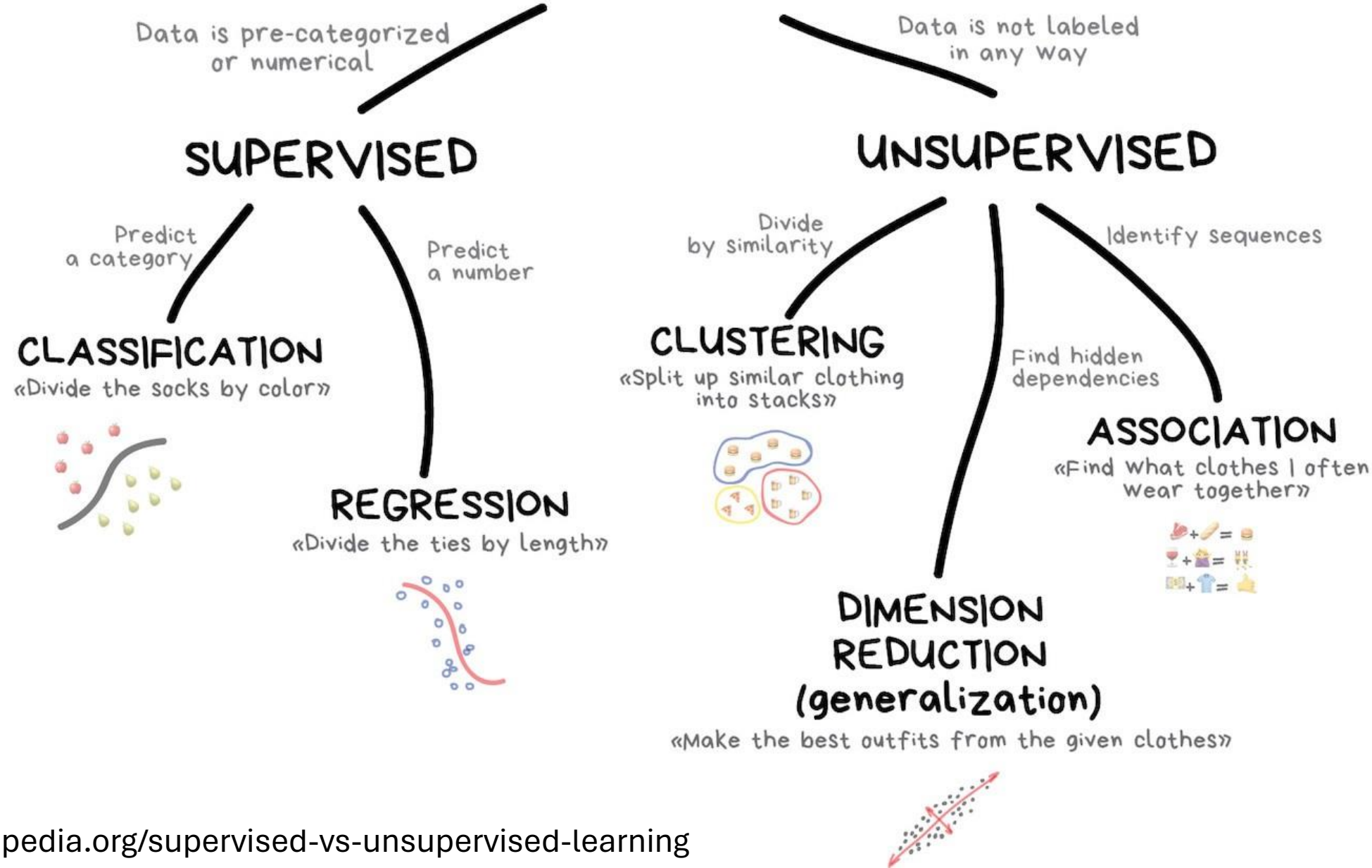
ML with Python

Rina BUOY

AMERICAN UNIVERSITY
OF PHNOM PENH

STUDY LOCALLY. LIVE GLOBALLY.

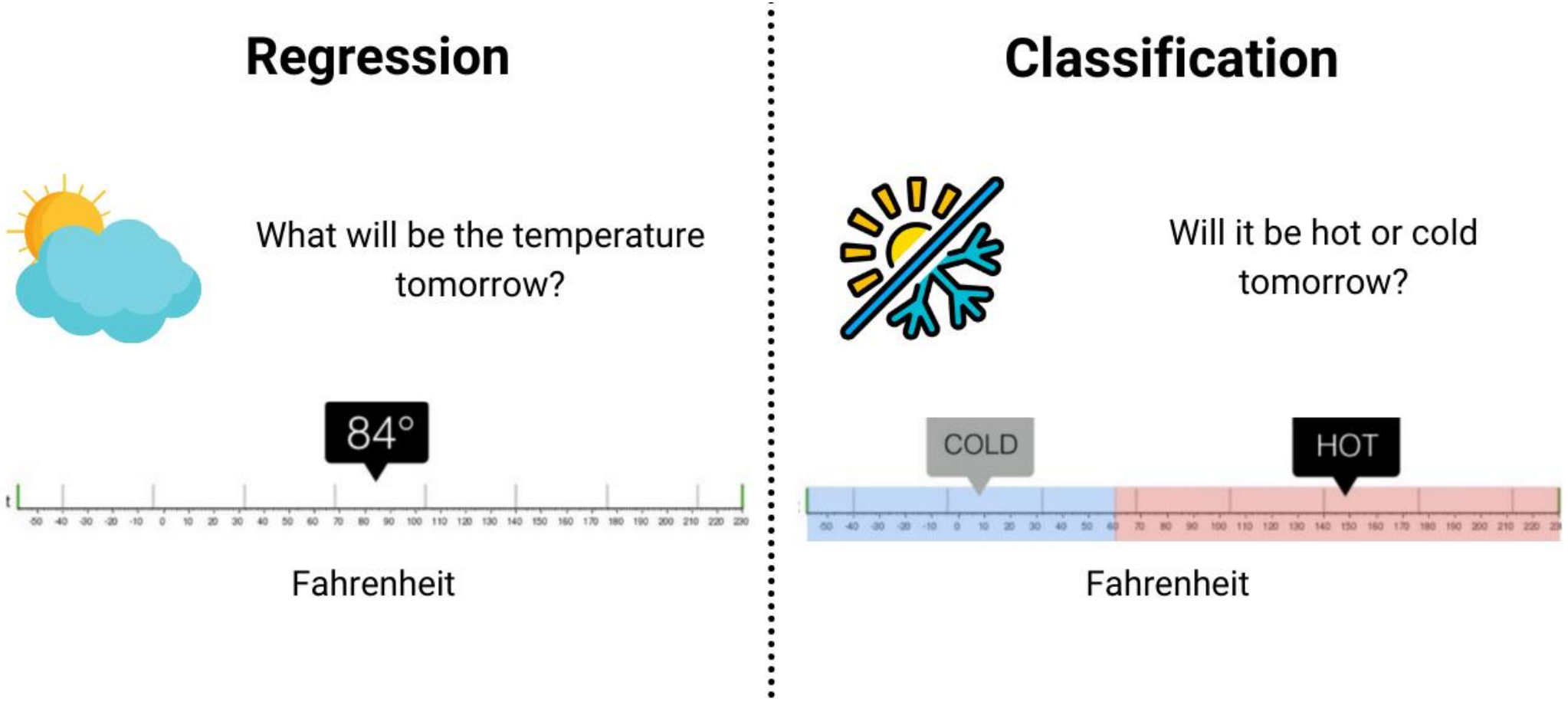
Overview of ML with Python



https://devopedia.org/supervised-vs-unsupervised-learning

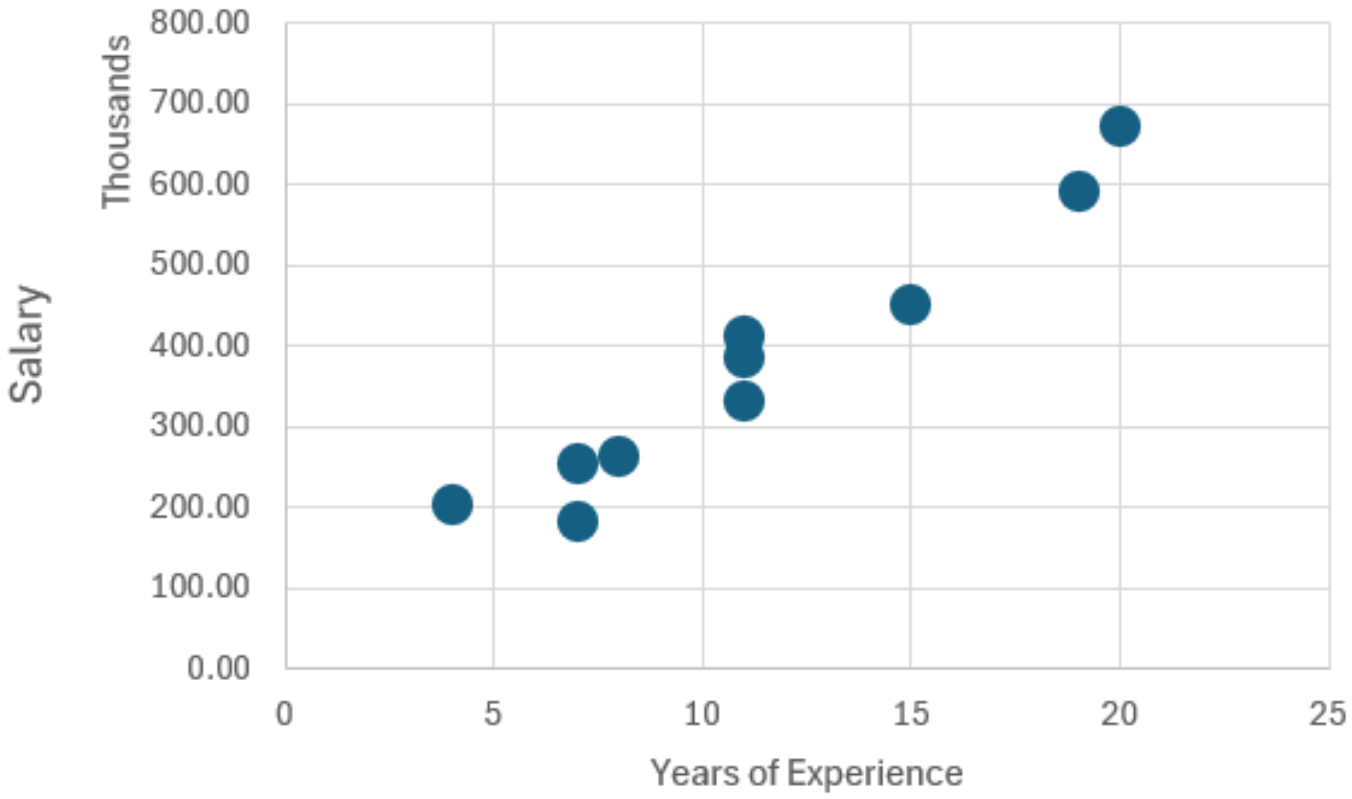
Regression vs. Classification

Regression vs. Classification



https://www.linkedin.com/pulse/aipart3regression-vs-classification-models-arnab-mukherjee/

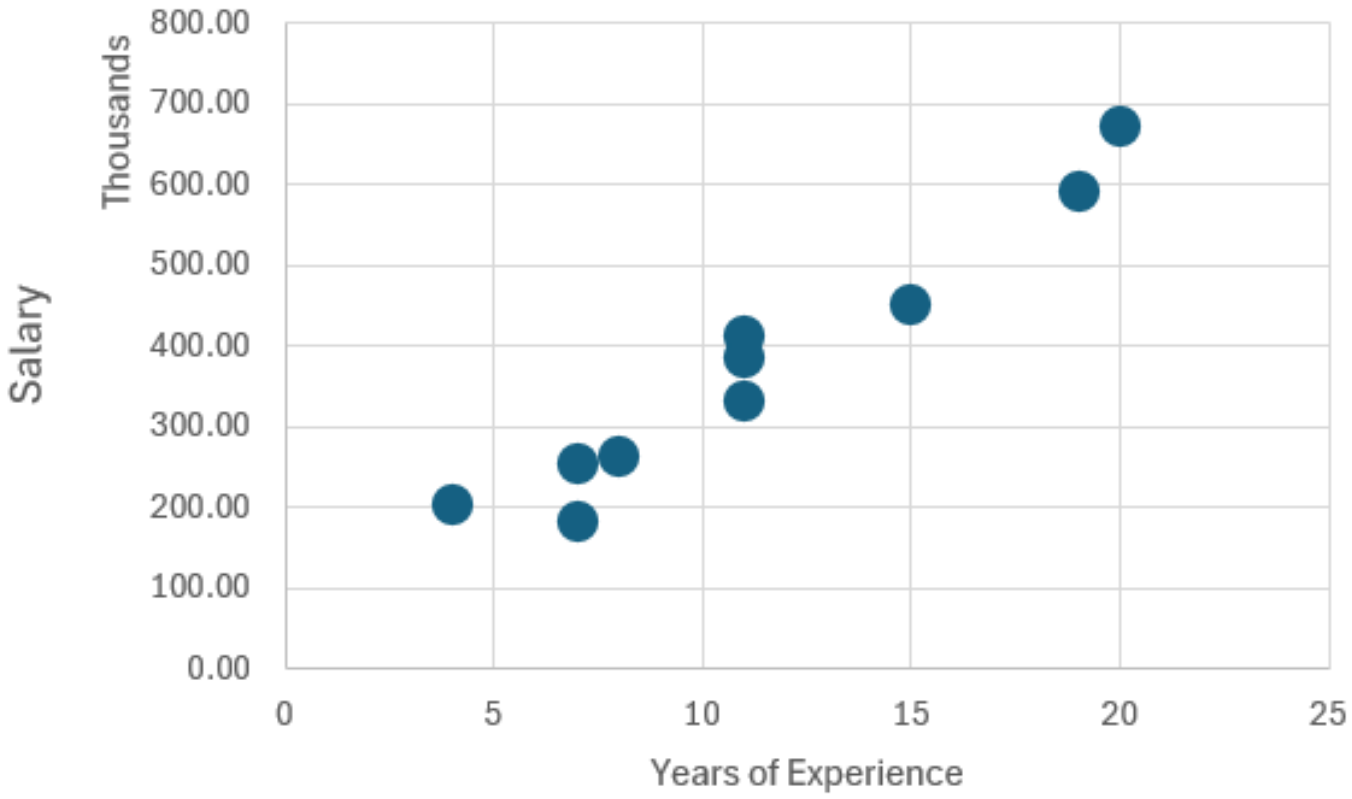
Correlation vs. Regression



- Correlation analysis confirms **the relationship** and **connection** between two variables.

- Linear regression analysis shows **how much one variable affects another** and how to **predict, estimate, or explain its behavior**.

Prediction vs. Inference



1. **Predictions :** predicting salary at any yrs of exp.
2. **Inference :** explaining whether a predictor (yrs of exp.) has an influence on the response variable (salary).

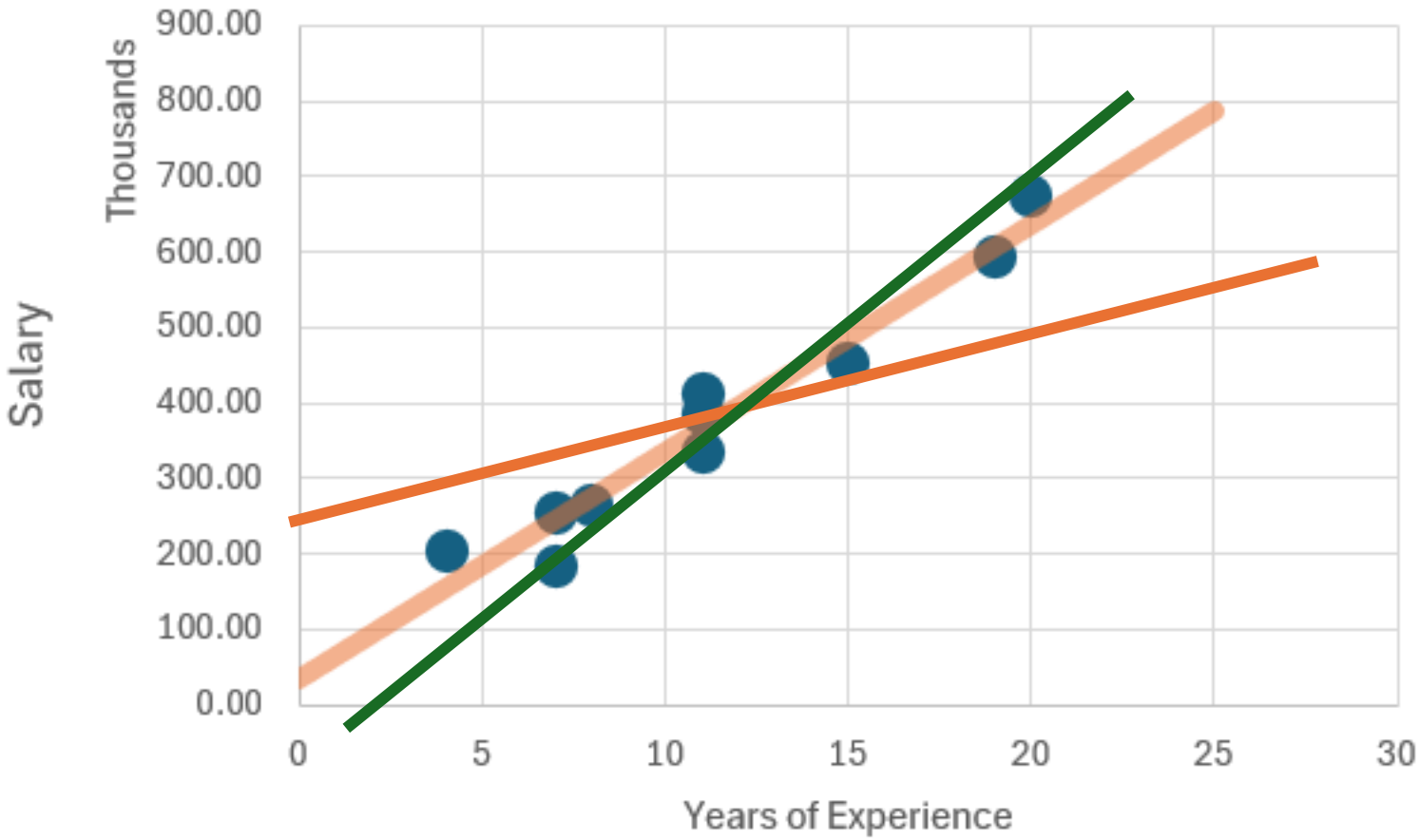
Simple Linear Regression

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Where:

- Y is the dependent variable (response).
- X is the independent variable (predictor).
- β_0 is the intercept (the value of Y when $X = 0$).
- β_1 is the slope (the change in Y for a unit change in X).
- ε represents the error term, which captures the variability in Y that is not explained by the linear relationship with X.

Best Fit-Line (function)

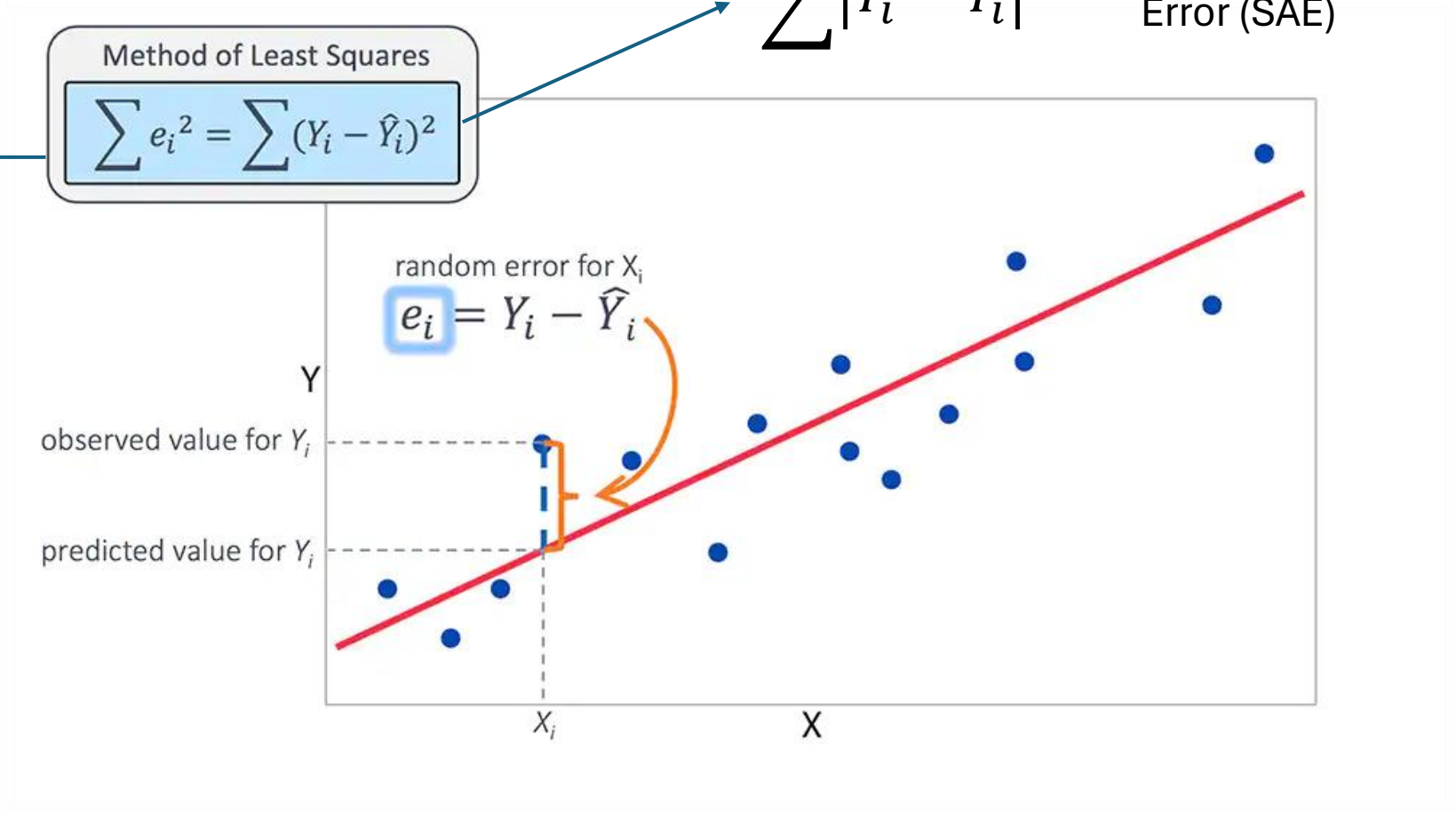


Best Fit-Line (function)

$$\sum |Y_i - \hat{Y}_i|$$

Sum Absolute Error (SAE)

Sum Squared Error (SSE)



Multiple Linear Regression

- **Model Representation**: In multiple linear regression, the relationship between the dependent variable y and the independent variables $x_1, x_2, \ldots, x_n$ is assumed to be linear.
- **Equation**: The equation for multiple linear regression is:

  $$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_n \cdot x_n + \epsilon$$

  where:

  - y is the dependent variable.
  - $x_1, x_2, \ldots, x_n$ are the independent variables.
  - $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (slopes) of the independent variables.
  - ϵ is the error term.

Polynomial Regression

- **Model Representation**: Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an n-degree polynomial.
- **Equation**: The equation for polynomial regression is:

$$y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \ldots + \beta_n \cdot x^n + \epsilon$$

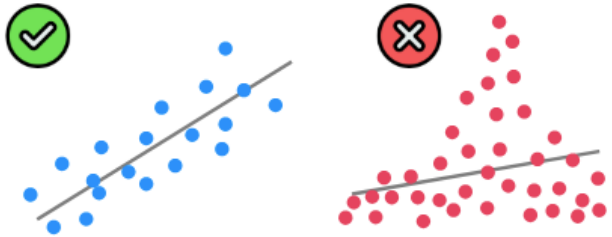
$x^2 \rightarrow x_2$, $x^n \rightarrow x_n$

where:

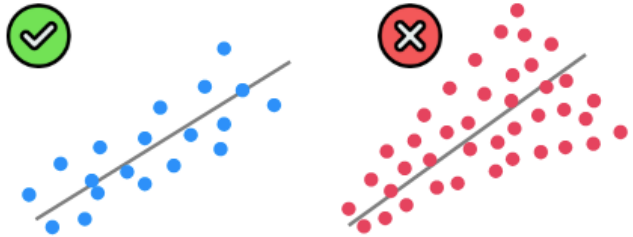
  - y is the dependent variable.
  - x is the independent variable.
  - $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (slopes) of the polynomial terms.
  - ϵ is the error term.

