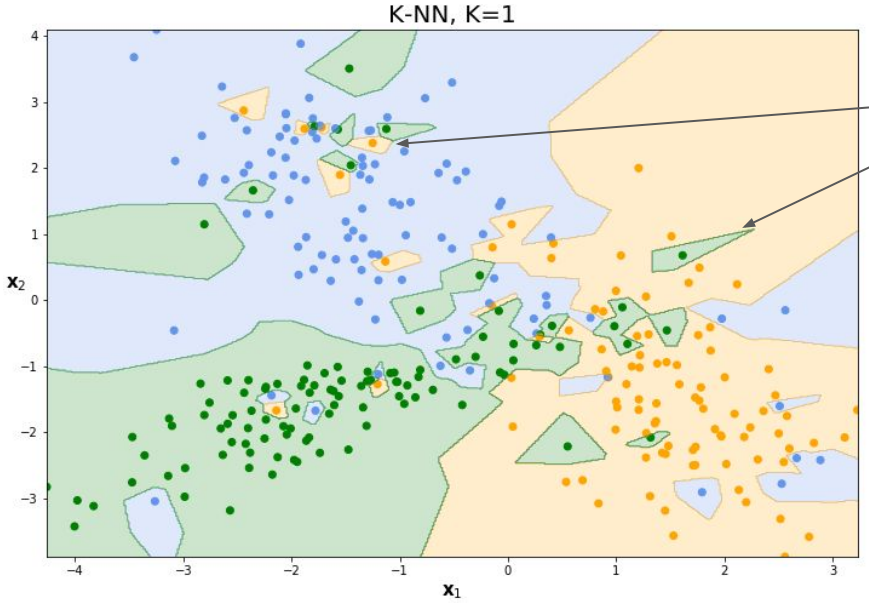


Effect of *K* in decision boundary



Noisy multiclass classification dataset

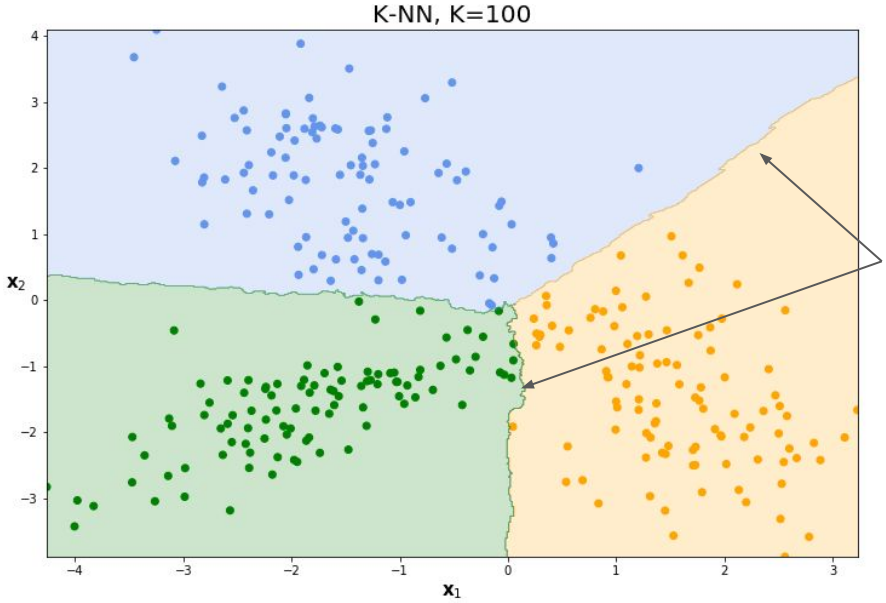
Effect of K in decision boundary



Jagged(Overfit)