Linear Regression's Assumptions

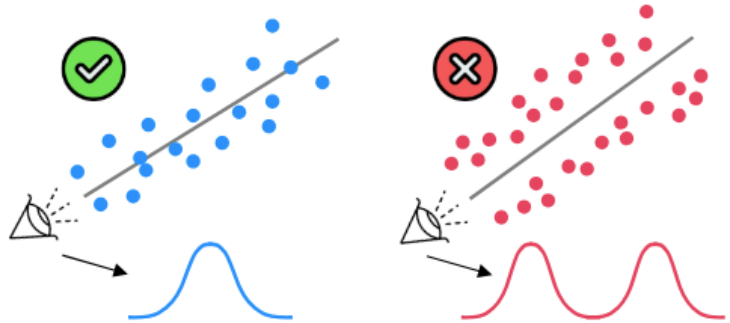
1. Linearity
(Linear relationship between Y and each X)



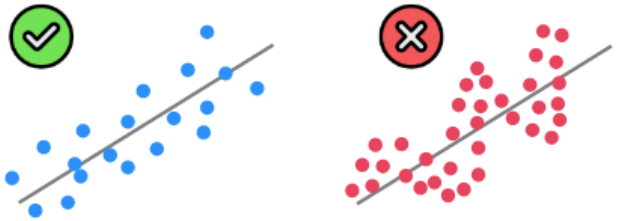
2. Homoscedasticity
(Equal variance)



3. Multivariate Normality
(Normality of error distribution)



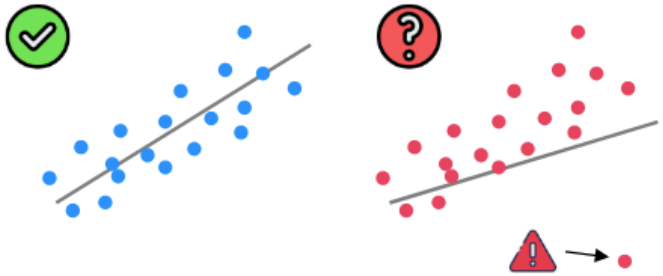
4. Independence
(of observations. Includes "no autocorrelation")



5. Lack of Multicollinearity
(Predictors are not correlated with each other)

$X_1 \not\sim X_2$ $\quad X_1 \sim X_2$

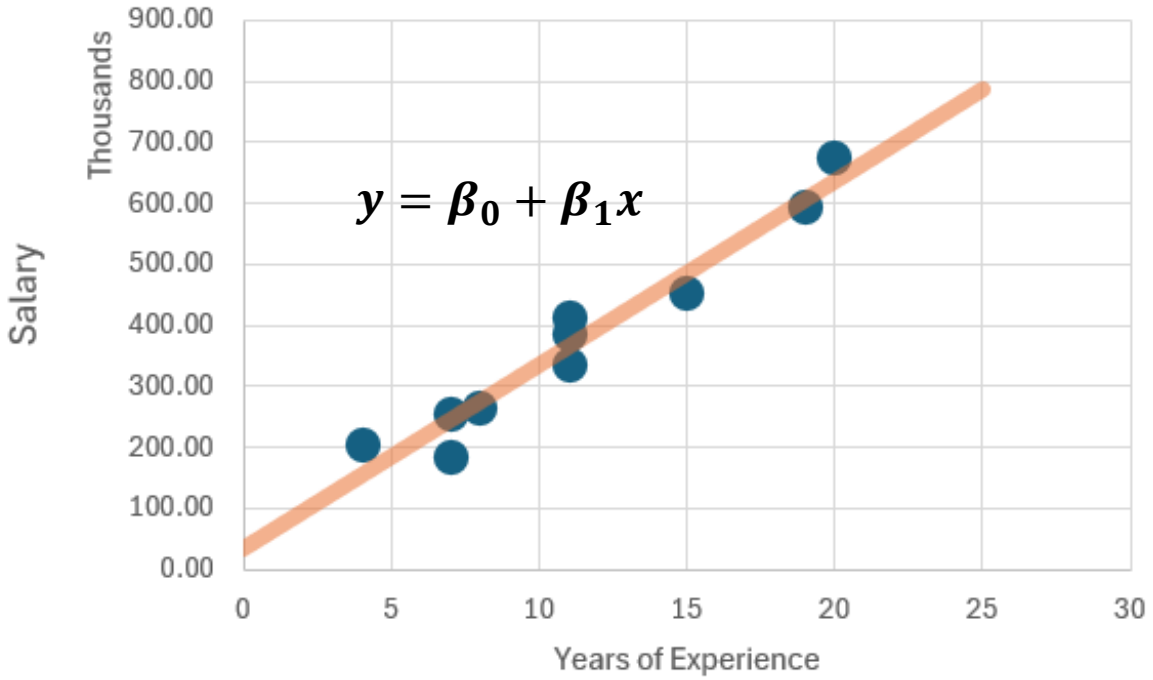
6. The Outlier Check
(This is not an assumption, but an "extra")



Interpretation of Coefficients

- β_1 represents the estimated increase in Y per unit increase in X. Note that the increase may be negative which is reflected when is negative.

- β_o?



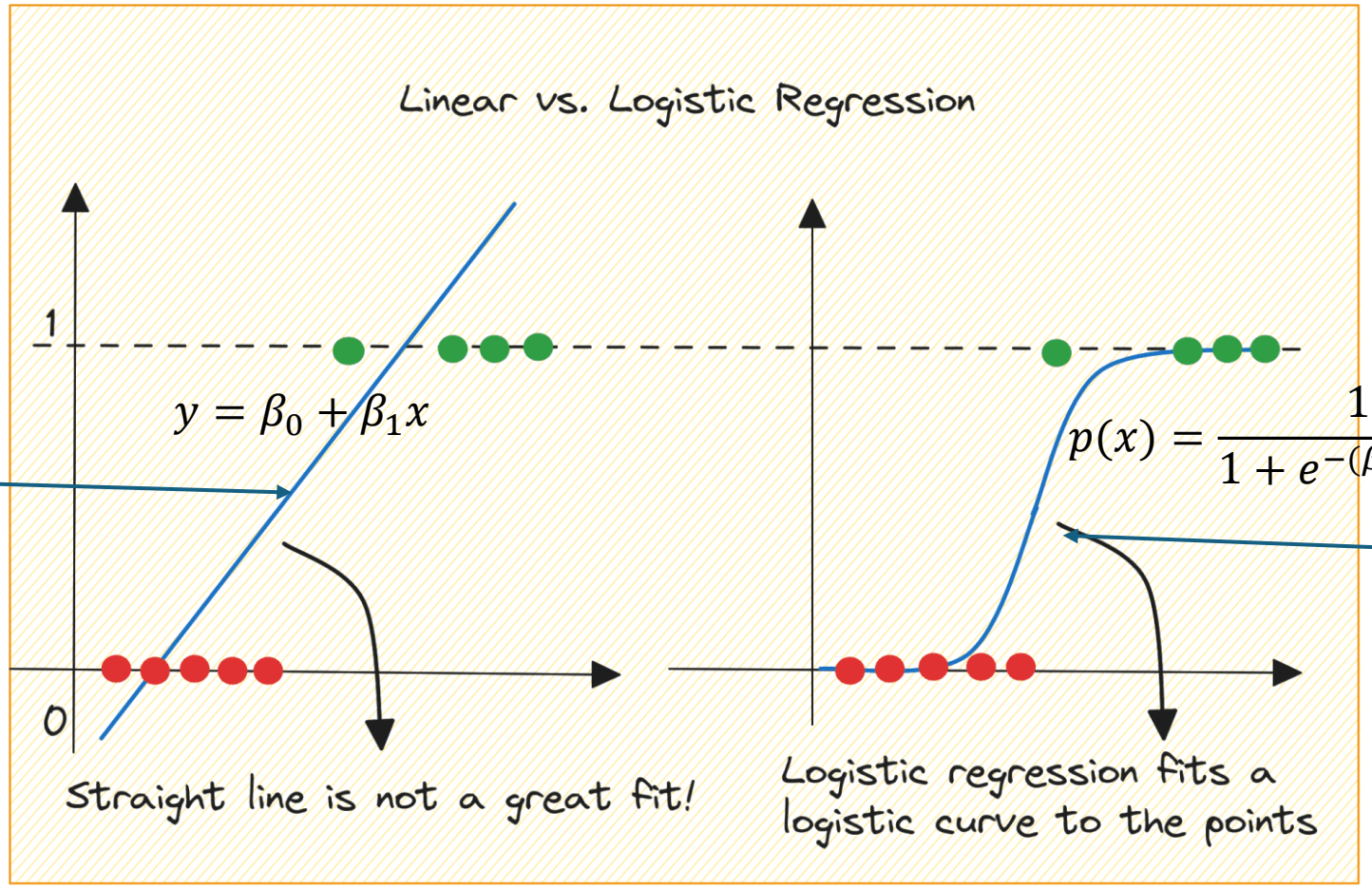
Hypothesis Test of Coefficients

The process involves making a hypothesis about the population parameter(s) and then using sample data to either reject or fail to reject the null hypothesis, based on the observed sample statistics.

Research Question	Is there a linear relationship?	Is there a positive linear relationship?	Is there a negative linear relationship?
Null Hypothesis	$\beta_1 = 0$	$\beta_1 = 0$	$\beta_1 = 0$
Alternative Hypothesis	$\beta_1 \neq 0$	$\beta_1 > 0$	$\beta_1 < 0$
Type of Test	Two-tailed, non-directional	Right-tailed, directional	Left-tailed, directional

Statistical Regression Analysis such as StatsModels can determine both the coefficients and its significant level.

Predicting a binary output



Unconstrained line

Constrained curve

Connection:

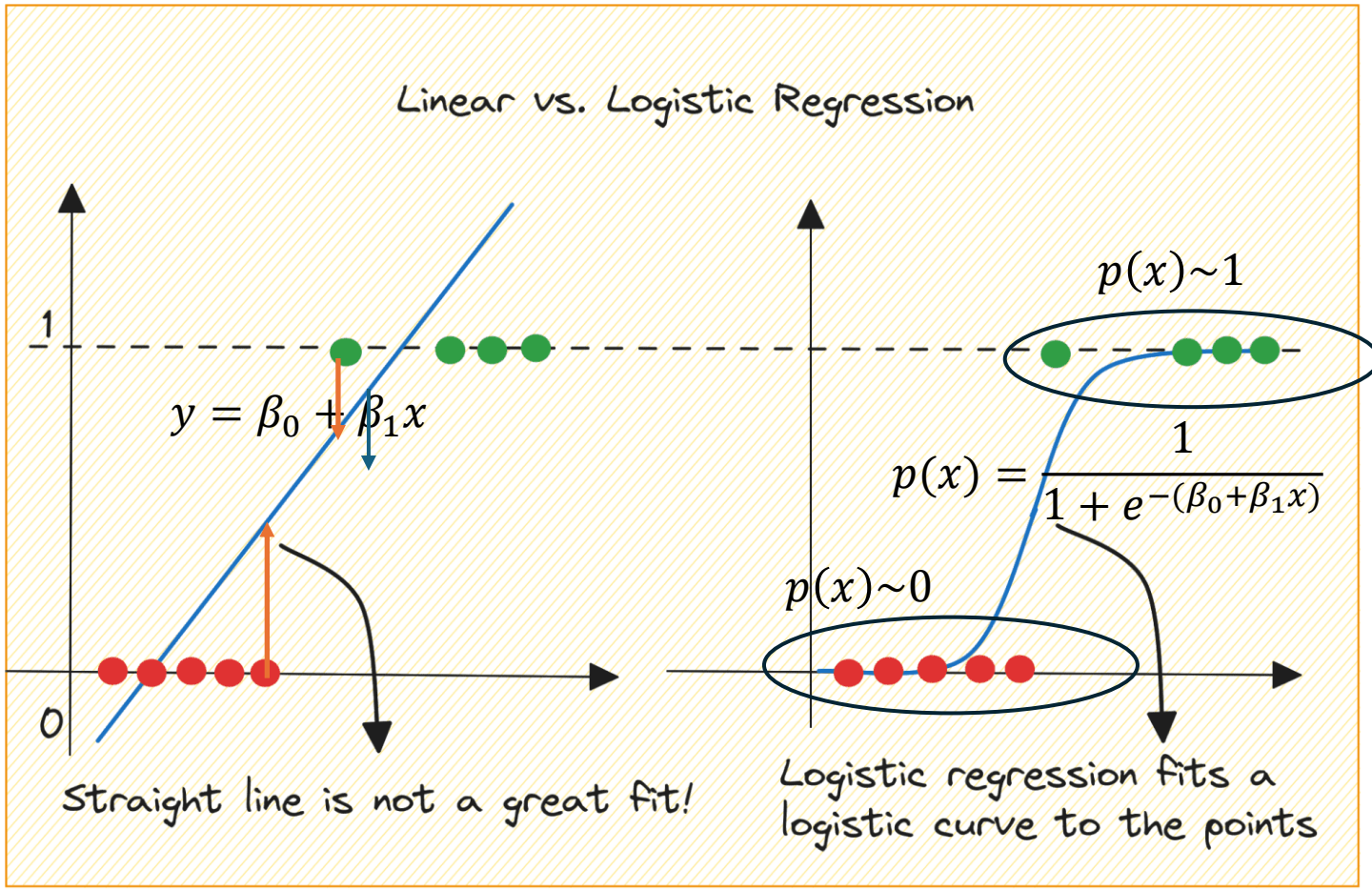
$$y = log(odds) = log\left(\frac{p(x)}{1 - p(x)}\right) \qquad odds = \frac{p(x)}{(1 - p(x))}$$

Odds vs. Probability

Example:
One want to go jogging when there is **no rain**.
According to the weather forecast, the **probability** for rain is 30%.
The **odds** for no-rain are 0.7 / 0.3 = 2.3 = 2.3 : 1.
The **log odds** are the logarithm of the odds = log(2.3) = 0.36.

Predicting a binary output

Find (β_0, β_1) that minimizes $SSE = \sum(y_i - \hat{y}_i)^2$



Find (β_0, β_1) that maximizes $\prod_{y_i=1} p(x_i) \prod_{y_i=0} (1 - p(x_i))$

Connection:

$$y = log(odds) = log\left(\frac{p(x)}{1 - p(x)}\right) \qquad odds = \frac{p(x)}{(1 - p(x))}$$

Likelihood Function

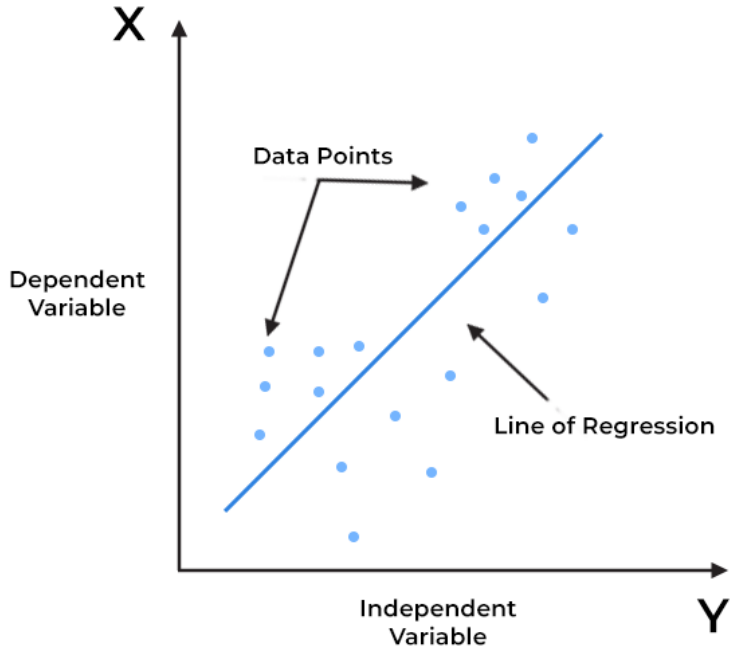
Goal: Find Beta to **maximize** this function.

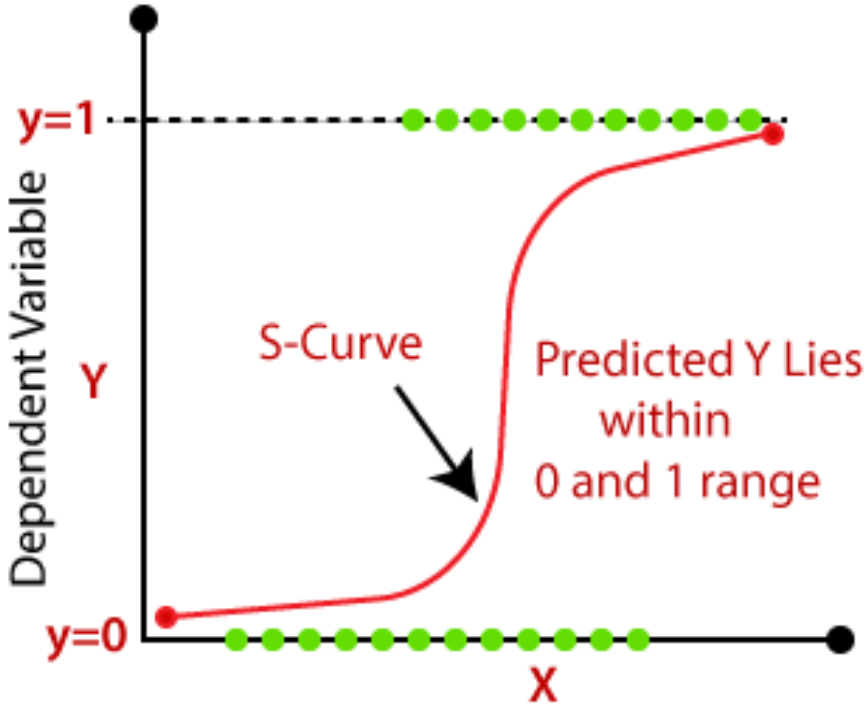
$$L(\beta) = \prod_{s\ in\ y_i = 1} p(x_i) \ * \prod_{s\ in\ y_i = 0} (1 - p(x_i))$$

Maximize the Likelihood objective = minimize the negative log-likelihood (NLL) objective.

$$J(\beta) = -\sum_{i=1}^{n} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

Interpreting Coefficients





$$y(x) = \beta_0 + \beta_1 x$$

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

One unit increase in x => β_1 increase in $y(x)$

One unit increase in x => β_1 increase in log odds.

Multiple Feature Cases

$$\hat{y} = \beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

$$\hat{y} = \sum_{i=0}^{n} \beta_i X_i$$

$$p_i = \frac{1}{1 + e^{-\sum_{i=0}^{n} \beta_i X_i}}$$

Model Training (Fitting)

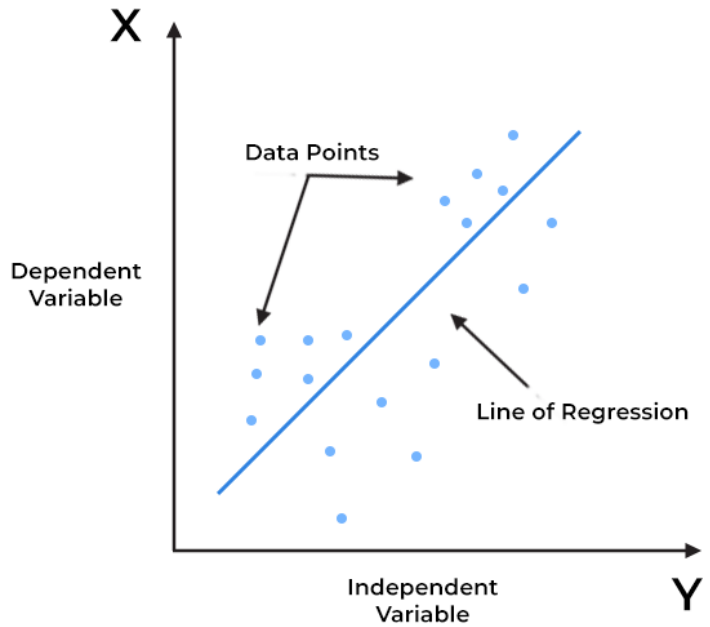
Loss Minimization – Linear Regression

$$\hat{y} = \beta_0 + \beta_1 x$$

Minimize the loss value by changing the trainable parameters (?)

$$J(\beta) = \sum_{i=1} (y_i - \hat{y}_i)^2$$

Training Data, n pairs
x_i, y_i



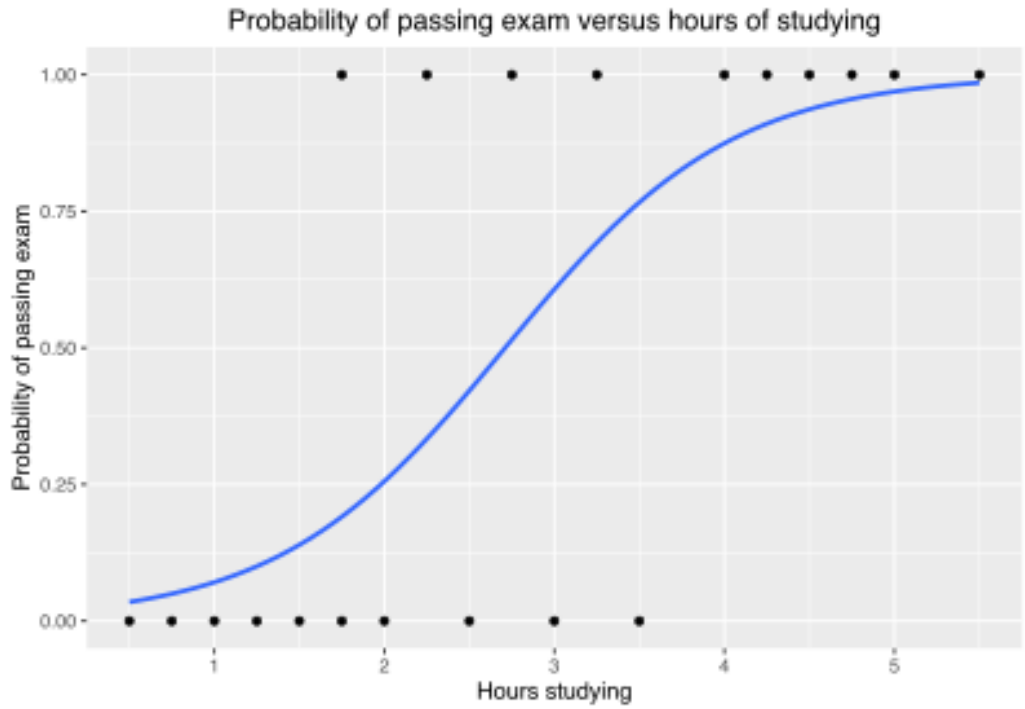
Loss Minimization – Logistic Regression

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Minimize the loss value by changing the trainable parameters (?)

$$J(\beta) = -\sum_{i=1}^{n} \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right)$$

Training Data, n pairs
x_i, y_i



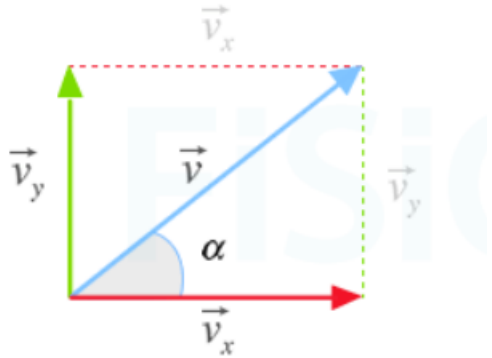
Generally

$$\hat{\beta} = argmax_{\beta} J(\beta)$$

Solving for $\hat{\beta}$ - Function Minimization

$$\frac{\partial J(\beta)}{\partial \beta} = 0$$

$$\begin{bmatrix} \frac{\partial J(\beta)}{\partial \beta_0} \\ \cdots \\ \frac{\partial J(\beta)}{\partial \beta_n} \end{bmatrix} = 0$$



Linear Regression

$$\frac{\partial J(\beta)}{\partial \beta} = 0$$

Closed-Form /Analytical Solution

$$\hat{\beta} = (X^T . X)^{-1} . X^T . Y$$

Closed-form /analytical solutions are rare for complex models.

Logistic Regression

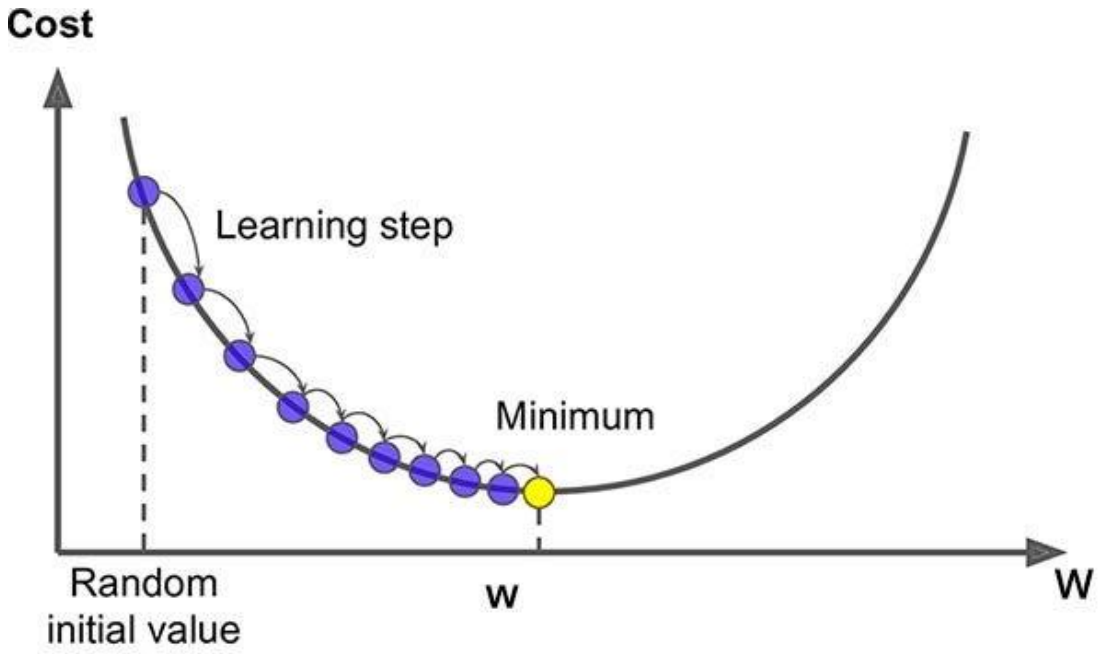
$$\frac{\partial J(\beta)}{\partial \beta} = 0$$

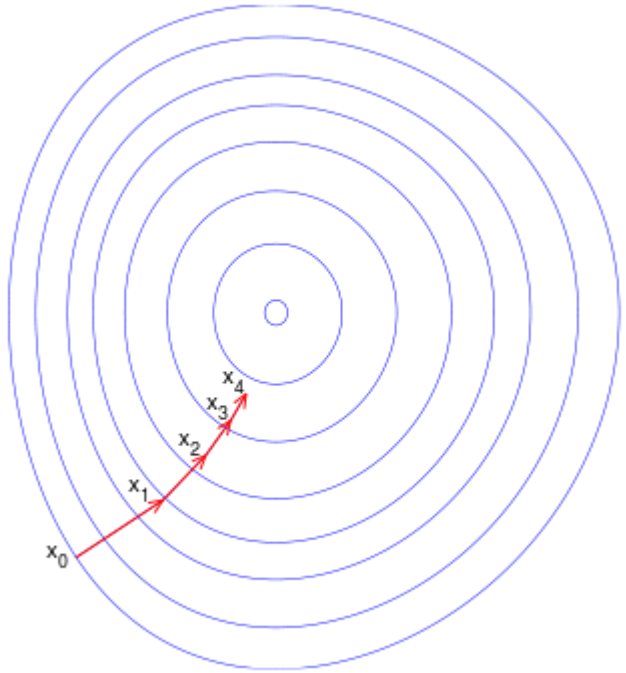
Closed-Form /Analytical Solution

Closed-form /analytical solutions are rare for complex models.

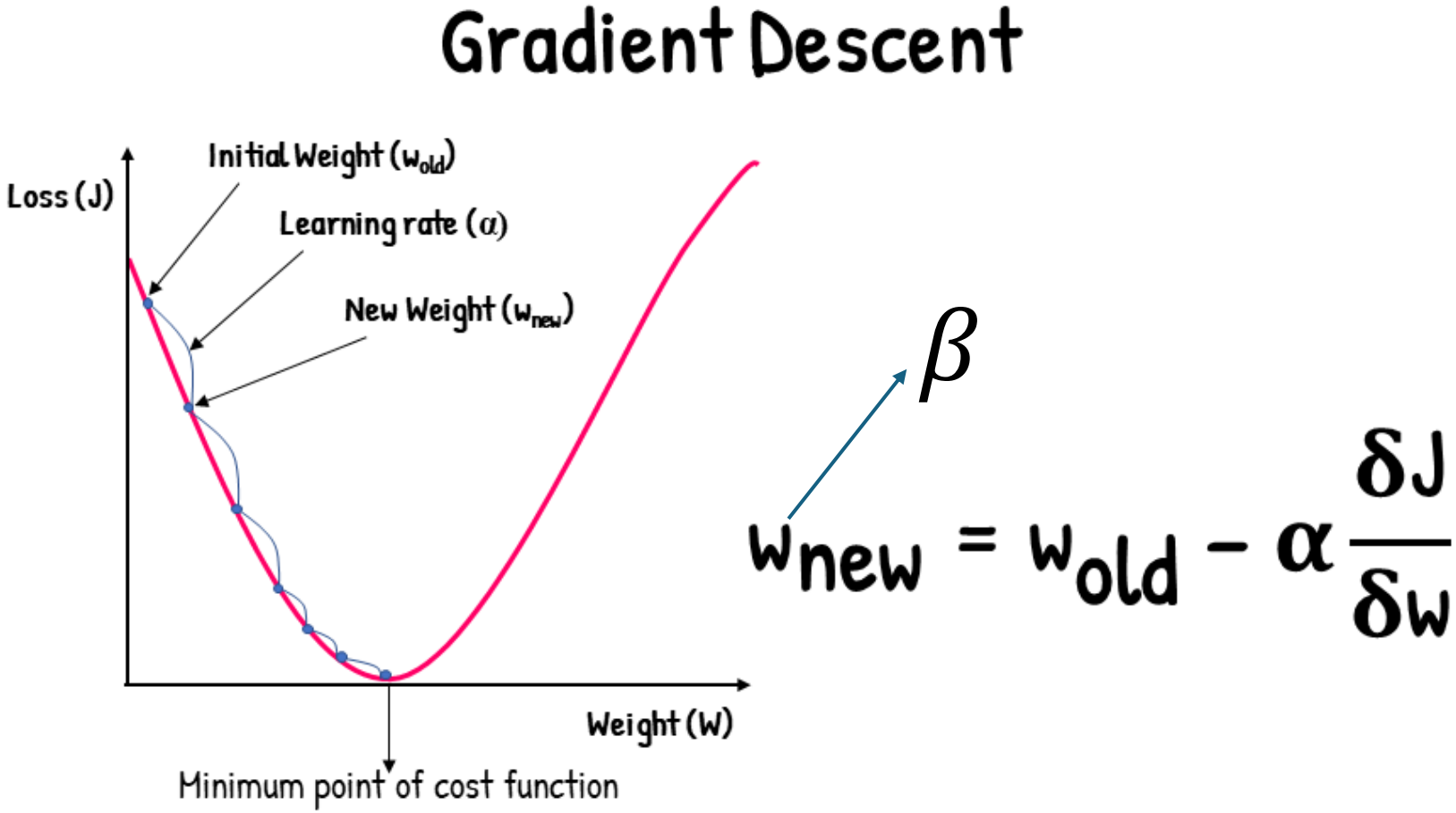
But we have an iterative solution approximation!

$\frac{\partial J(\beta)}{\partial \beta}$ tells us the direction, the opposite of which the function decreases. (i.e., don't need to tell me where the answer is, but just tell me which direction I should go!!!)





Gradient Descent – In a nutshell



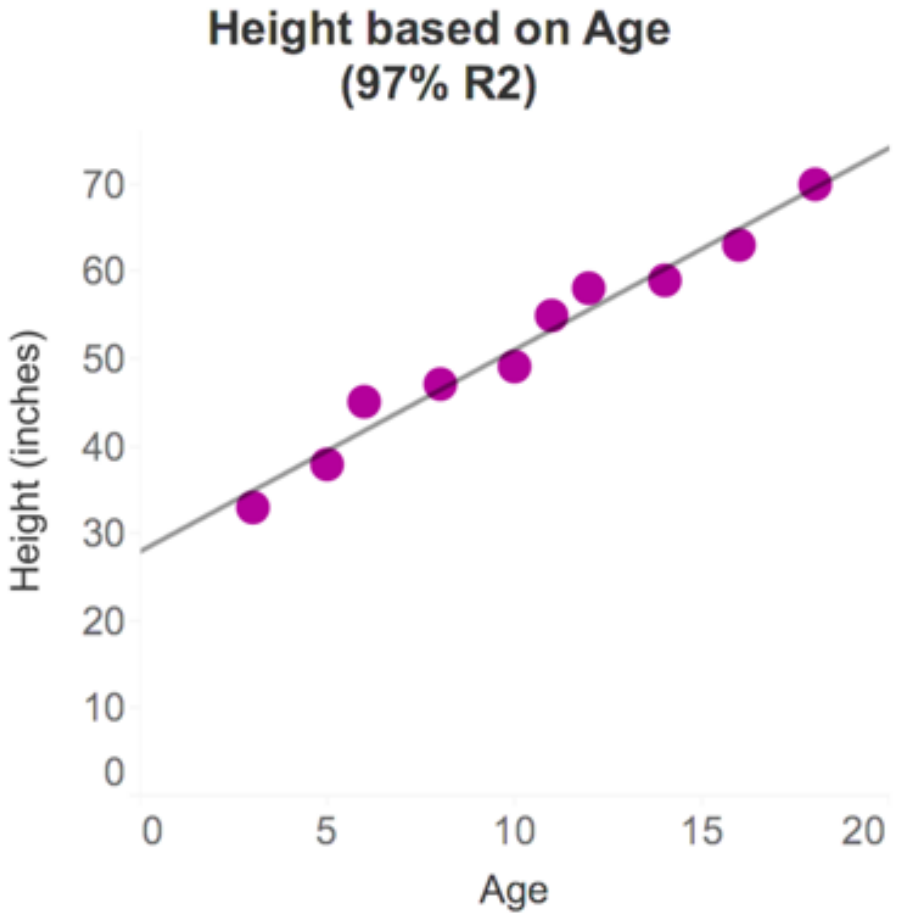
Evaluation Metrics

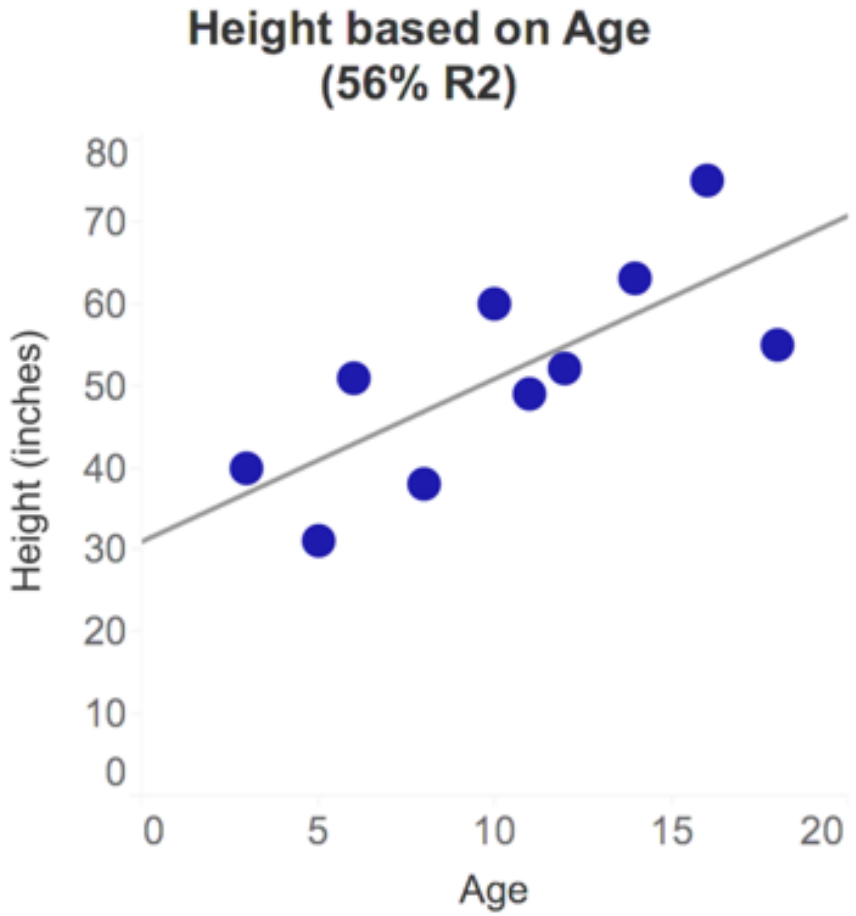
Regression - comparative

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	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$

https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/#google_vignette

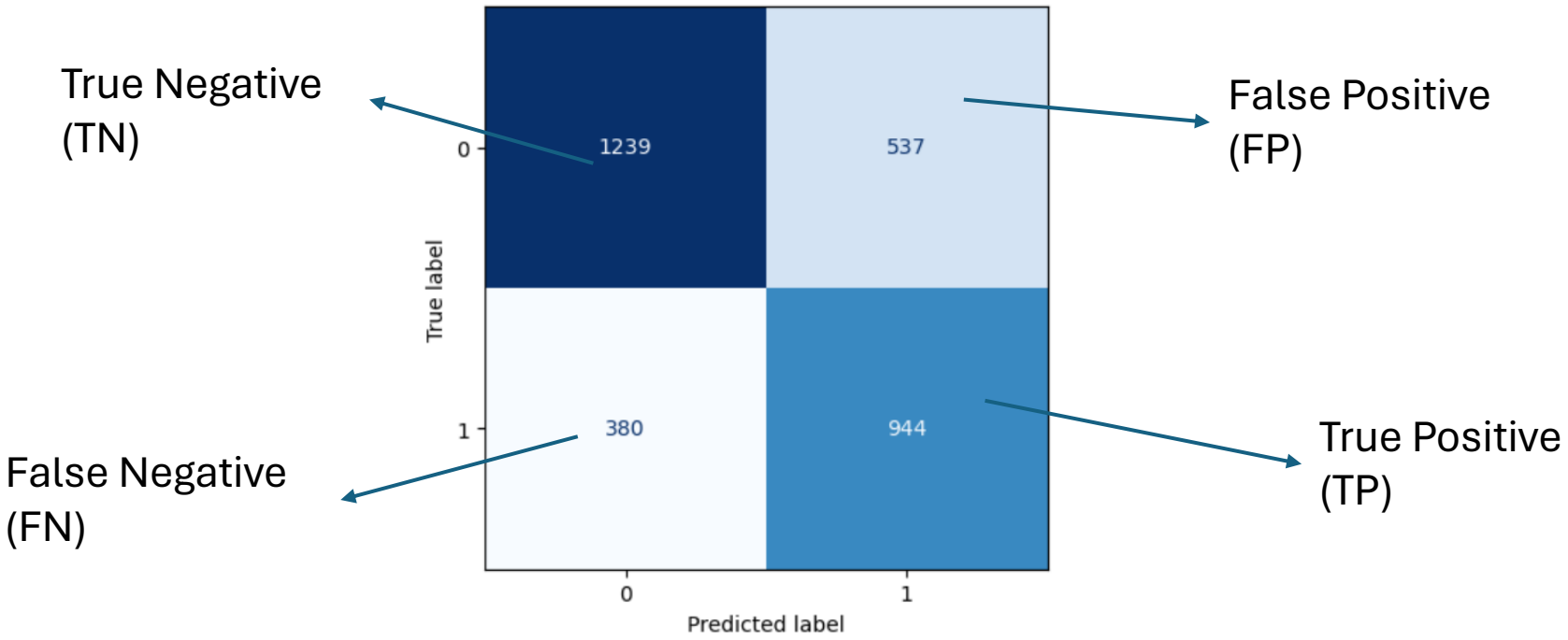
R-Squared – more interpretable





Classification – Confusion Matrix

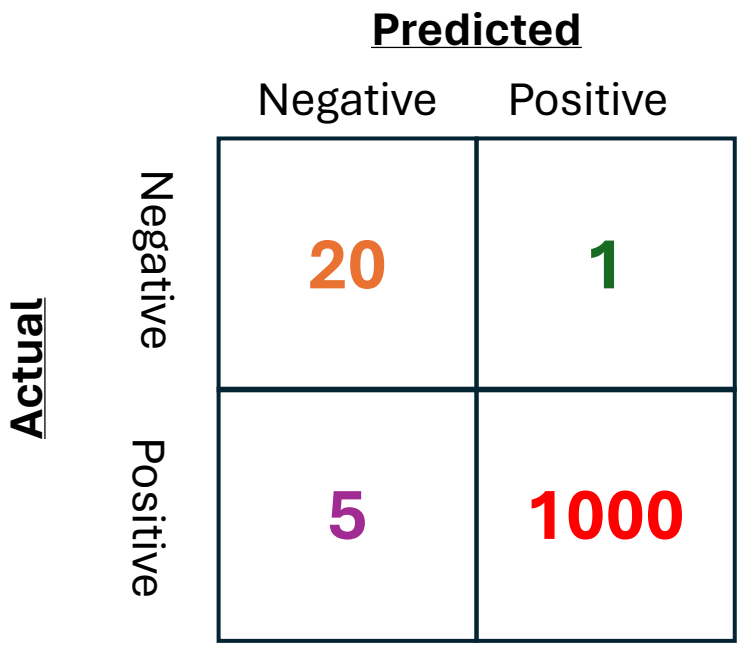
- Confusion Matrix: A table used to describe the performance of a classification model. It presents the number of true positives, true negatives, false positives, and false negatives.



Classification – Accuracy

- Accuracy is defined as the number of correct predictions over the total predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

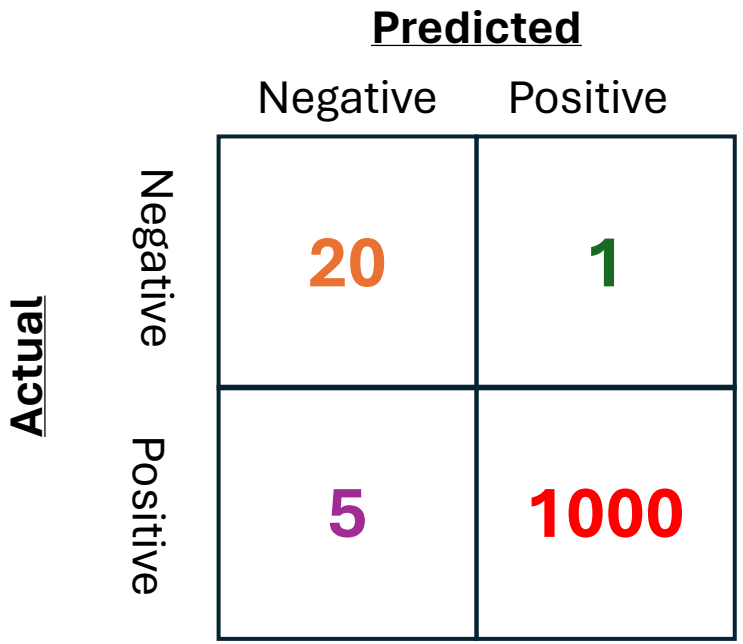


$$Accuracy = \frac{1000 + 20}{1000 + 20 + 1 + 5}$$

Classification – Precision

- Precision: It is the proportion of true positive predictions out of all positive predictions made by the model. Precision is useful when the cost of false positives is high (e.g., fraud).

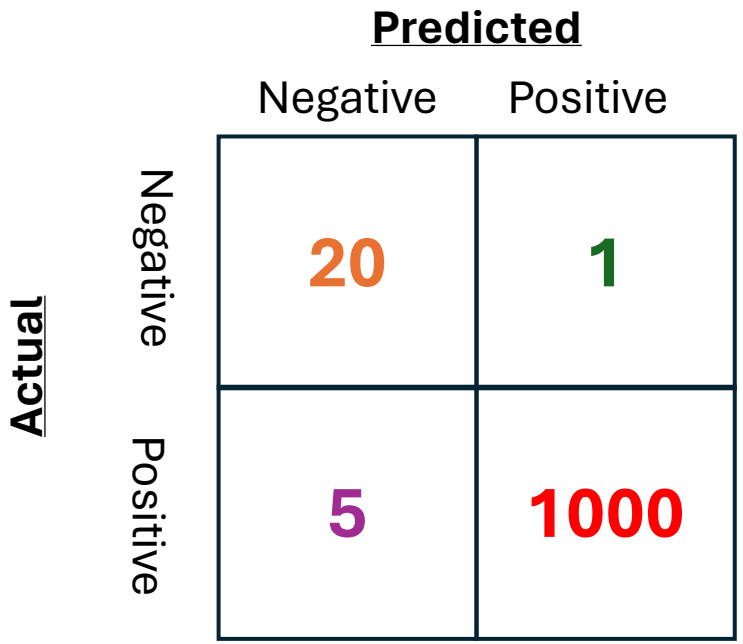
$$Precision = \frac{TP}{TP + FP}$$



$$Precision = \frac{1000}{1000 + 1}$$

Classification – Recall (Sensitivity)

- Recall: It is the proportion of true positive instances that were correctly identified by the model. Recall is useful when the cost of false negatives is high (e.g., Customer Churn Prediction).