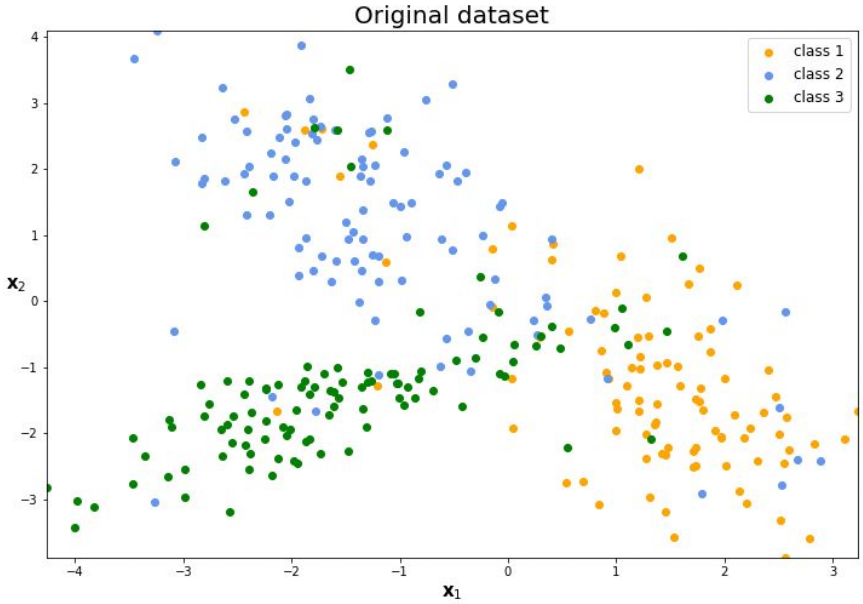
Small value of K - **High variance, low bias classifier**(flexible/jagged decision boundary)

Effect of K in decision boundary



Large value of K - **Low variance, High bias classifier**(less flexible boundary)

Effect of K in decision boundary



Effect of K on K-NN classifier decision boundary.

Working of K-NN Regression

Given, Training Dataset

$$D = \{\{\mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}^1, i = 1, 2, \cdots, n\}\}$$

↓

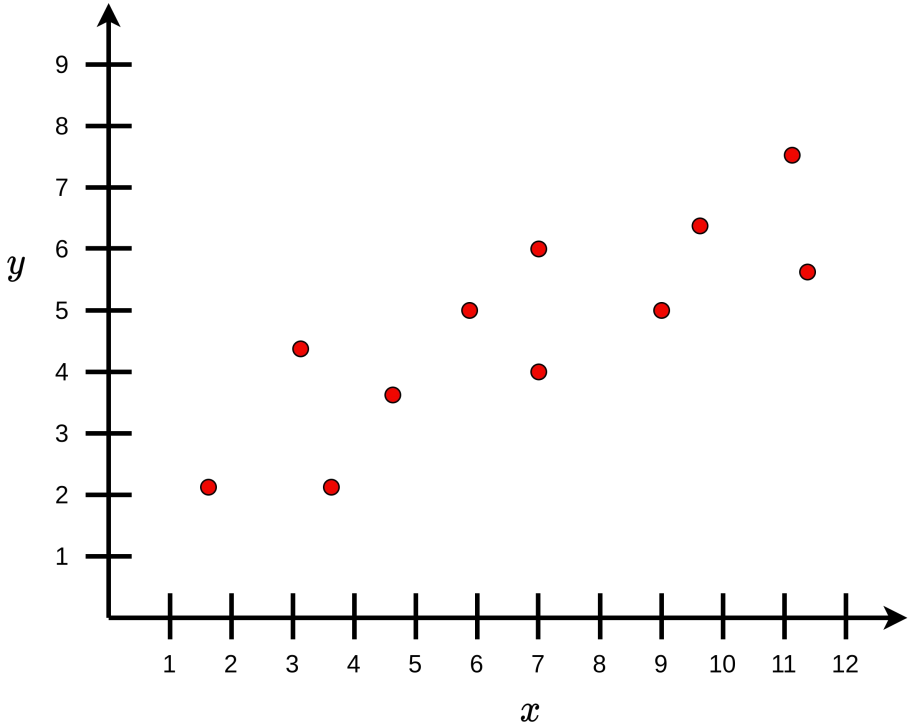
Set hyperparameters

K = 4 and **Euclidean** distance metric

↓

Training Phase

Store D (either directly or using data structure)



Working of K-NN Regression

Prediction Phase

For single unseen test sample, $\mathbf{x}_t$

↓

Compute the set of "**K**"(4)-**nearest neighbors**, $S_{\mathbf{x}_t}$

↓

Predict the target value of unseen sample, $h(\mathbf{x}_t)$

↓

Aggregation Scheme

$$h(\mathbf{x}_t) = \frac{\sum_{y'' \in S_{\mathbf{x}_t}} y''}{K}$$