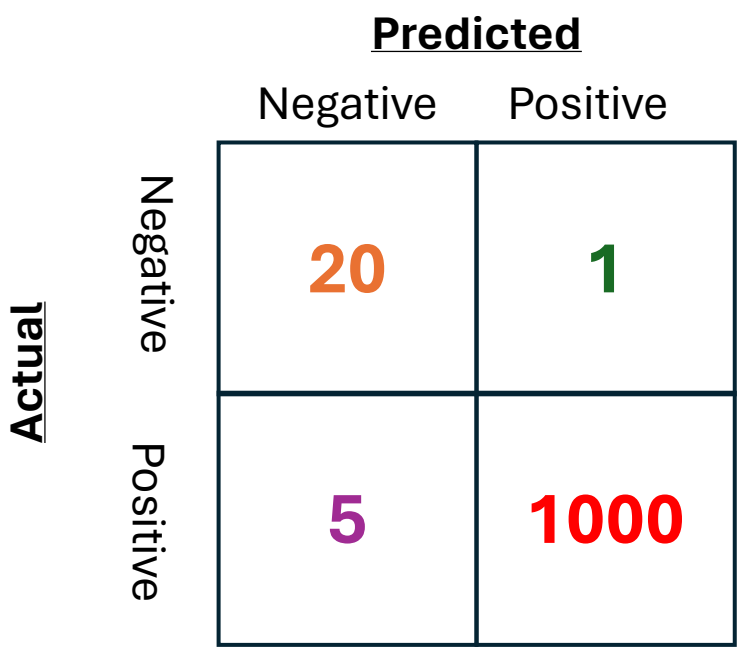


$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{1000}{1000 + 5}$$

Classification – F1 Score

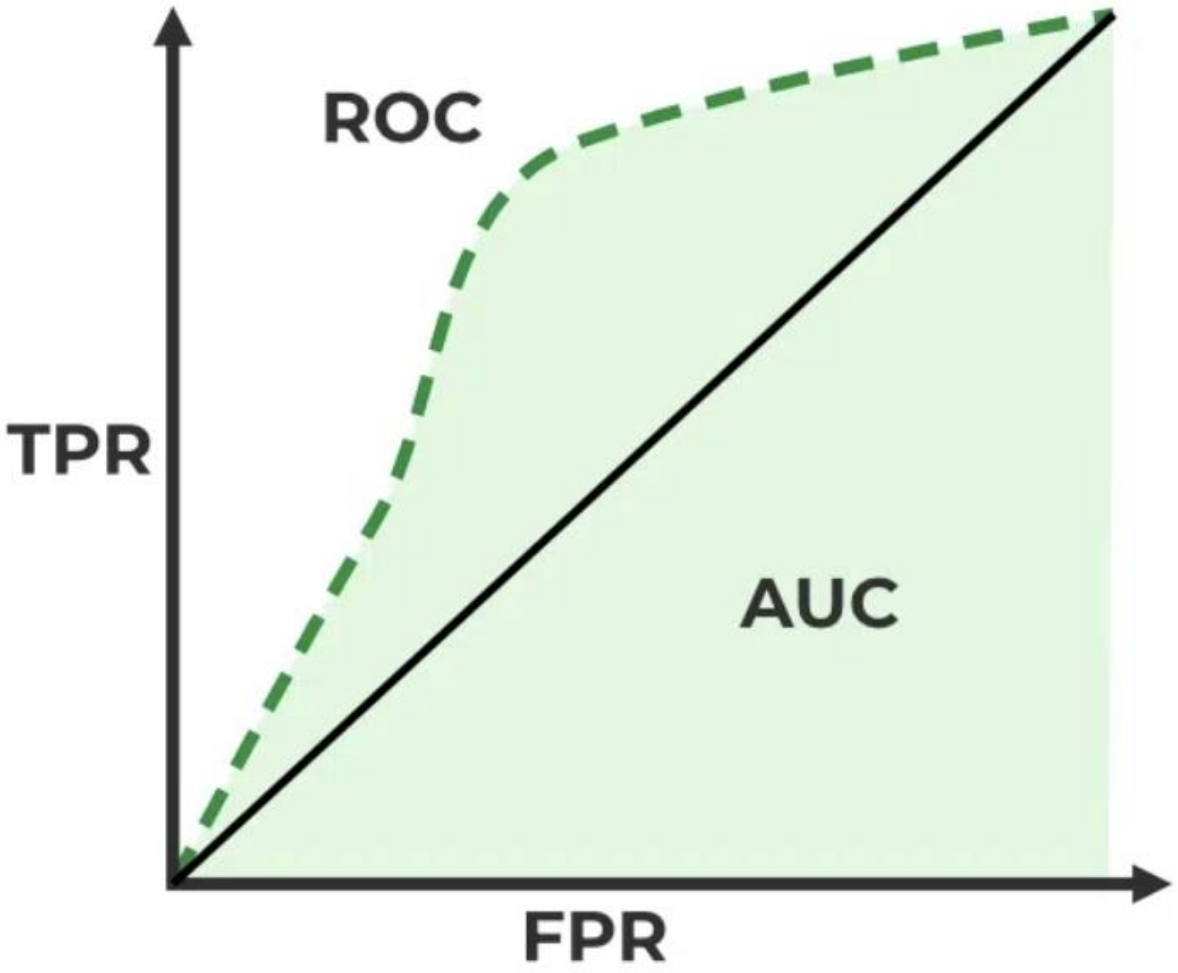
- Recall: It is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is a useful metric when there is an uneven class distribution.



$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

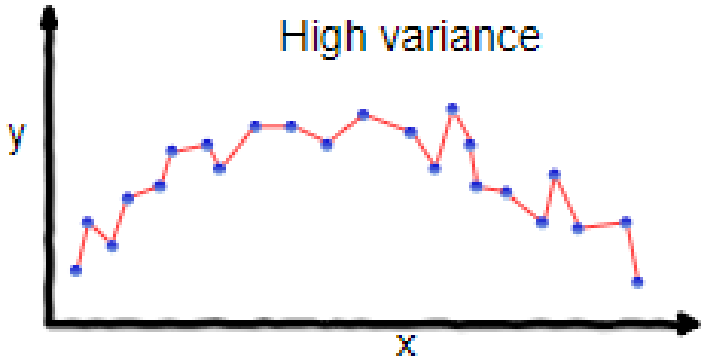
Classification - ROC vs. AUC

- The ROC (Receiver Operating Characteristic) curve is a graphical plot that illustrates the performance of a binary classification model across different threshold settings.
- The AUC, as discussed earlier, summarizes the performance of a classifier across all possible threshold settings. It provides a single scalar value representing the overall quality of the model's predictions, regardless of the threshold chosen. A higher AUC indicates better model performance.



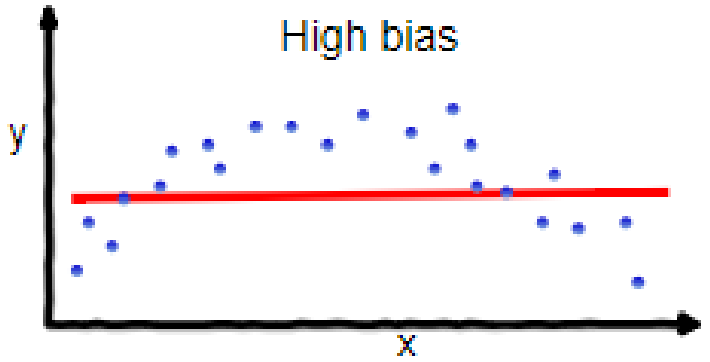
Machine Learning Fundamental

Underfitting vs. Good Fit vs. Overfitting



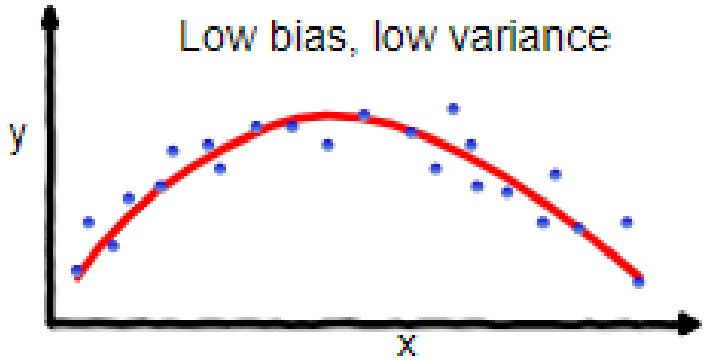
Too high complexity

overfitting



Too low complexity

underfitting

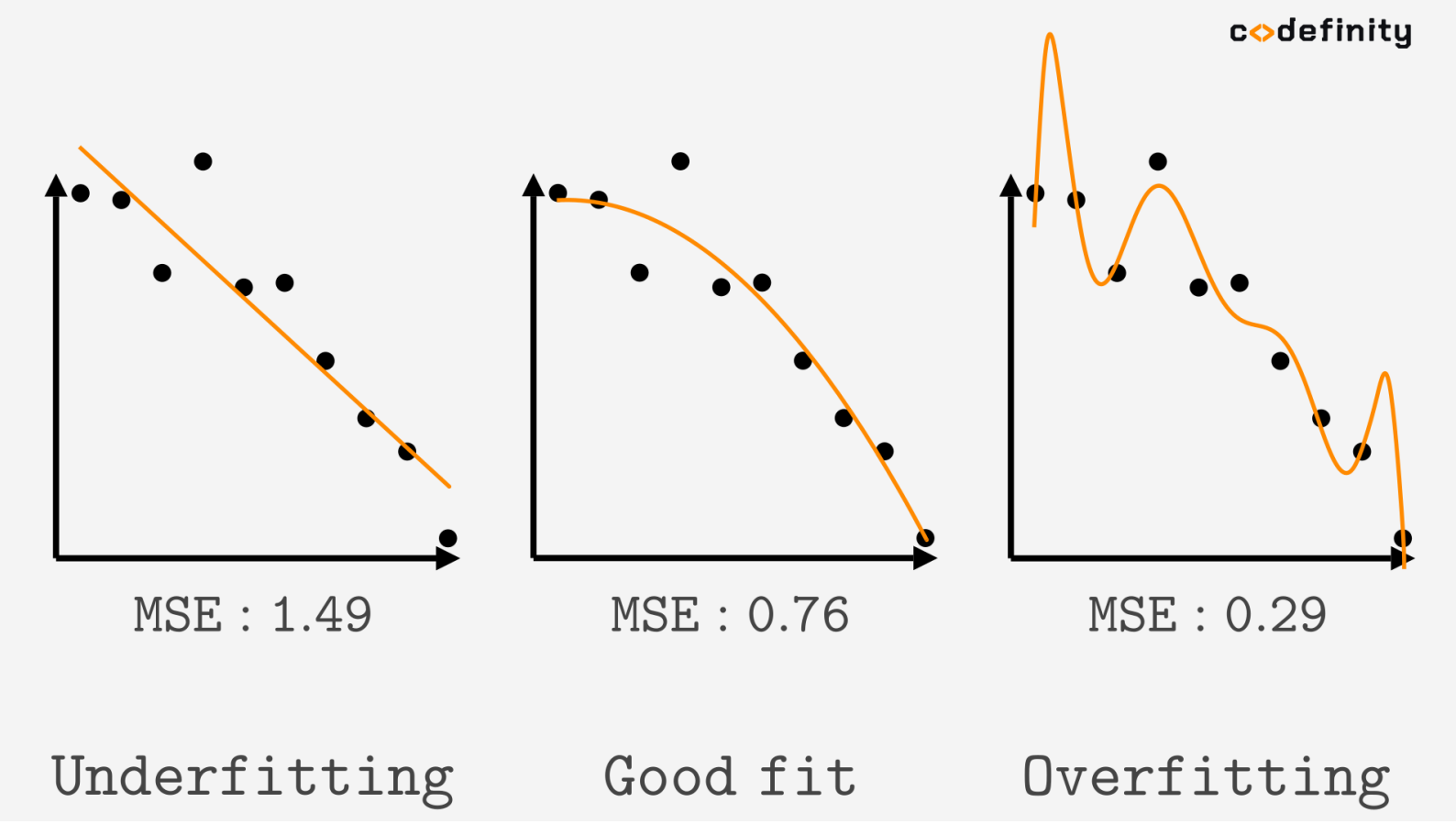


Good complexity

Good balance

https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

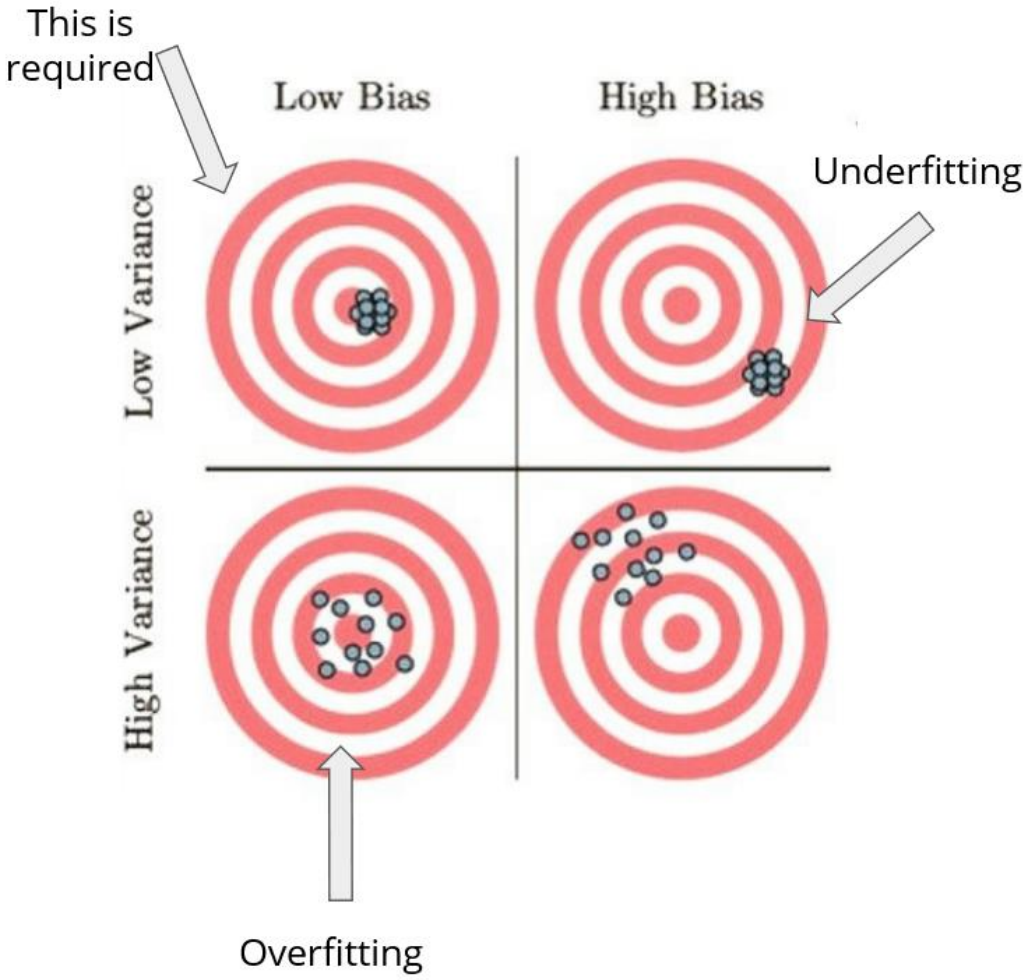
Underfitting vs. Good Fit vs. Overfitting



https://codefinity.com/courses/v2/b22d1166-efda-45e8-979e-6c3ecfc566fc/a00949b2-6964-47ec-af3d-71b51cf5f40a/5bdd281d-badc-4579-9e33-9c841b58206c

Variance-Bias Analysis

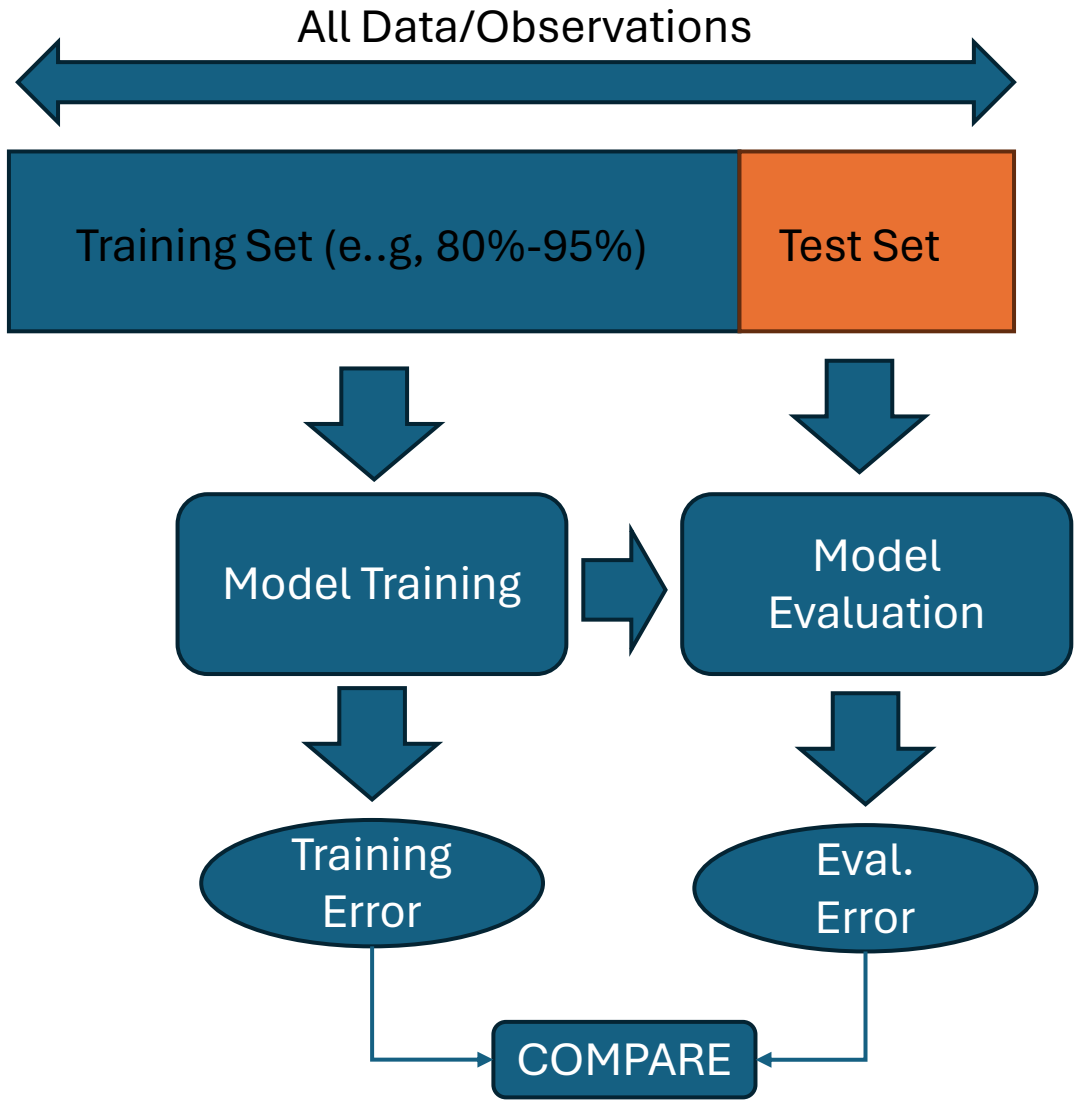
- Good fit: low-variance, low-bias model. Accurate, consistent predictions for small changes in inputs.
- Underfitting: high-bias model. Inaccurate, consistent (low-var.)/inconsistent(high-var) predictions for small changes in inputs.
- Overfitting: high-variance model. Accurate (low-bias)/inaccurate(high-bias), inconsistent predictions for small changes in inputs.



Causes of Overfitting

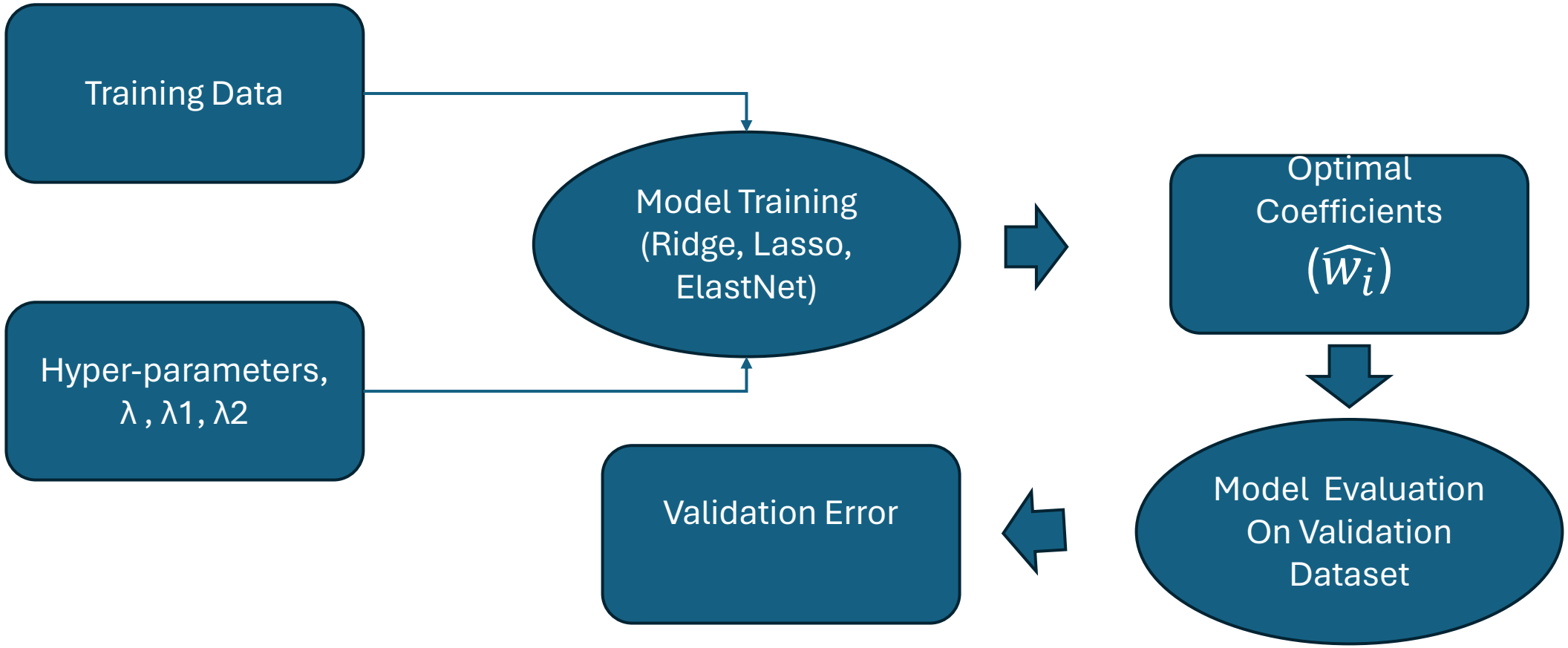
- Overfitting occurs when a machine learning model learns the training data too well, capturing noise or random fluctuations that don't represent the underlying relationships in the data. Here are some common causes of overfitting:
  - Complexity of the Model: e.g., Simple vs. multiple LR.
  - Insufficient Training Data: e.g., less observations, more parameters.
  - Irrelevant Features: e.g., not correlation between inputs and output.
  - Duplicate features: e.g., high correlated inputs.
  - etc.

Model Evaluation



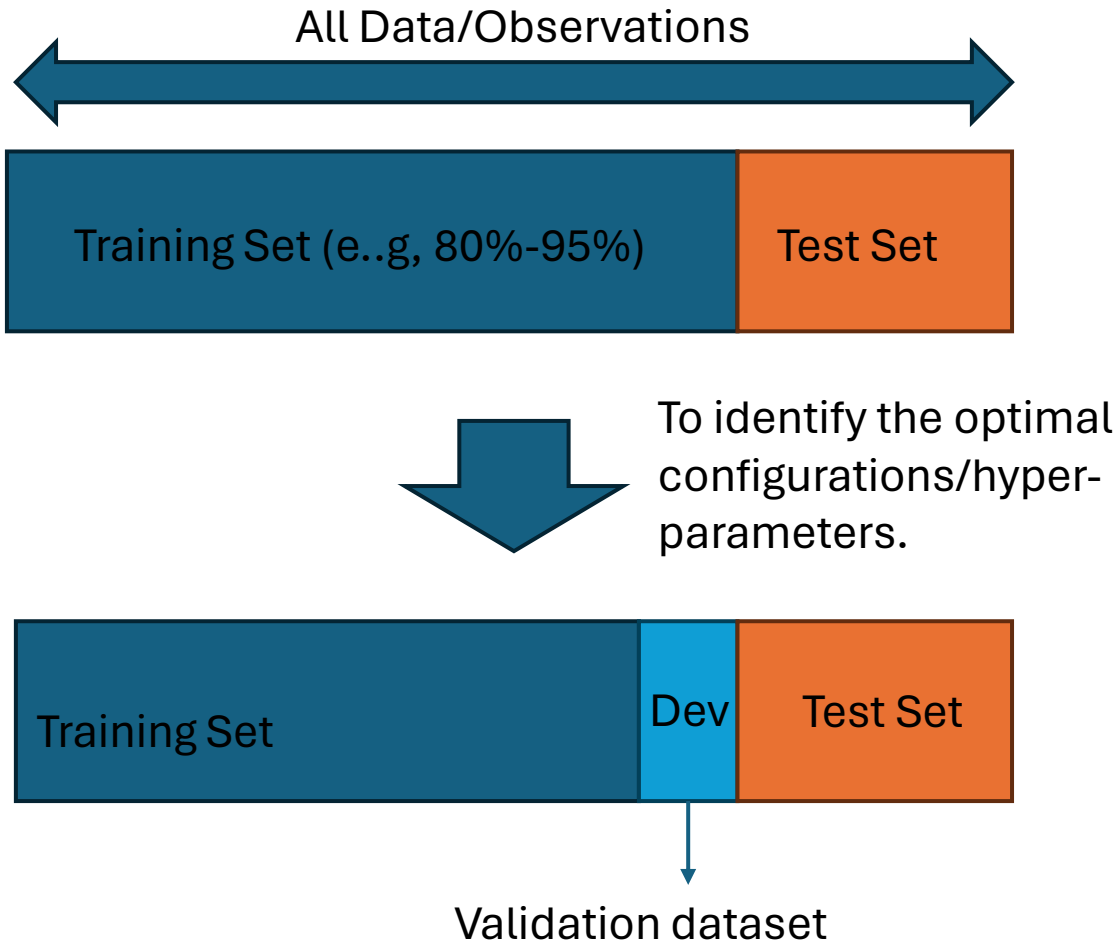
- Training error (low) and eval. error (high): **overfitting**/underfitting/good fit?
- Training error (high) and eval. error (high)=> Overfitting/**underfitting**/good fit?
- Training error (low) and eval. error (low)=> Overfitting/underfitting/**good fit**?

Hyper-parameters Selection



Select the choice of hyper-parameters, which gives the lowest validation error.

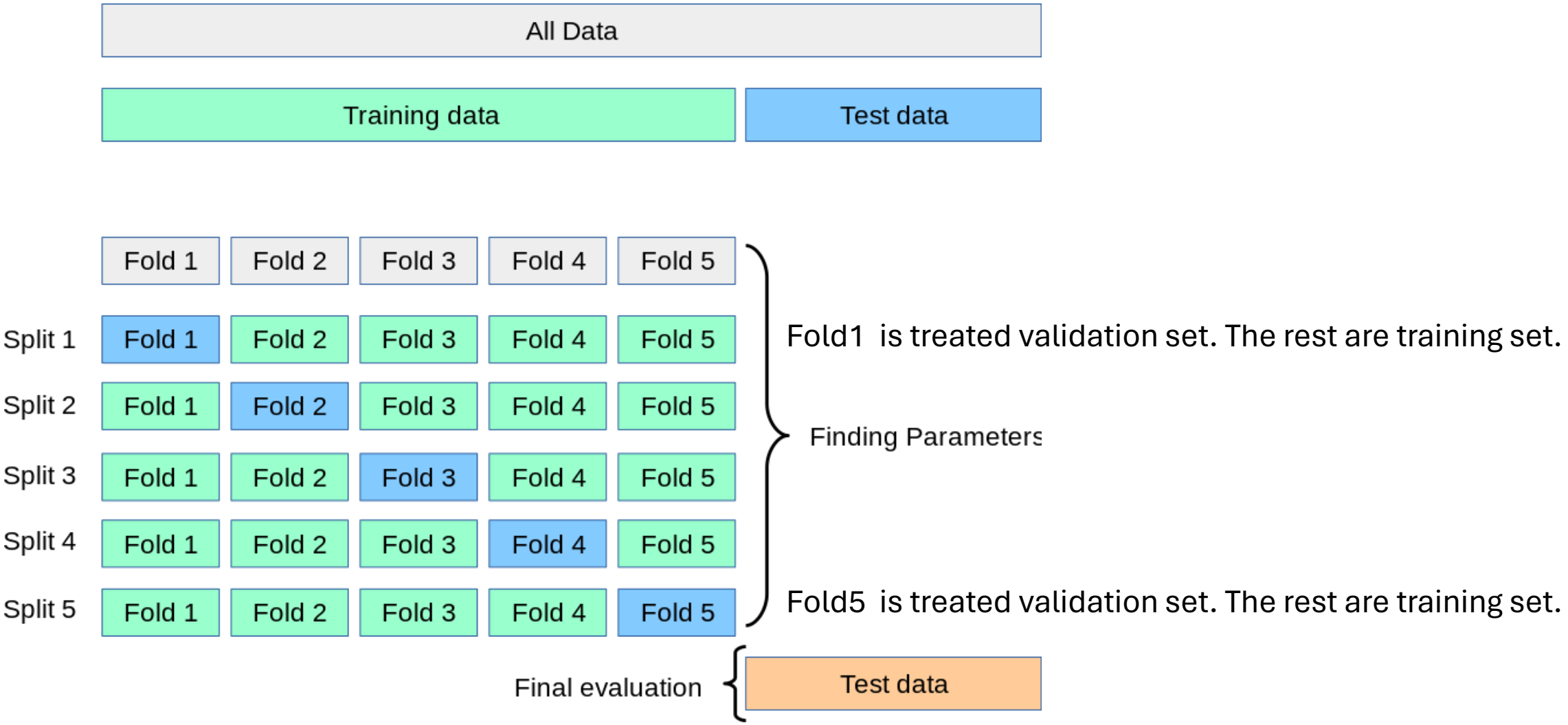
What is Validation Dataset?



1. If we need to identify the optimal hyper-parameters, we need to split the training part into training and validation sets.
2. For each value of hyper-parameter (HP), train a ML model and computer error on validation set.
3. Pick the optimal value of HP, which has the lowest validation error.
4. Retrain the final model on with the optimal value of HP on a combined training + validation sets.
5. Compute the final error on the test set.

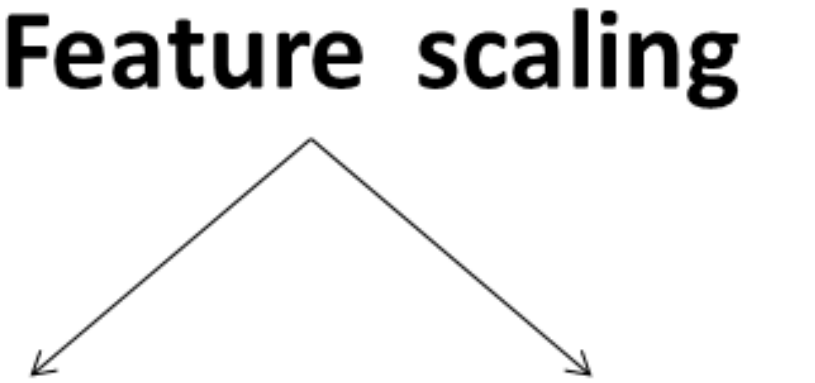
This process is called cross-validation.

K-Fold Cross-Validation



Feature Engineering

Scaling – Normalization and standardization



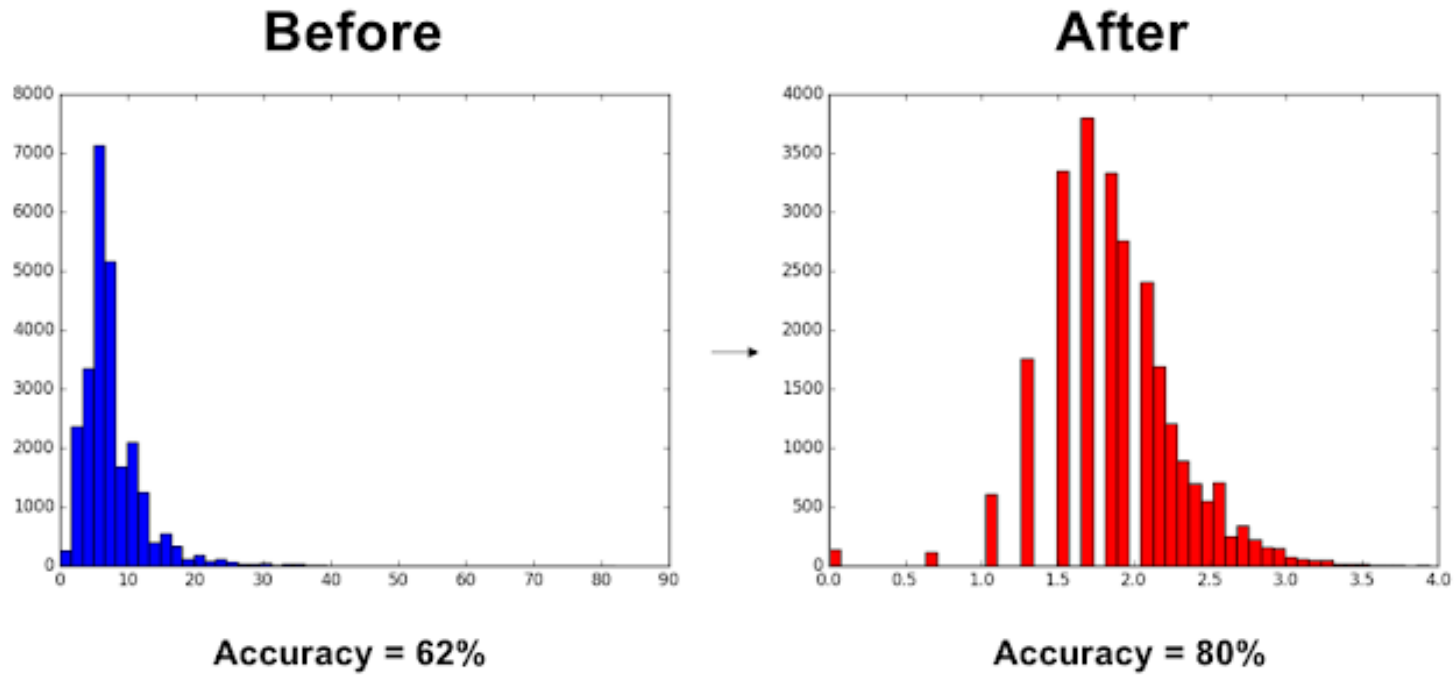
$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad X' = \frac{\text{X} - \text{Mean}}{\text{Standard deviation}}$$

https://medium.com/aimonks/normalizing-scaling-and-standardizing-data-a-crucial-prelude-to-machine-learning-success-a71307f7c1f0

1. Raw data seldom arrives in a pristine, ready-to-use state; instead, it often exhibits a multitude of variations in scale, distribution, and units.
2. To harness the full potential of data for building accurate and robust machine learning models, practitioners frequently employ techniques like normalizing, scaling, and standardizing.
3. These techniques play a pivotal role in ensuring that the data is appropriately prepared for analysis.

Feature Engineering

- Box-cox transformations involve raising the data to various powers to transform it. Box-cox transformations can normalize data, make it more linear, or decrease the complexity.



Feature Encoding - Dealing with Discreet Inputs

- Nominal variables: These variables represent categories without any inherent order or ranking. Examples include colors, types of fruits, or gender.
- Ordinal variables: These variables have a natural ordering or ranking among the categories. For example, ratings such as "low," "medium," and "high" represent ordinal variables.

One-Hot Encoding

id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Ordinal Encoding

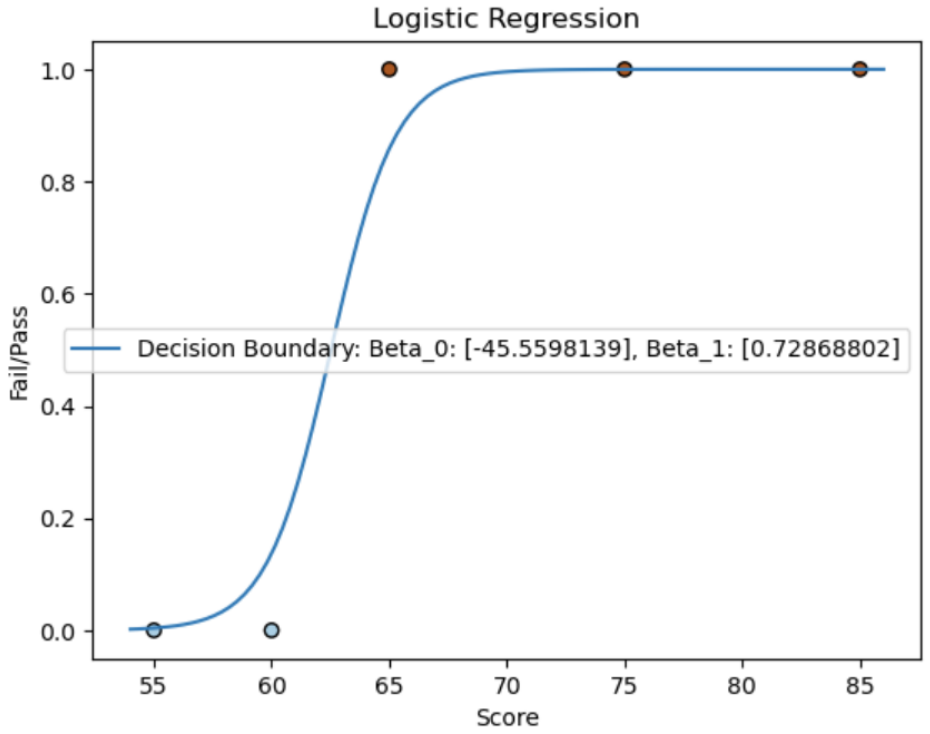
Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

Other Machine Learning Algorithms

Parametric vs. Non-parametric Models?

Exam Score	Pass/Fail
65	Pass
75	Pass
55	Fail
85	Pass
60	Fail

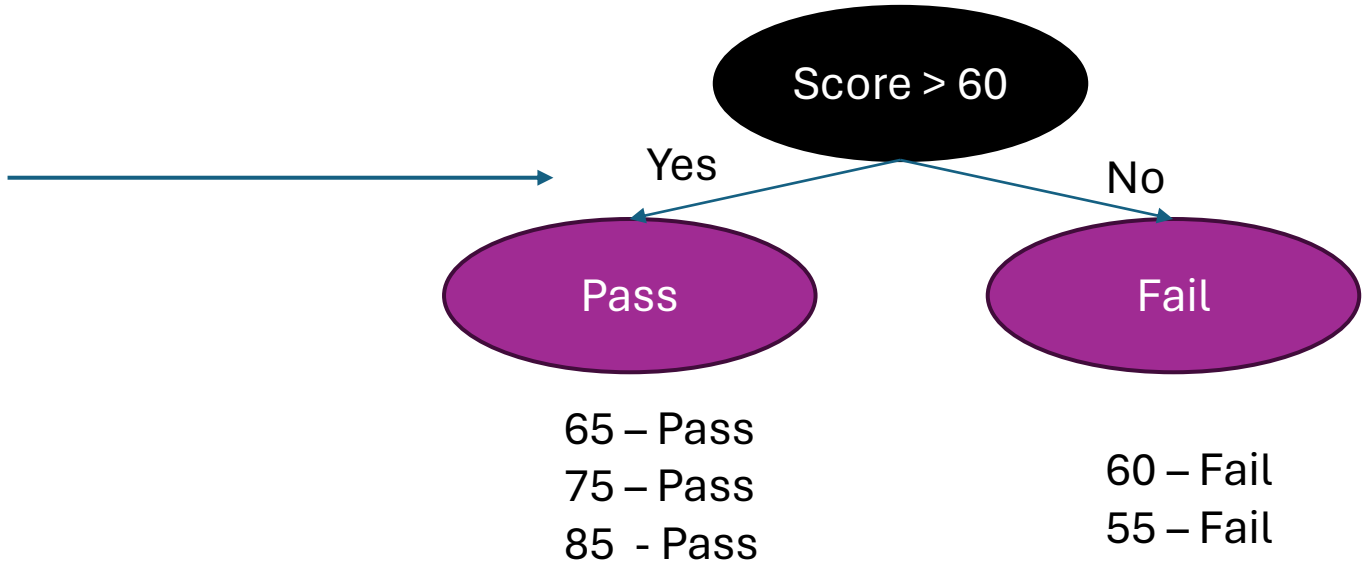
$$Y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$



What are the parameters ?

Parametric vs. Non-parametric Models?

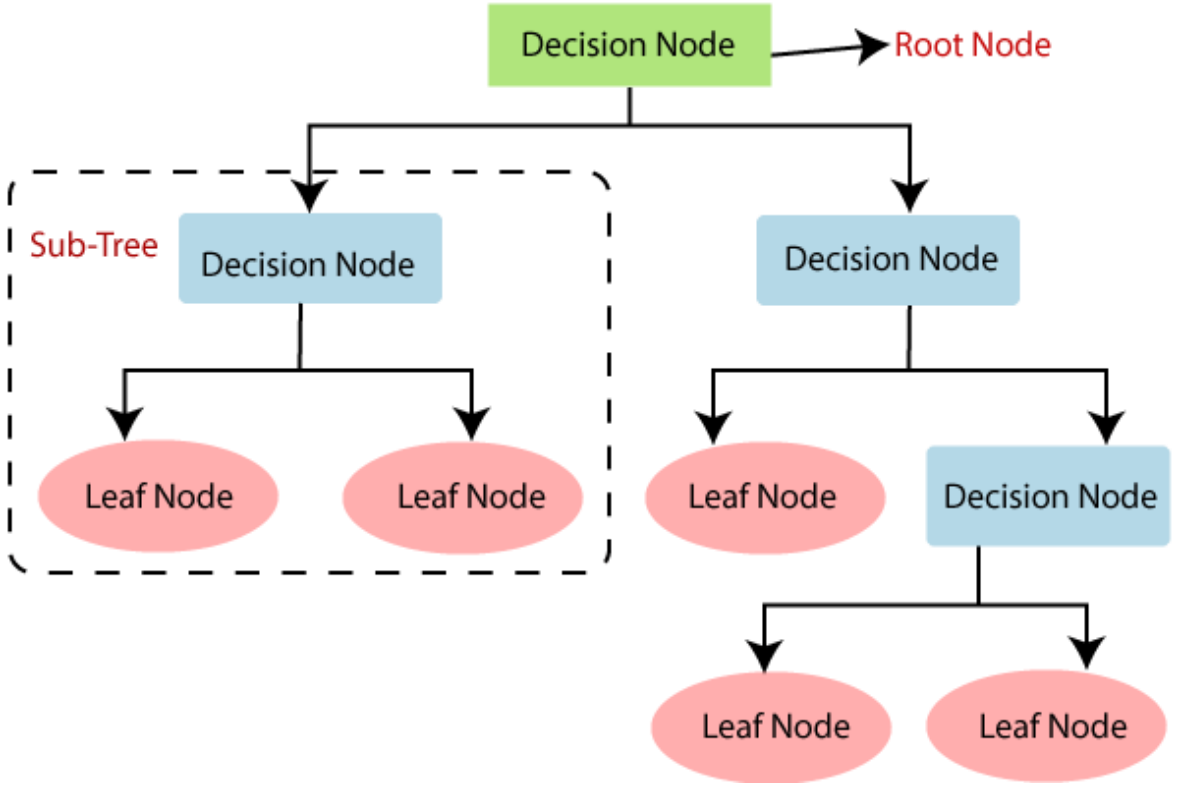
Exam Score	Pass/Fail
65	Pass
75	Pass
55	Fail
85	Pass
60	Fail



What are the parameters ?

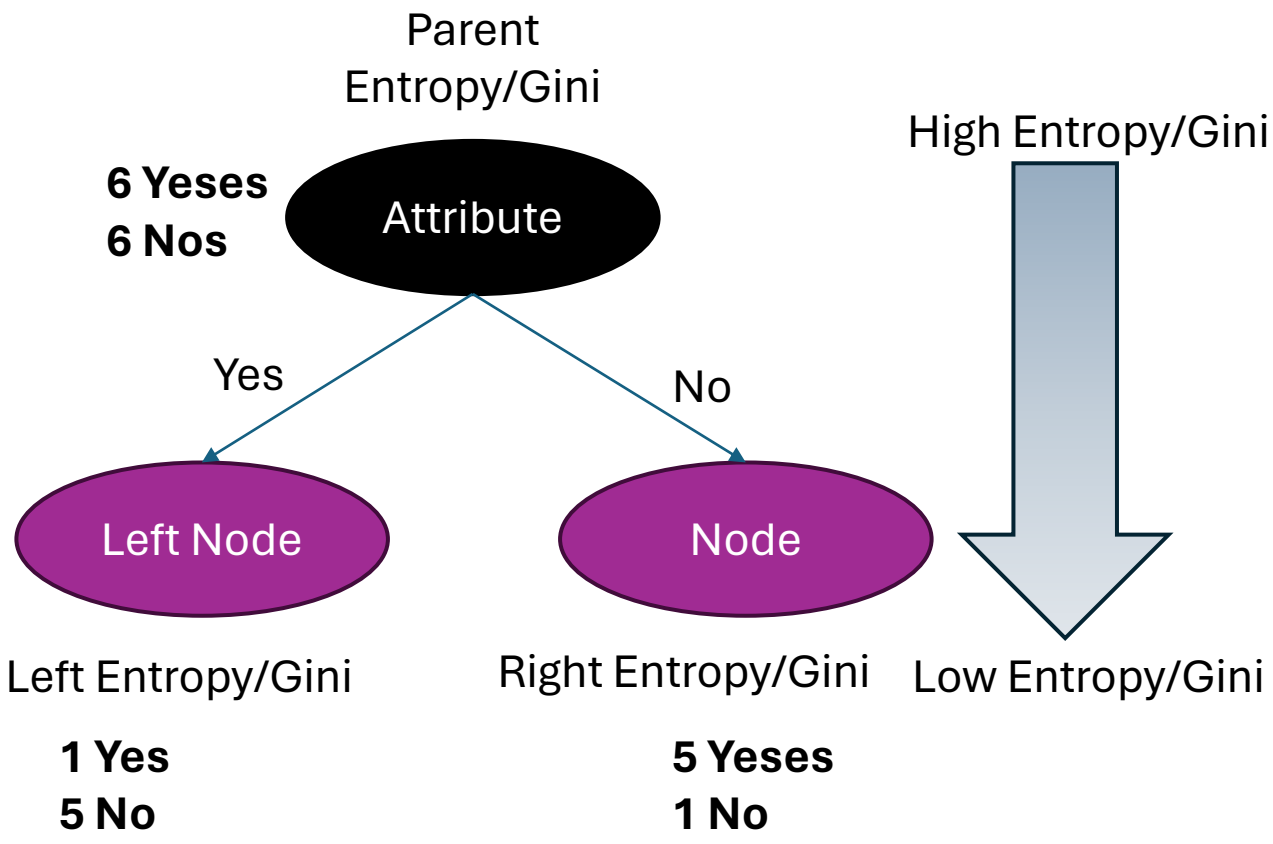
Multiple Features

Outlook	Temperature	Humidity	Windy	Play Tennis?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



A LG model will require multiple parameters, while a DT requires multiple branches.

Identify the best attributes for subtrees



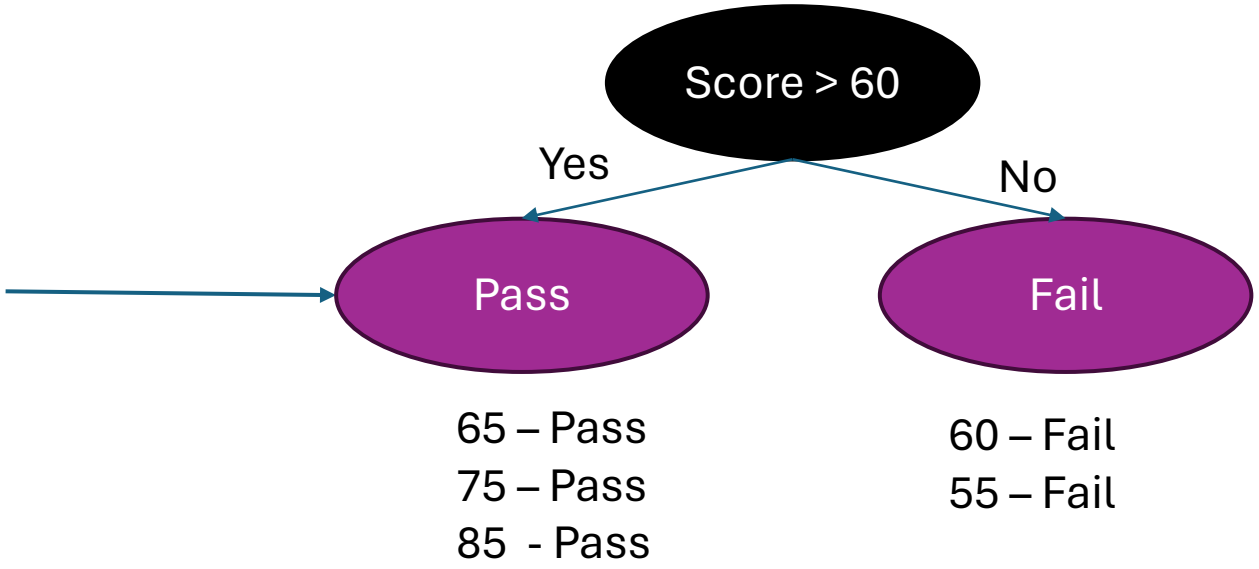
Attribute	IG
Attribute-1	…
…	…
Attribute-n	…

Pick the attribute with the highest IG.

IG = Entropy(parent) – Entropy(children)

Continuous attribute case

Exam Score	Pass/Fail
65	Pass
75	Pass
55	Fail
85	Pass
60	Fail



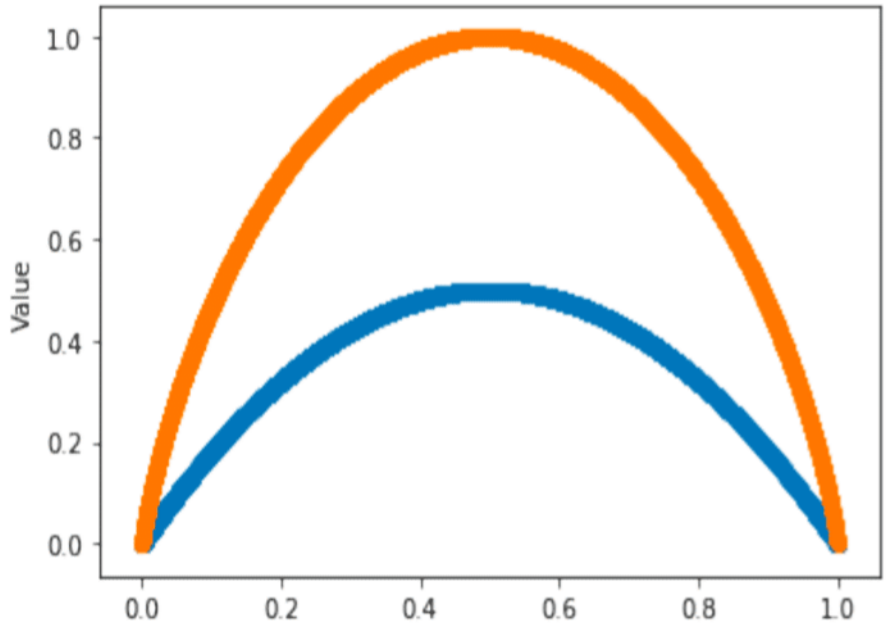
Split Value	60	…	85
IG	…		…

Pick the split point with the highest IG.

Entropy vs. Gini

$$Gini = 1 - \sum_j p_j^2$$

$$Entropy = -\sum_j p_j \log_2 p_j$$



Entropy (orange) vs. Gini (blue) for a binary distribution (i.e., Bernoulli)

Decision Tree Variants

1. **CART (Classification and Regression Trees)**: CART is a versatile decision tree algorithm that can be used for both classification and regression tasks. It works by recursively partitioning the feature space into regions, aiming to minimize impurity (Gini index or entropy) in each split.
2. **ID3 (Iterative Dichotomiser 3)**: ID3 is one of the earliest decision tree algorithms primarily designed for classification tasks. It builds the tree by iteratively selecting the best attribute to split on based on information gain.
3. **C4.5**: C4.5 is an extension of ID3 that addresses some of its limitations. It handles both discrete and continuous attributes, deals with missing values, and incorporates pruning to reduce overfitting.

Hyper-parameters

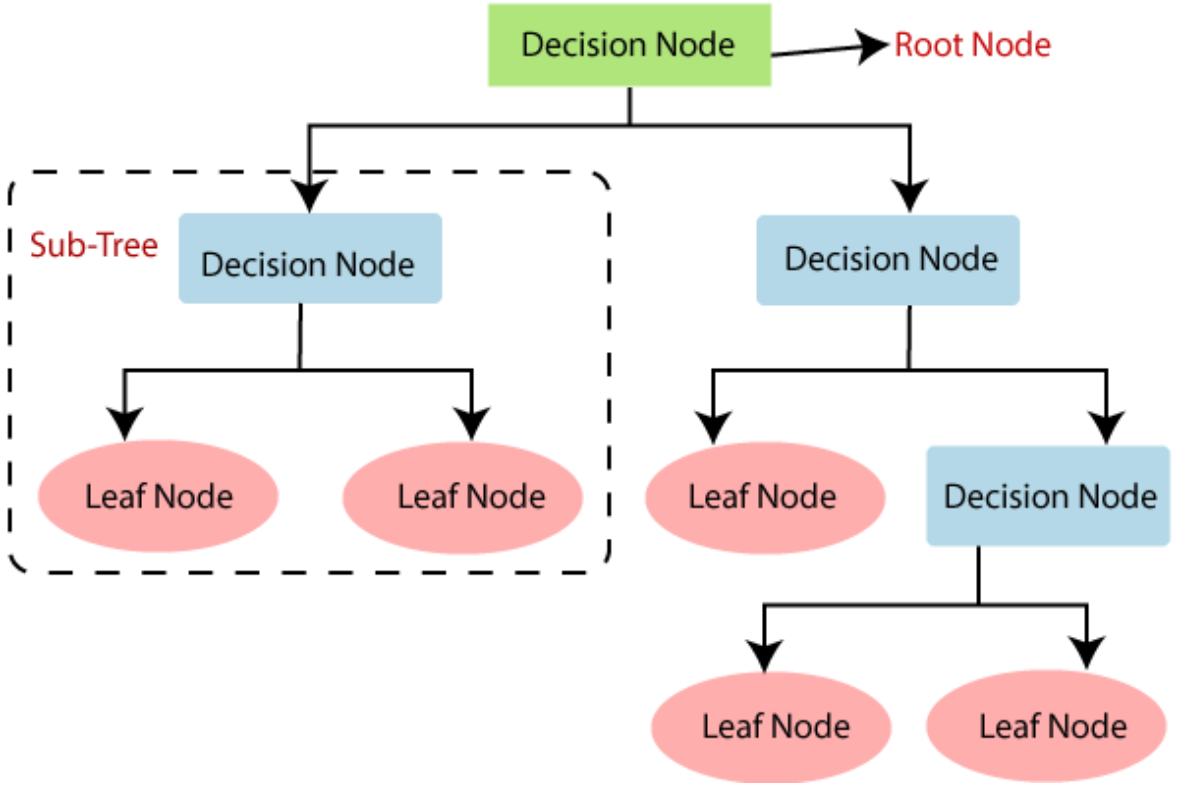
1. **Criterion**: The function used to measure the quality of a split. Common criteria are "gini" for the Gini impurity and "entropy" for information gain.
2. **Max Depth**: The maximum depth of the decision tree. It limits the number of levels in the tree and helps prevent overfitting.
3. **Min Samples Split**: The minimum number of samples required to split an internal node. It controls the creation of new nodes by requiring a minimum amount of data in each split.
4. **Min Samples Leaf**: The minimum number of samples required to be at a leaf node. It controls the size of terminal nodes and prevents the model from creating nodes with very few samples.
5. **Max Features**: The number of features to consider when looking for the best split. It helps to reduce overfitting and speed up the training process by randomly selecting a subset of features at each split.

- Etc.

Ensemble Machine Learning

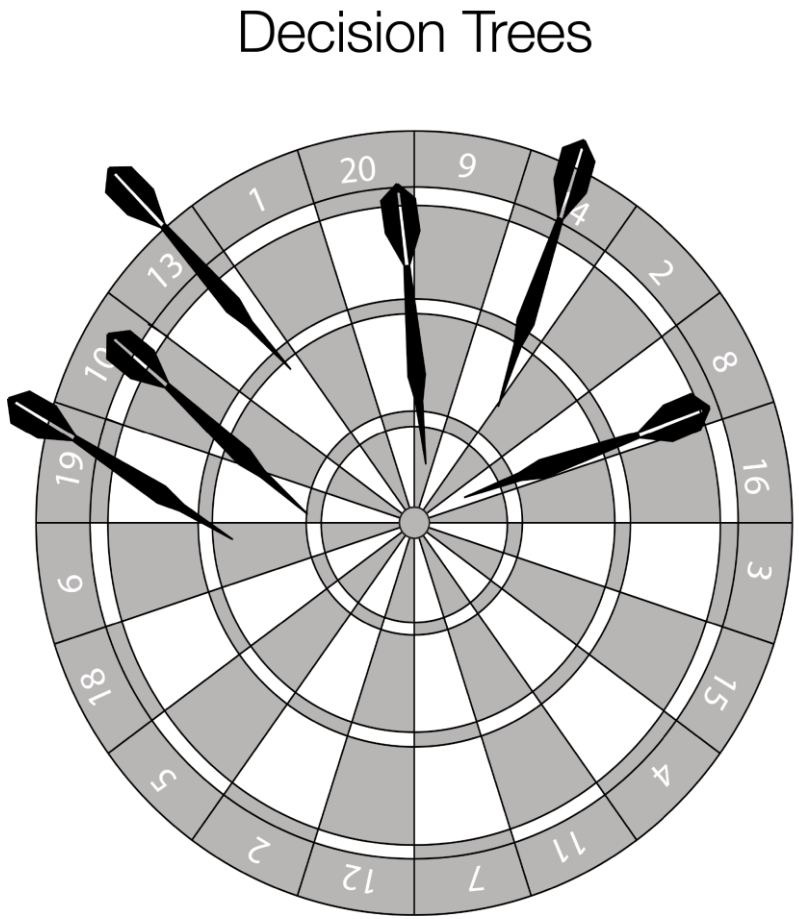
Decision Tree

Outlook	Temperature	Humidity	Windy	Play Tennis?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



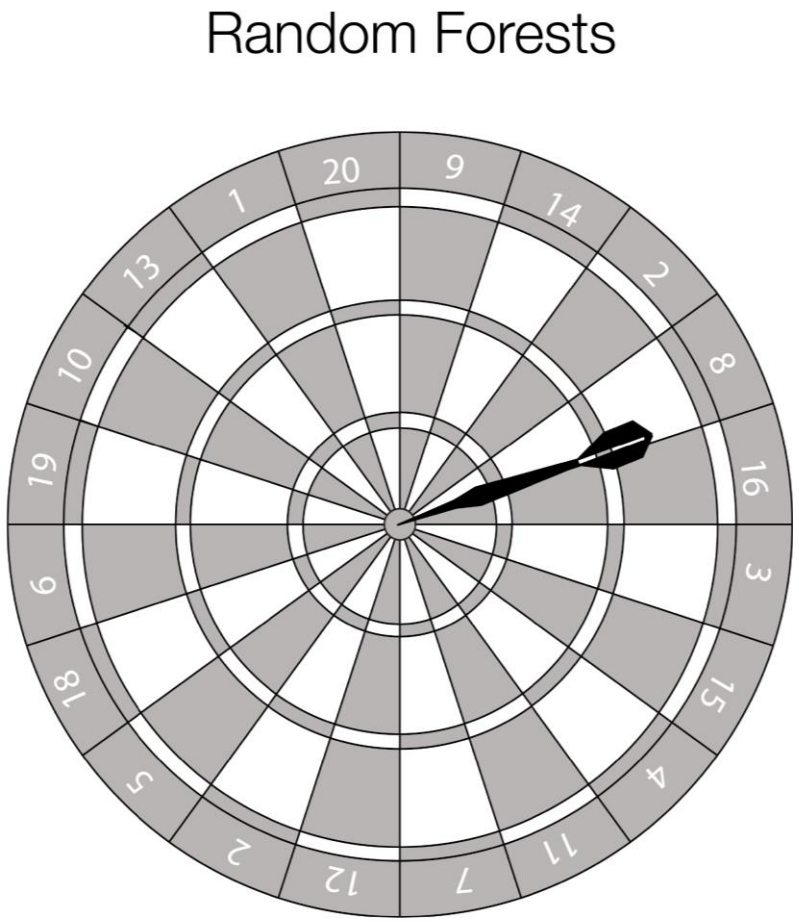
A LG model will require multiple parameters, while a DT requires multiple branches.

Deep Tree => Accuracy => High Variance



Solution : Many Shallow Trees

- A shallow tree => high bias and almost no variance
- Many shallow trees => low bias, low variance (compared with a deep tree)



Bootstrapping

N = 3

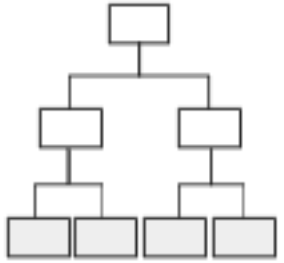
	debt	assets	price	status
0	500	2500	1250	OK
1	250	4500	1500	OK
2	500	2500	1250	OK
3	1000	4000	4500	default

Original Dataset

N = 1
sample(features, 2)

Sample 1

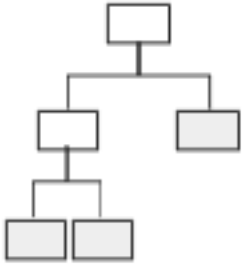
	debt	assets	status
0	500	2500	OK
1	250	4500	OK
2	500	2500	OK
3	1000	4000	default



N = 2
sample(features, 2)

Sample 2

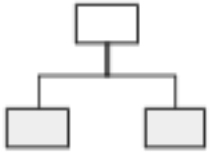
	debt	price	status
0	500	1250	OK
1	250	1500	OK
2	500	1250	OK
3	1000	4500	default



N = 3
sample(features, 2)

Sample 3

	assets	price	status
0	2500	1250	OK
1	4500	1500	OK
2	2500	1250	OK
3	4000	4500	default



Important Hyper Parameters

Number of decision trees

- Specifies the number of independent decision trees in your ensemble.
- Higher value usually result in better / more stable predictions, since errors average out.

Maximum depth of trees

- Specifies the maximum depth a tree in the ensemble can have.
- Rule of thumb: Deeper trees give better accuracy but increase the risk of overfitting.

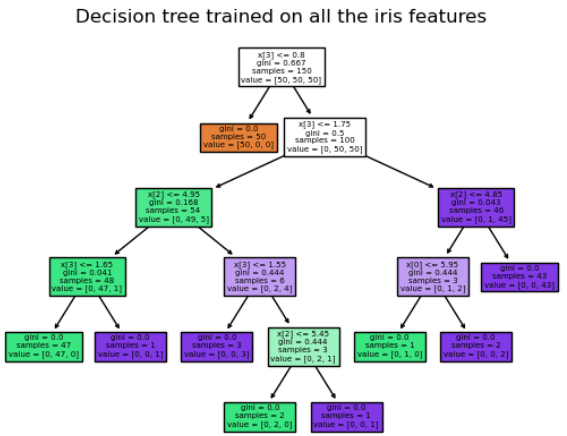
Minimum leaf size

- Determines the smallest size of a leaf node in the ensemble.
- Too many leaves can cause overfitting and poor model generalization.

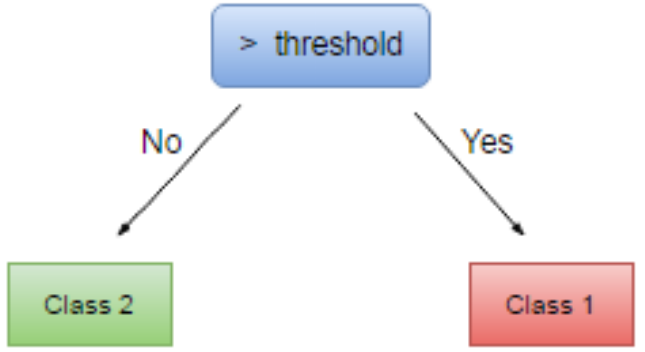
Boosting

- Boosting refers to a family of machine learning meta-algorithms which combine the outputs of many “weak” classifiers into a powerful “committee”.
- Each of the weak classifiers alone may have an error rate which is only slightly better than random guessing.

A deep decision tree



A decision stump (i.e., weak classifier)

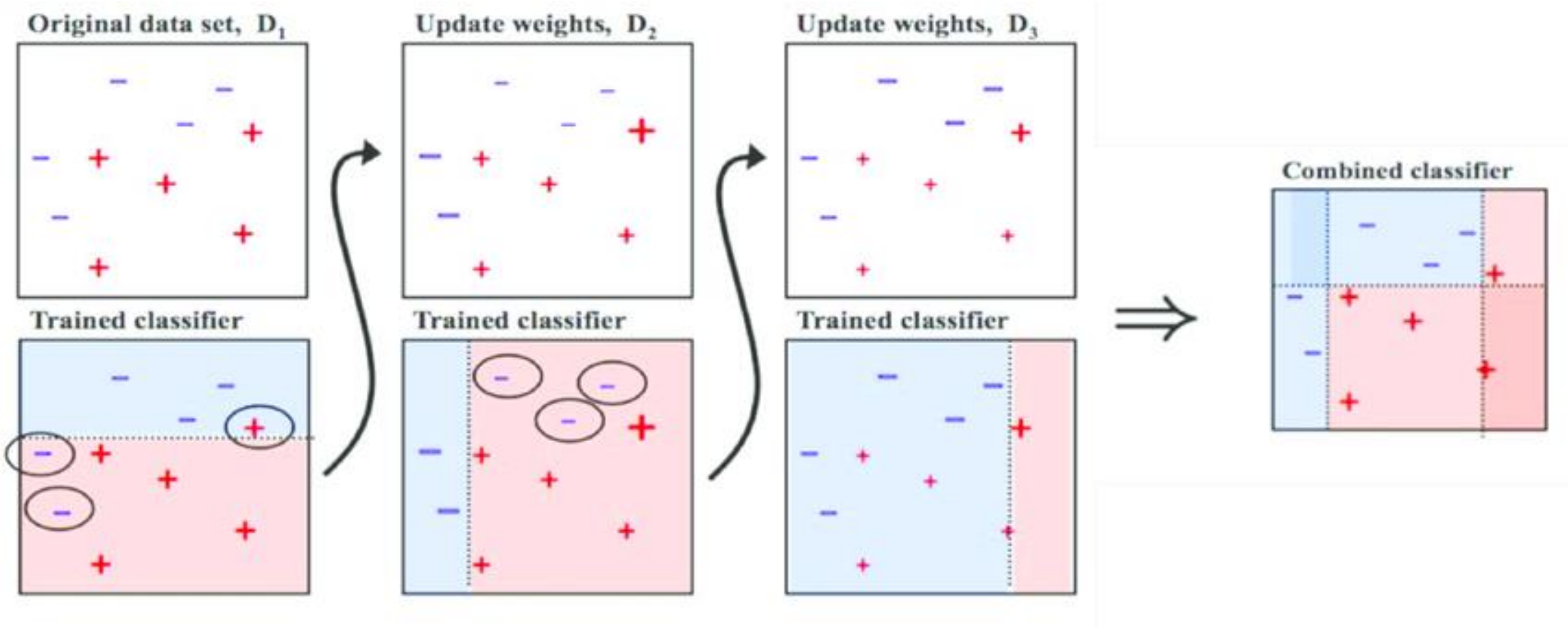


Adaptive Boosting

- AdaBoost (Adaptive Boosting) is a particular boosting algorithm in which we fit a sequence of “stumps” (decision trees with a single node and two leaves) and weight their contribution to the final vote by how accurate their predictions are.

- After each iteration, we re-weight the dataset to assign greater importance to data points which were misclassified by the previous weak learner, so that those data points get “special attention” during iteration

Adaptive Boosting



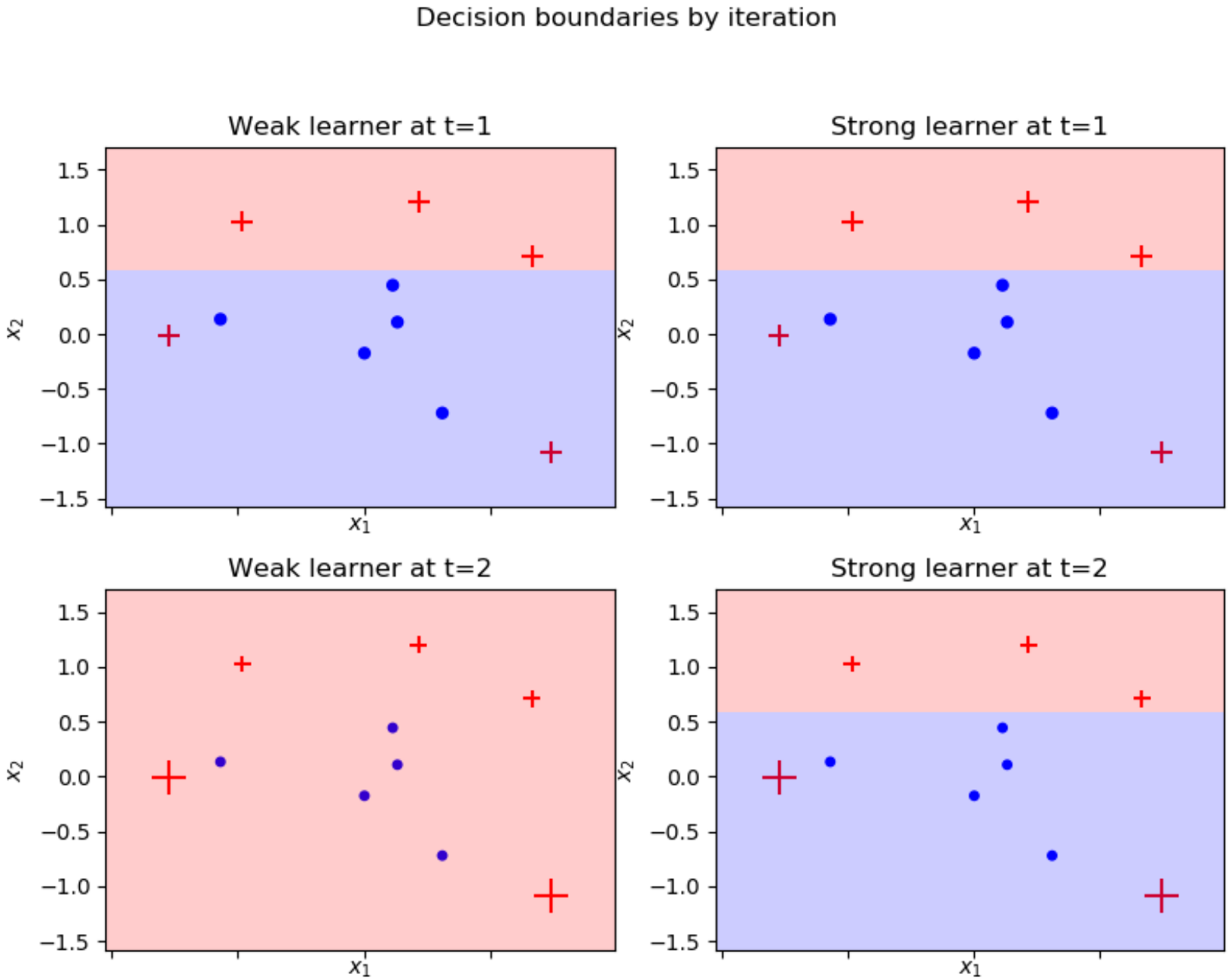
Adaptive Boosting

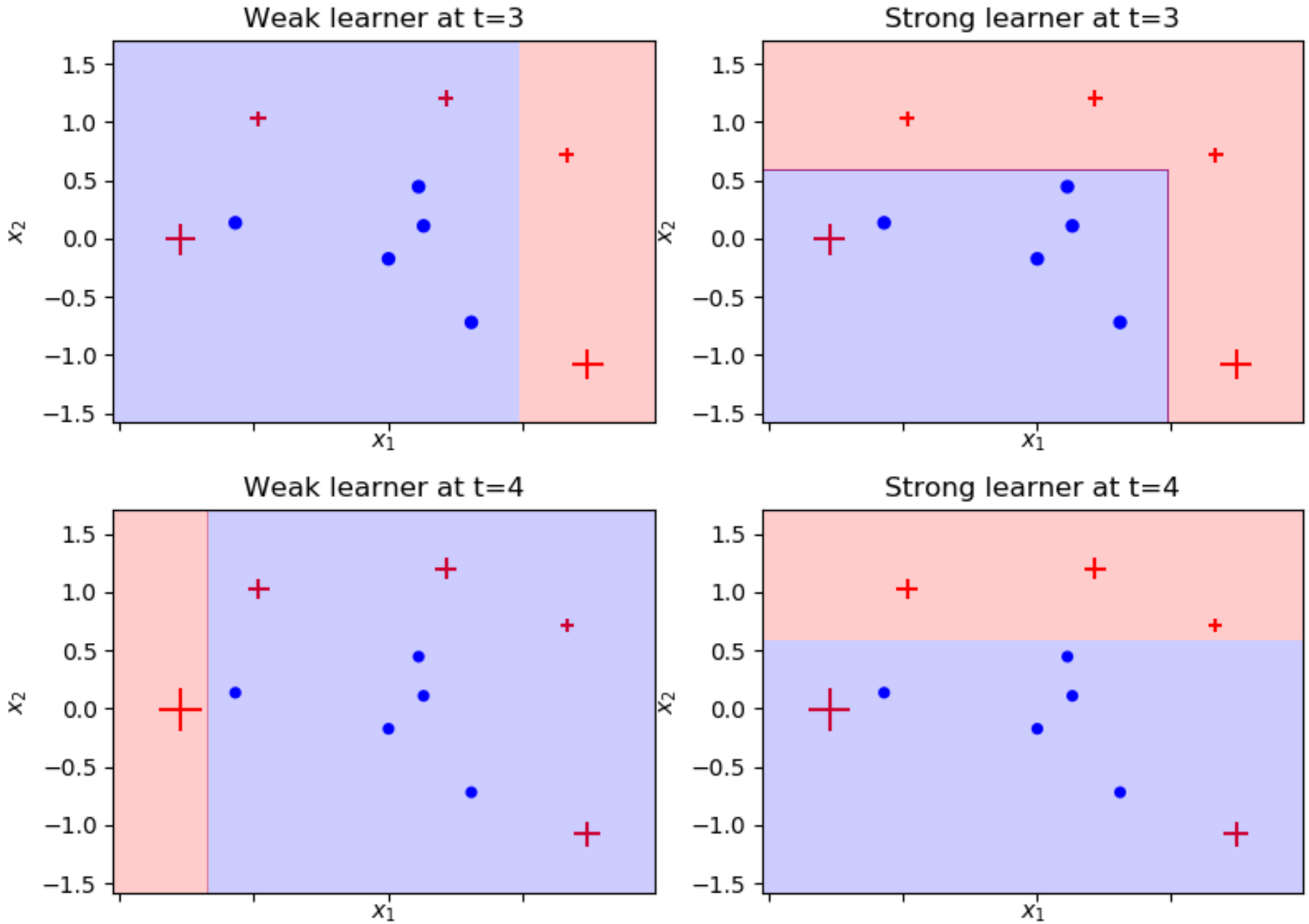
A) Initialize sample weights uniformly as $w_i^1 = \frac{1}{n}$.

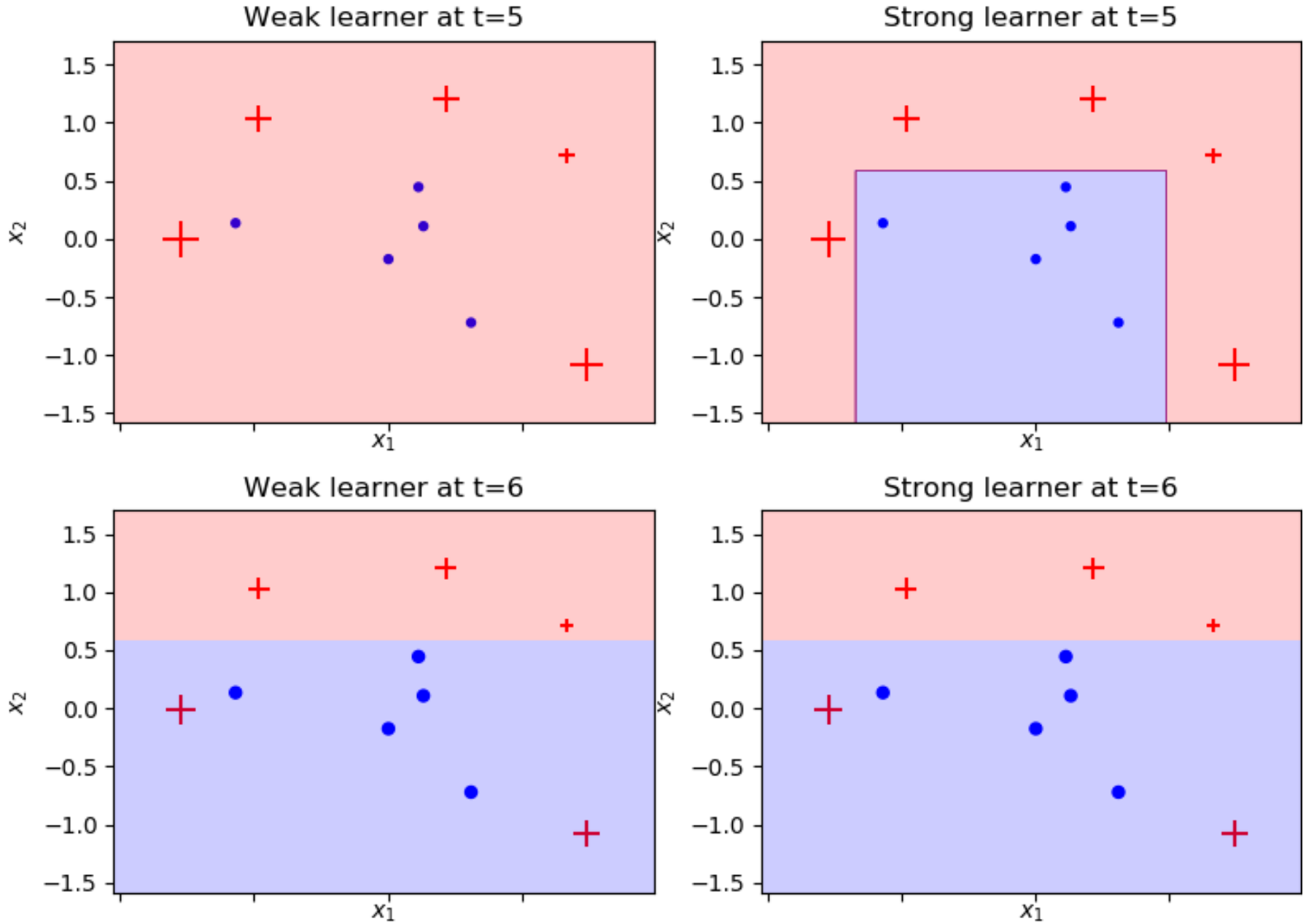
B) For each iteration t:

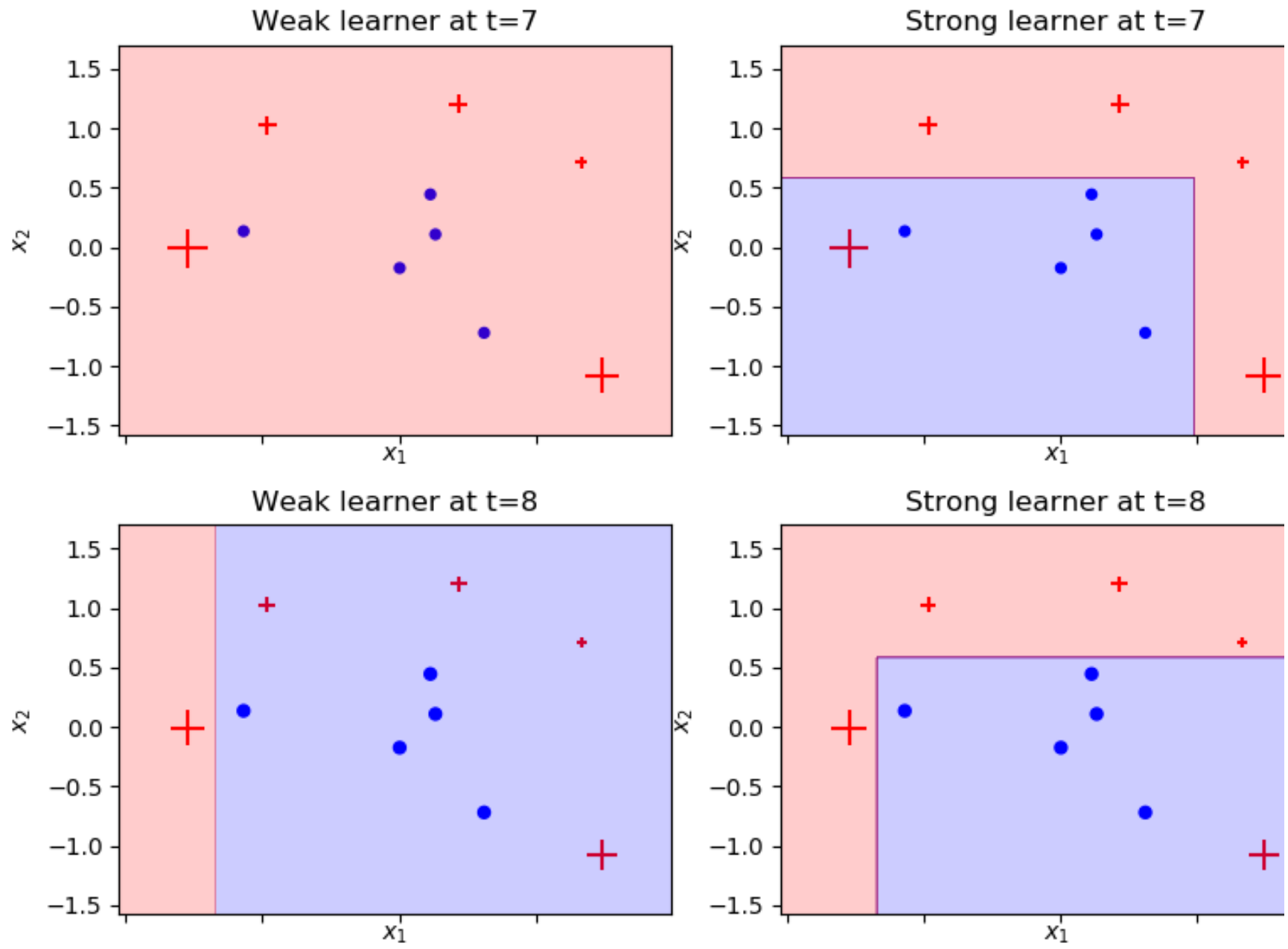
1. Find weak learner $h_t(x)$ which minimizes $\epsilon_t = \sum_{i=1}^{n} \mathbf{1}[h_t(x_i) \neq y_i]\, w_i^{(t)}$.
2. We set a weight for our weak learner based on its accuracy: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
3. Increase weights of misclassified observations: $w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\alpha^t y_i h_t(x_i)}$.
4. Renormalize weights, so that $\sum_{i=1}^{n} w_i^{(t+1)} = 1$.

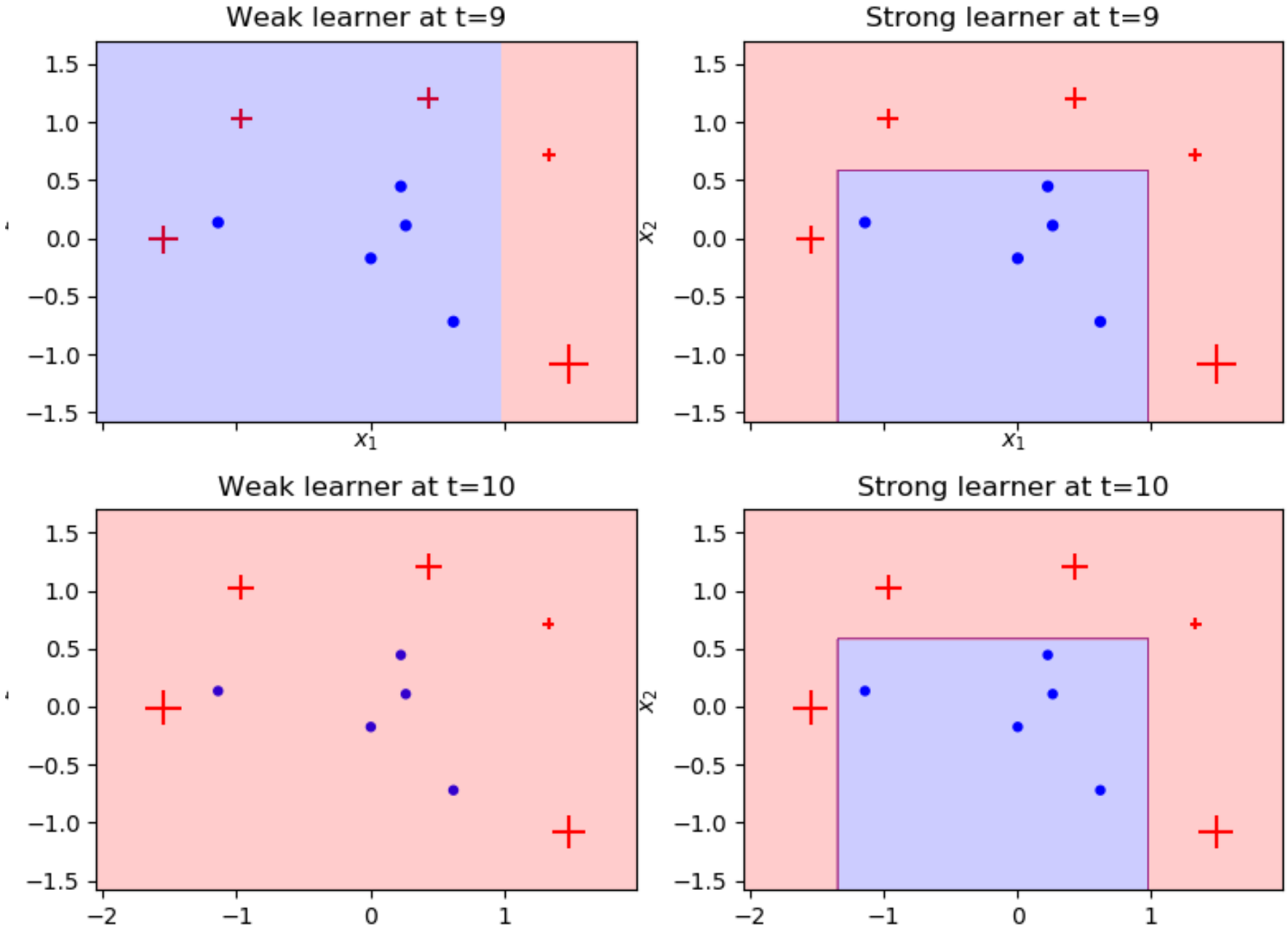
C) Make final prediction as weighted majority vote of weak learner predictions:
$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.


Decision boundaries by iteration
Weak learner at t=1
Strong learner at t=1
Weak learner at t=2
Strong learner at t=2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=3
Strong learner at t=3
Weak learner at t=4
Strong learner at t=4
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=5
Strong learner at t=5
Weak learner at t=6
Strong learner at t=6
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=7
Strong learner at t=7
Weak learner at t=8
Strong learner at t=8
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=9
Strong learner at t=9
Weak learner at t=10
Strong learner at t=10
x_1
x_2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1

How does it compare to Random Forest?

Property	Random Forest	AdaBoost
Depth	Unlimited (a full tree)	Stump (single node w/ 2 leaves)
Trees grown	Independently	Sequentially
Votes	Equal	Weighted

Clustering Analysis

What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation,* …)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market resarch

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

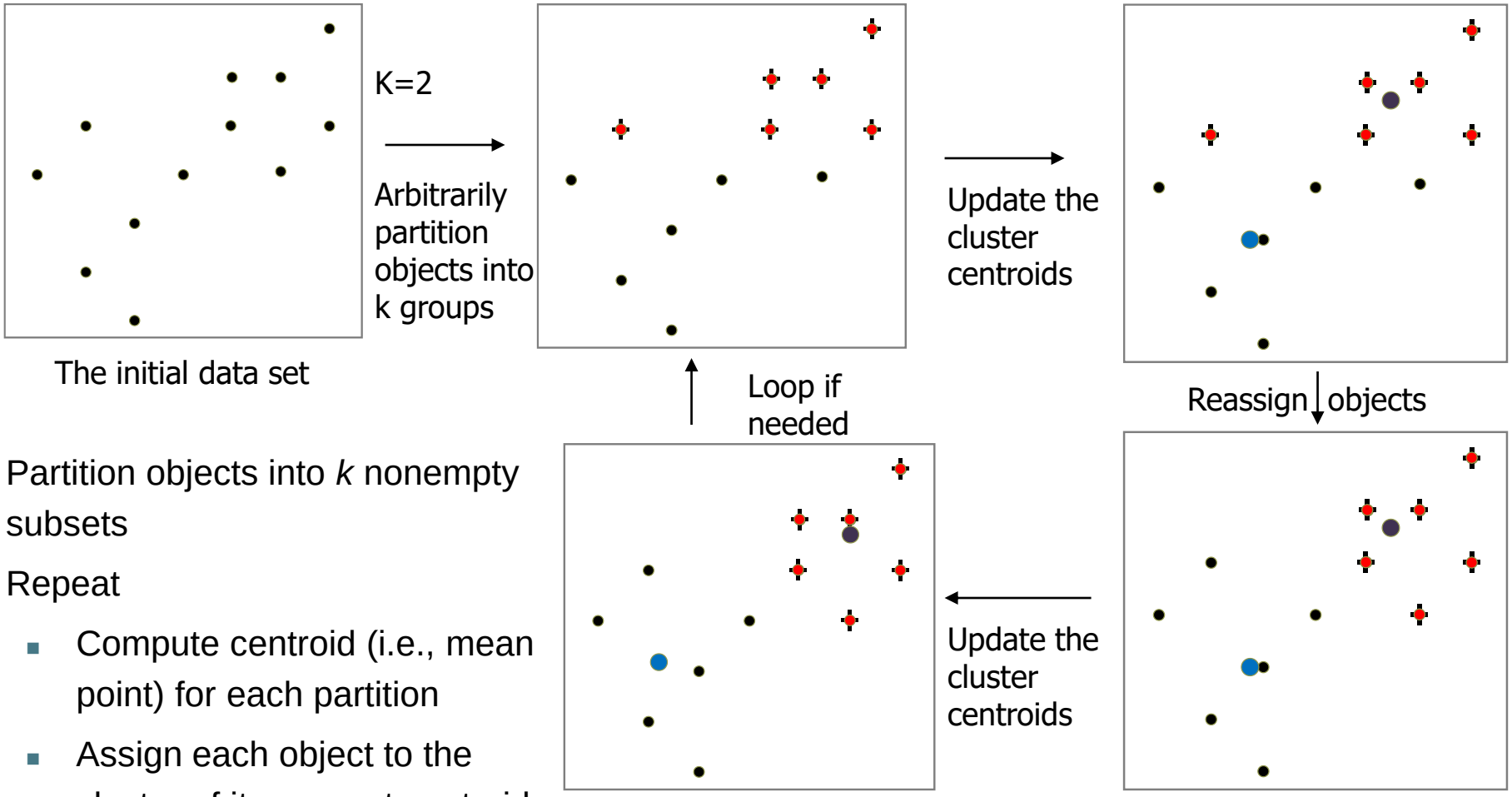
Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Agglomerative
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

An Example of *K-Means* Clustering



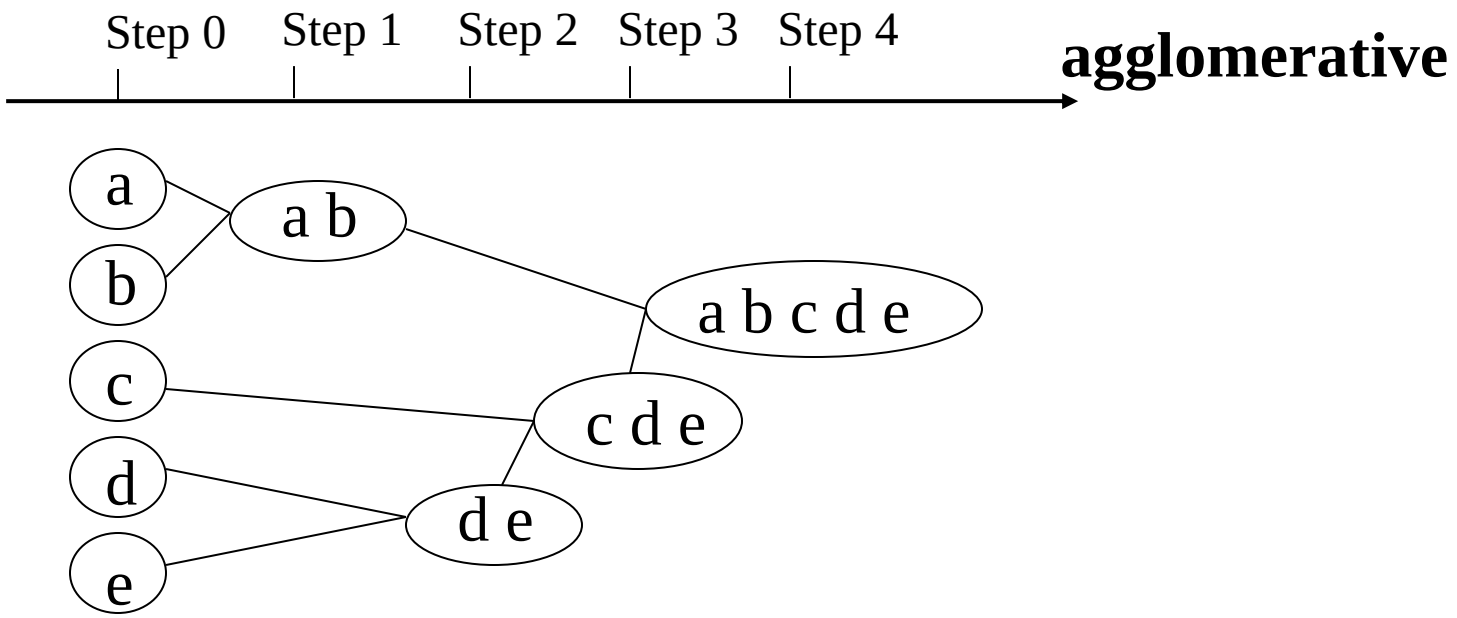
- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data

Hierarchical Clustering

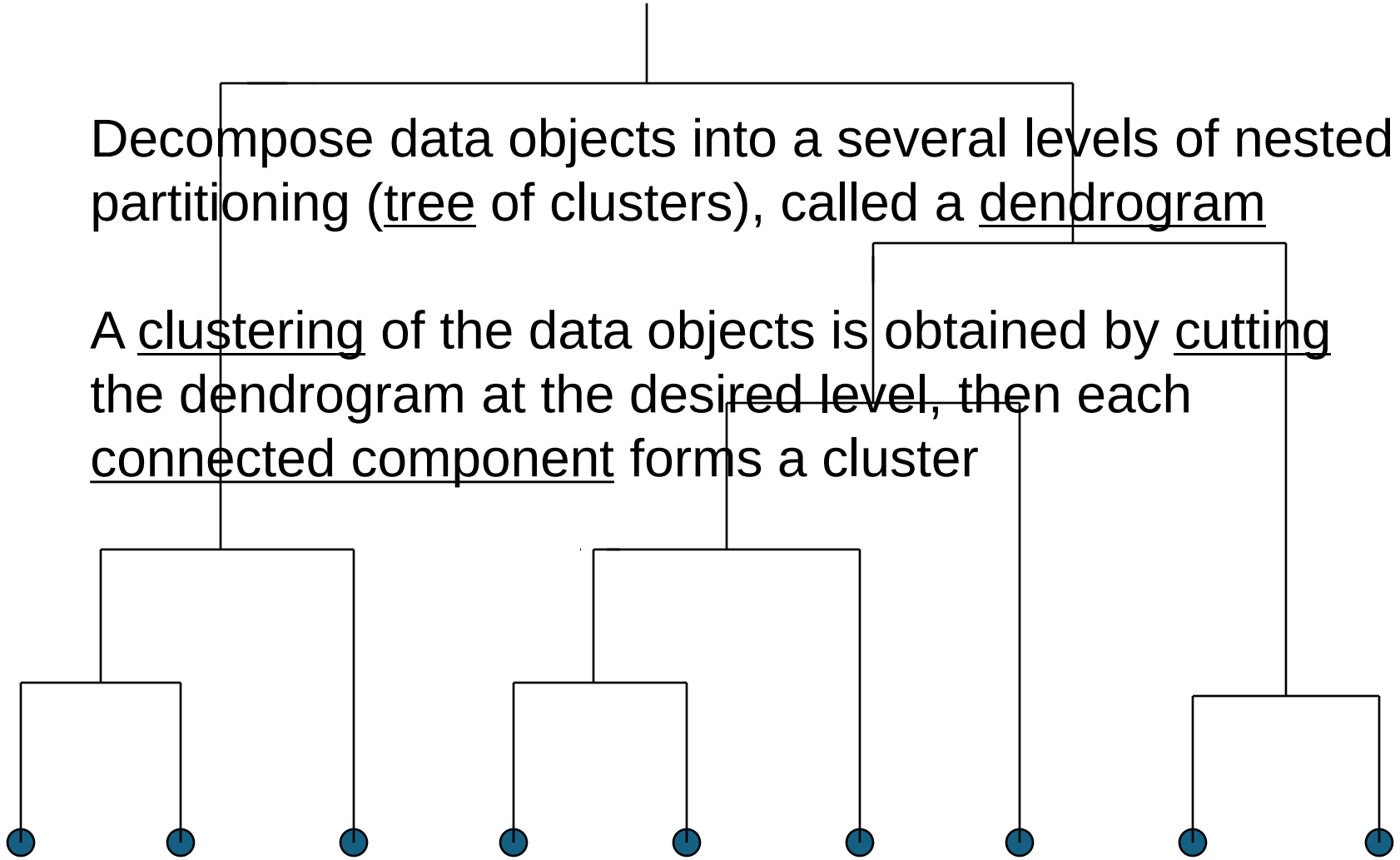
- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



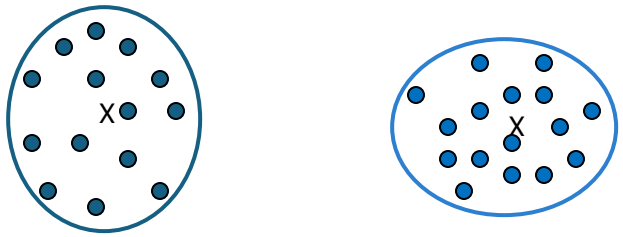
Dendrogram: Shows How Clusters are Merged

Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



Distance between Clusters



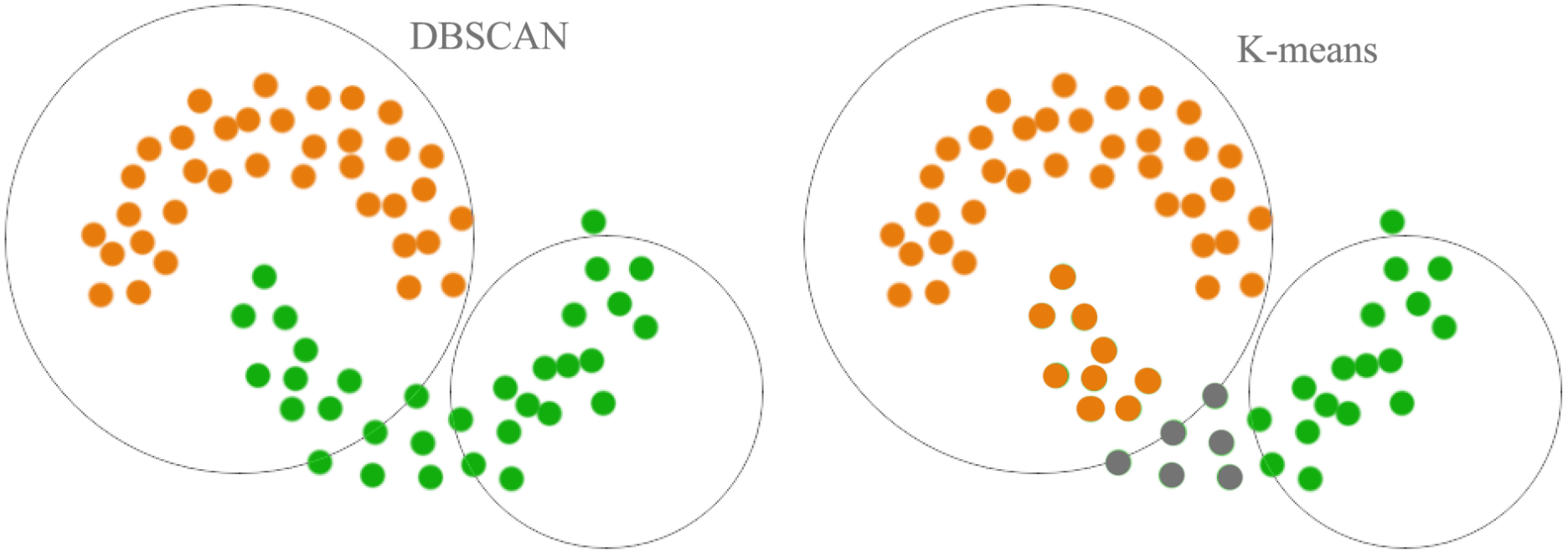
- Single link: smallest distance between an element in one cluster and an element in the other, i.e., dist(K_i, K_j) = min(t_{ip}, t_{jq})
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., dist(K_i, K_j) = max(t_{ip}, t_{jq})
- Average: avg distance between an element in one cluster and an element in the other, i.e., dist(K_i, K_j) = avg(t_{ip}, t_{jq})
- Centroid: distance between the centroids of two clusters, i.e., dist(K_i, K_j) = dist(C_i, C_j)
- Medoid: distance between the medoids of two clusters, i.e., dist(K_i, K_j) = dist(M_i, M_j)
  - Medoid: a chosen, centrally located object in the cluster

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition

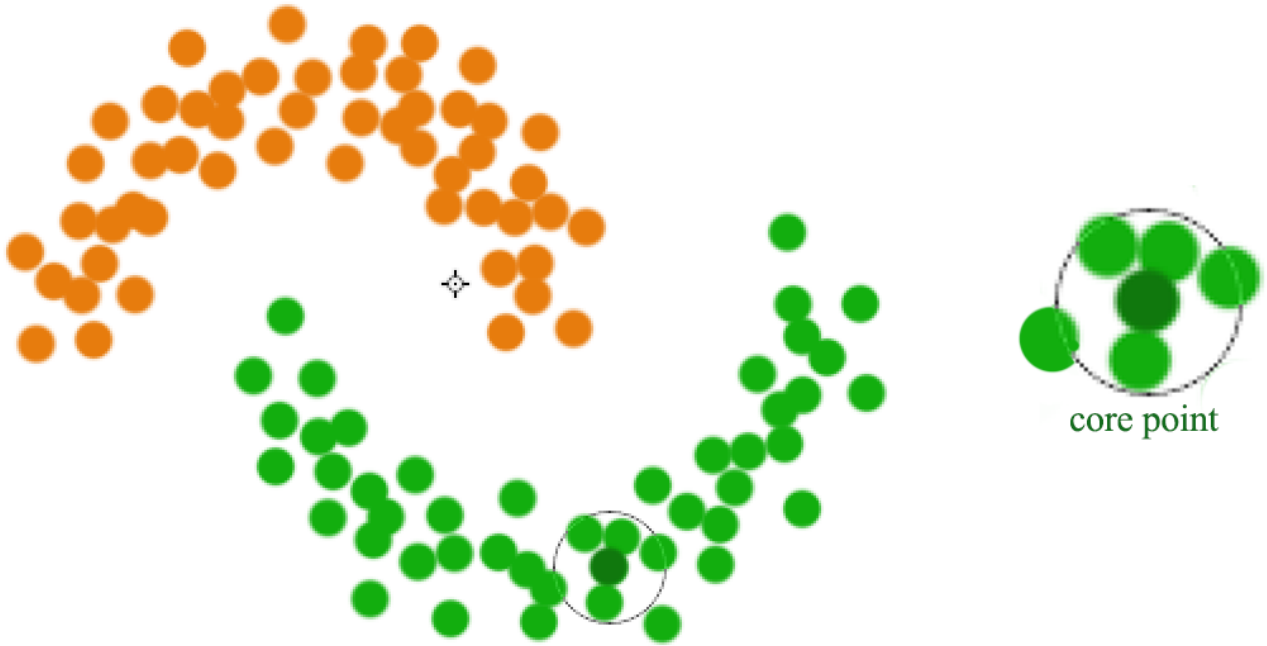
Density based clustering (DBSCAN)

- As shown below, a distance based cluster like K-means will have problem to cluster concave shape cluster:



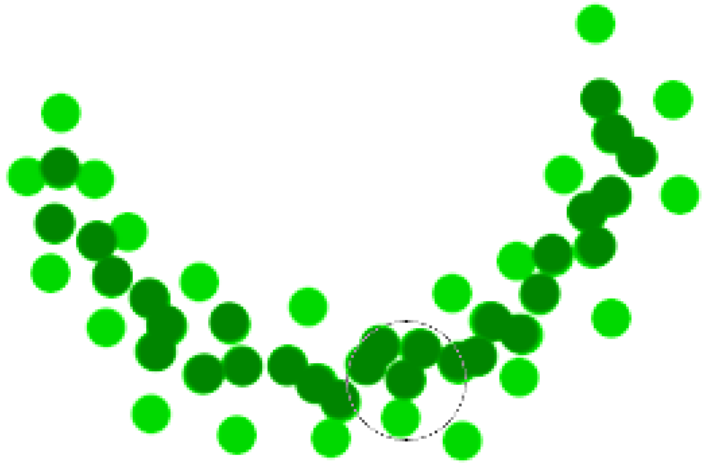
Density based clustering (DBSCAN)

- Density based clustering connects neighboring high density points together to form a cluster.



Density based clustering (DBSCAN)

- A datapoint is a core point if within radius r, there are m reachable points. A cluster is form by connecting core points (the darker green) that are reachable from the others.



The green cluster is formed by

1. Locating all the core points (dark green)
2. Joining all the core points that are within r
3. Joining all points that are within r from all those core points (shown as light green)

Density based clustering (DBSCAN)

•The green cluster contains both the dark and light green dots.

Unlike other clustering, a datapoint may not belong to any cluster.

Association Rule Mining

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$$\{Diaper\} \rightarrow \{Beer\},$$
$$\{Milk, Bread\} \rightarrow \{Eggs, Coke\},$$
$$\{Beer, Bread\} \rightarrow \{Milk\},$$

Implication means co-occurrence, not causality!

Definitions

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread, Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper})
    = σ({Milk, Bread, Diaper}) / |T| = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$s(X) = \frac{\sigma(X)}{|T|}$$

Definitions

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example: $\{Milk, Bread\} \rightarrow \{Diaper\}$

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

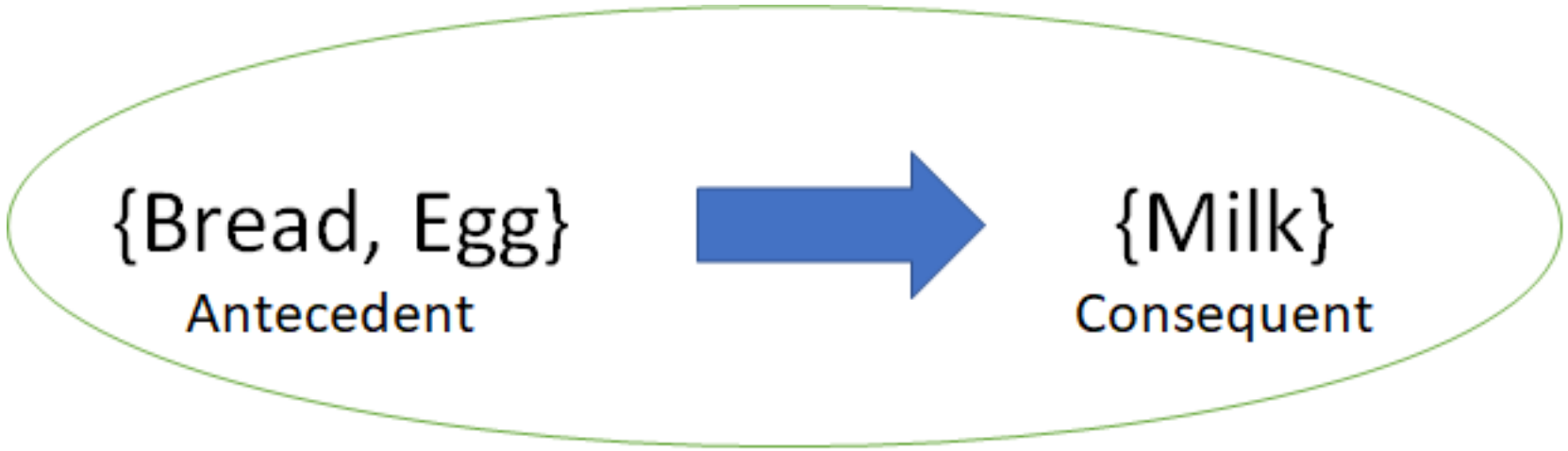
$\{Milk, Bread\} \rightarrow \{Diaper\}$

$$s = \frac{\sigma(\{Milk, Bread, Diaper\})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\{Milk, Bread, Diaper\})}{\sigma(\{Milk, Diaper\})} = \frac{2}{3} = 0.67$$

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{s(X \cup Y)}{s(X)}$$

Definitions



Itemset = {Bread, Egg, Milk}

Association Rule Mining Tasks

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

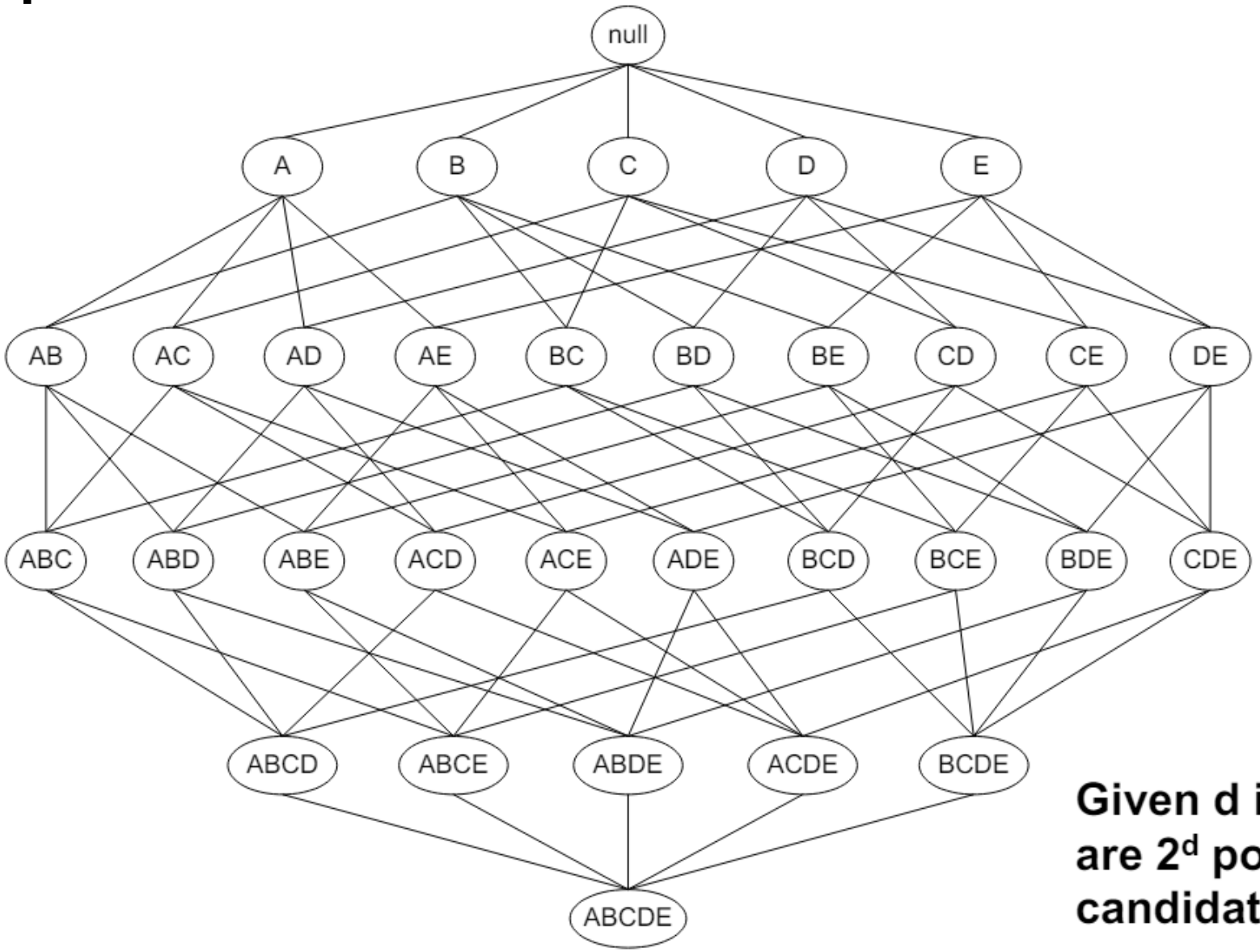
  ⇒ Computationally prohibitive!

Association Rule Mining Tasks

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Illustrating Apriori Principle

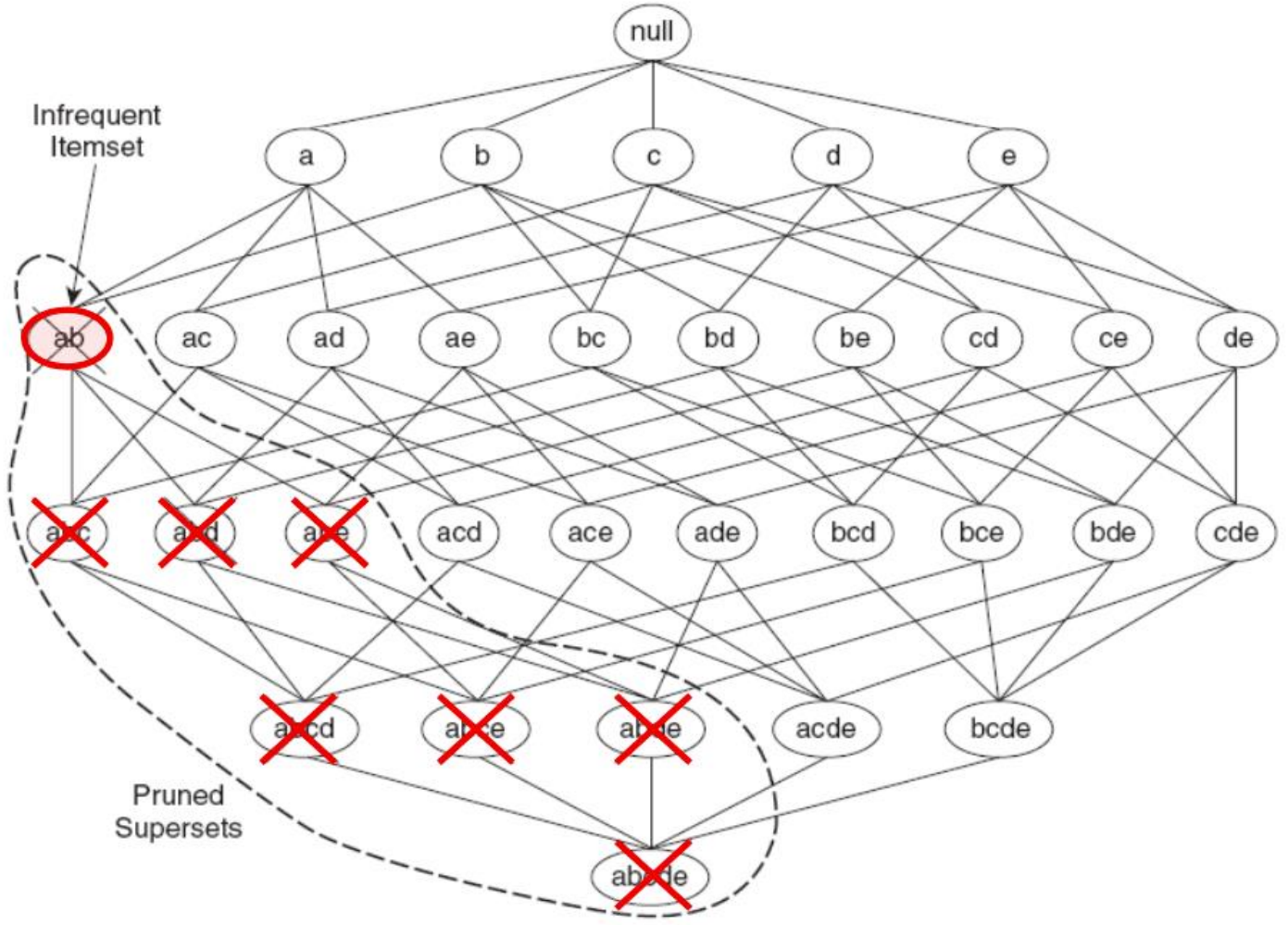
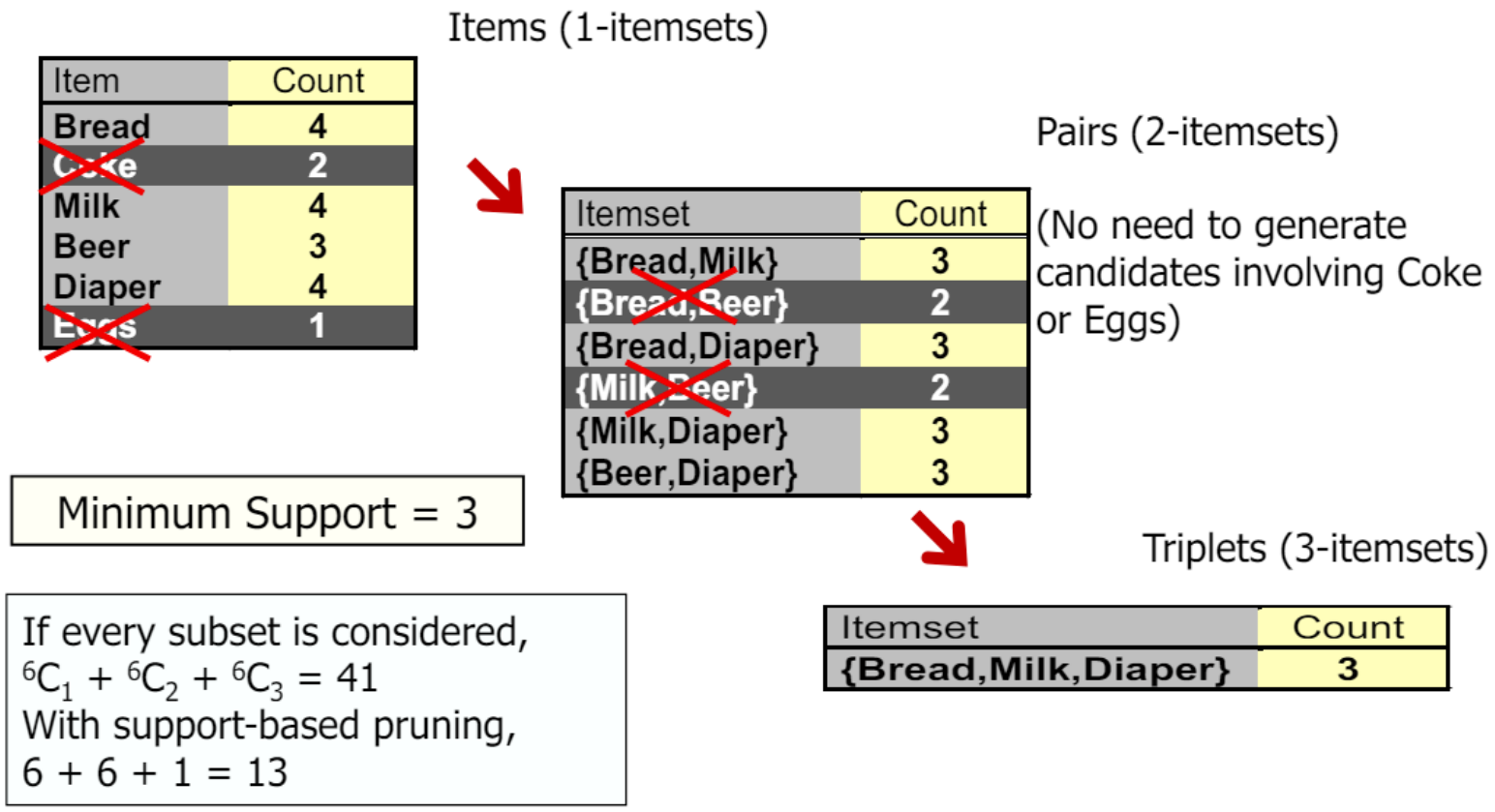


Figure 6.4. An illustration of support-based pruning. If $\{a, b\}$ is infrequent, then all supersets of $\{a, b\}$ are infrequent.

Illustrating Apriori Principle



Apriori Aglorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

$ABC \rightarrow D$,	$ABD \rightarrow C$,	$ACD \rightarrow B$,	$BCD \rightarrow A$,
$A \rightarrow BCD$,	$B \rightarrow ACD$,	$C \rightarrow ABD$,	$D \rightarrow ABC$
$AB \rightarrow CD$,	$AC \rightarrow BD$,	$AD \rightarrow BC$,	$BC \rightarrow AD$,
$BD \rightarrow AC$,	$CD \rightarrow AB$,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule Generation

- In general, confidence does not have an anti-monotone property

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm