



# Advanced Data Science

Algorithms

Session 7

**Kiran Waghmare**

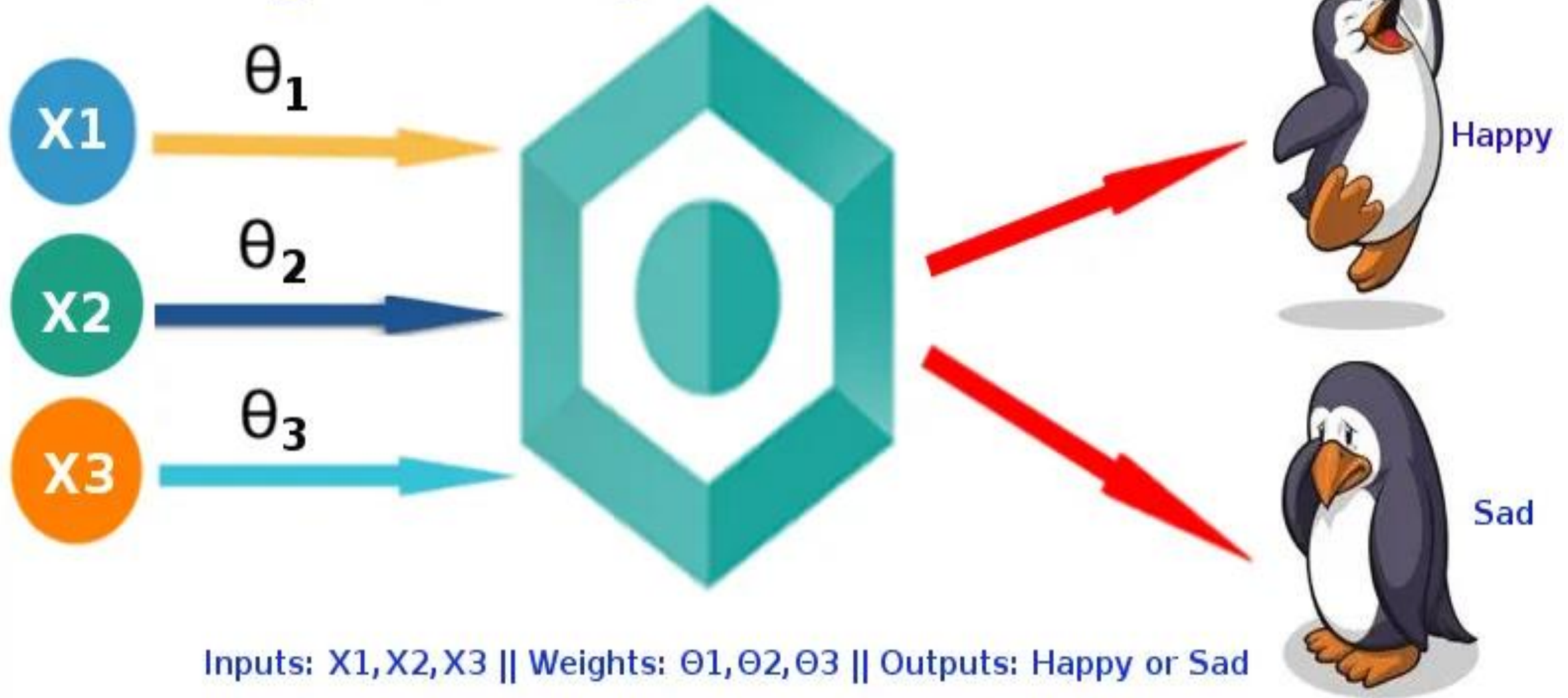
Program Manager

C-DAC Mumbai

# Agenda

- Algorithm
  - Logistic Regression
  - Decision Tree Classifier
  - Random Forest Classifier
  - Bagging and Boosting
  - Hyperparameters and Model Validation;
- Dimensionality Reduction;
  - Principal Component Analysis
- Singular Value Decomposition;
- Manifold Learning and Diffusion Maps;
- Spectral Clustering;
- Approximation Algorithms;
- Synchronization.

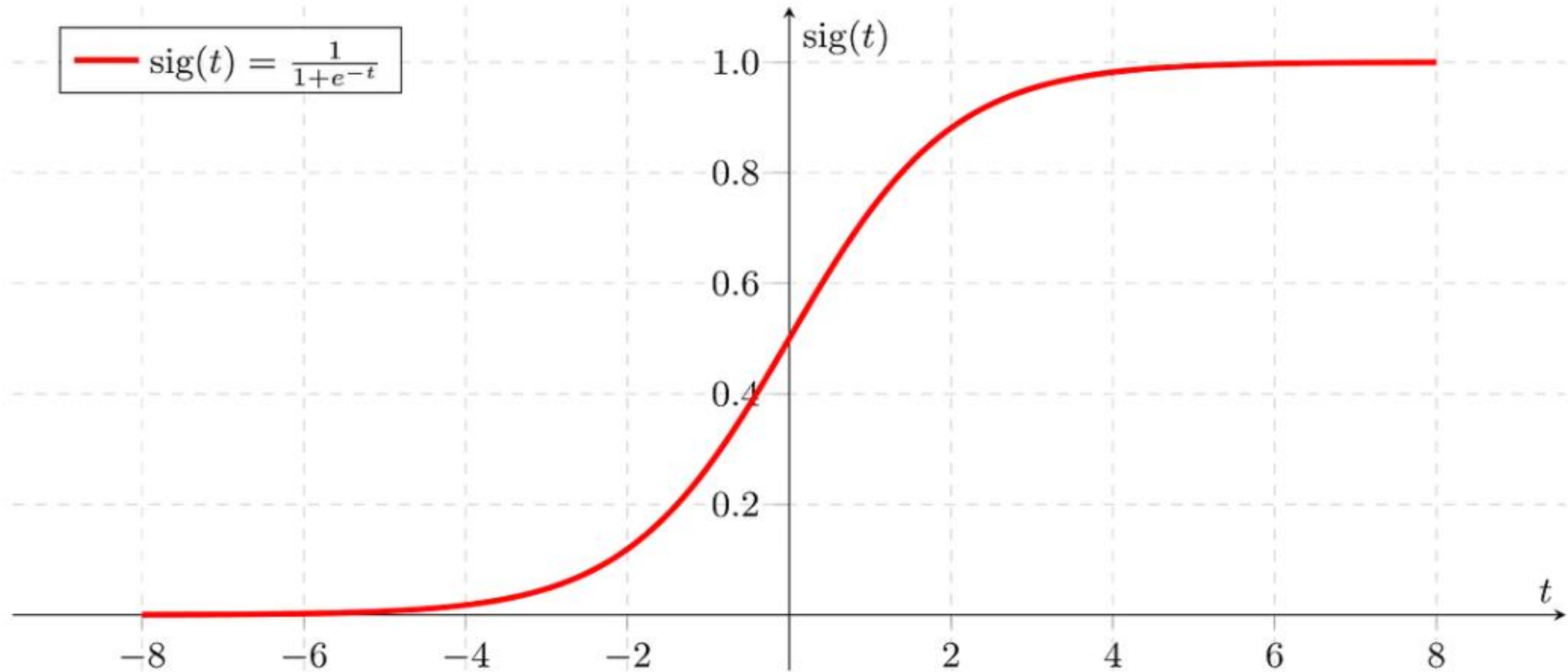
# Logistic Regression Model



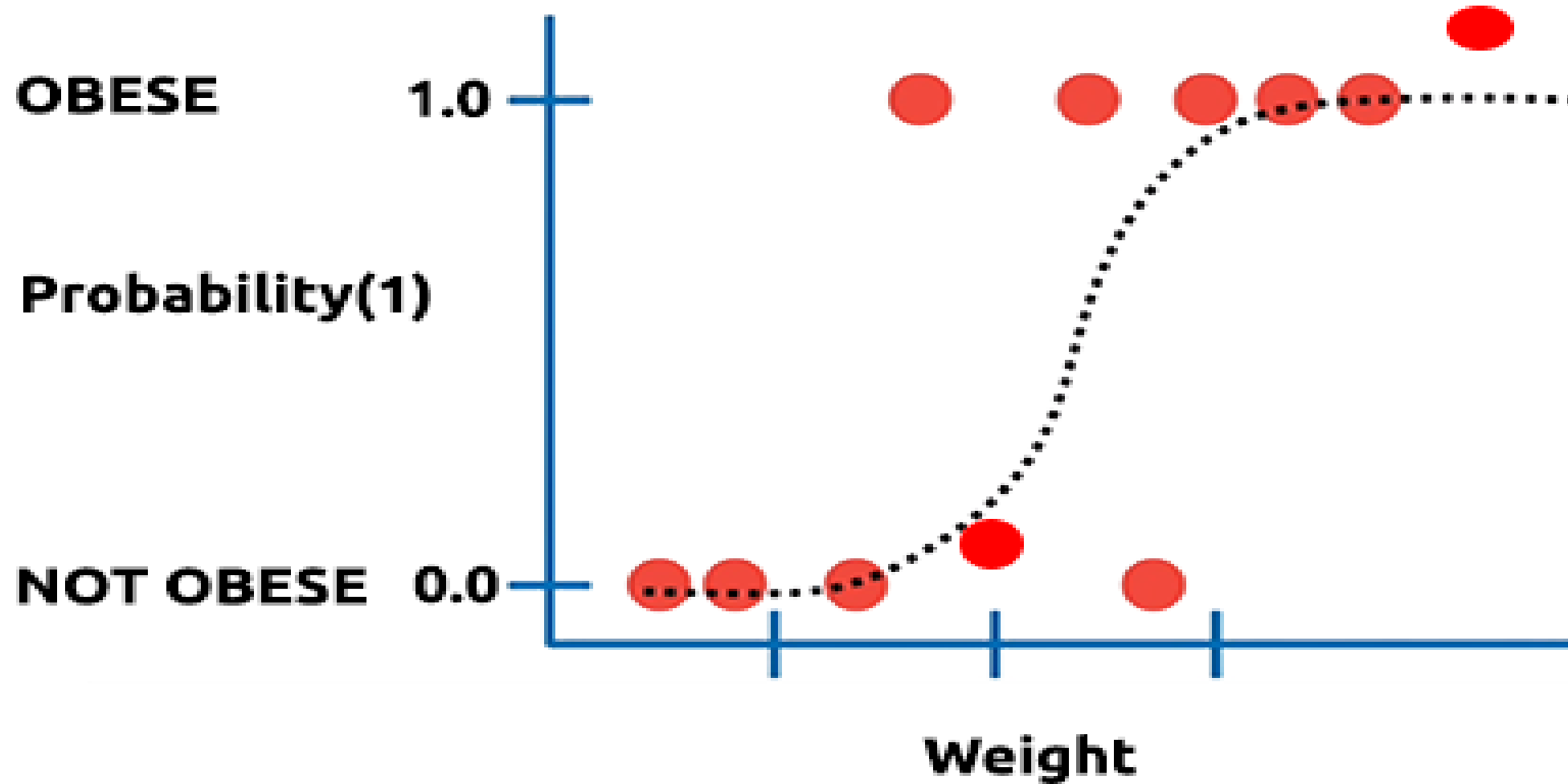
# Introduction to Logistic Regression

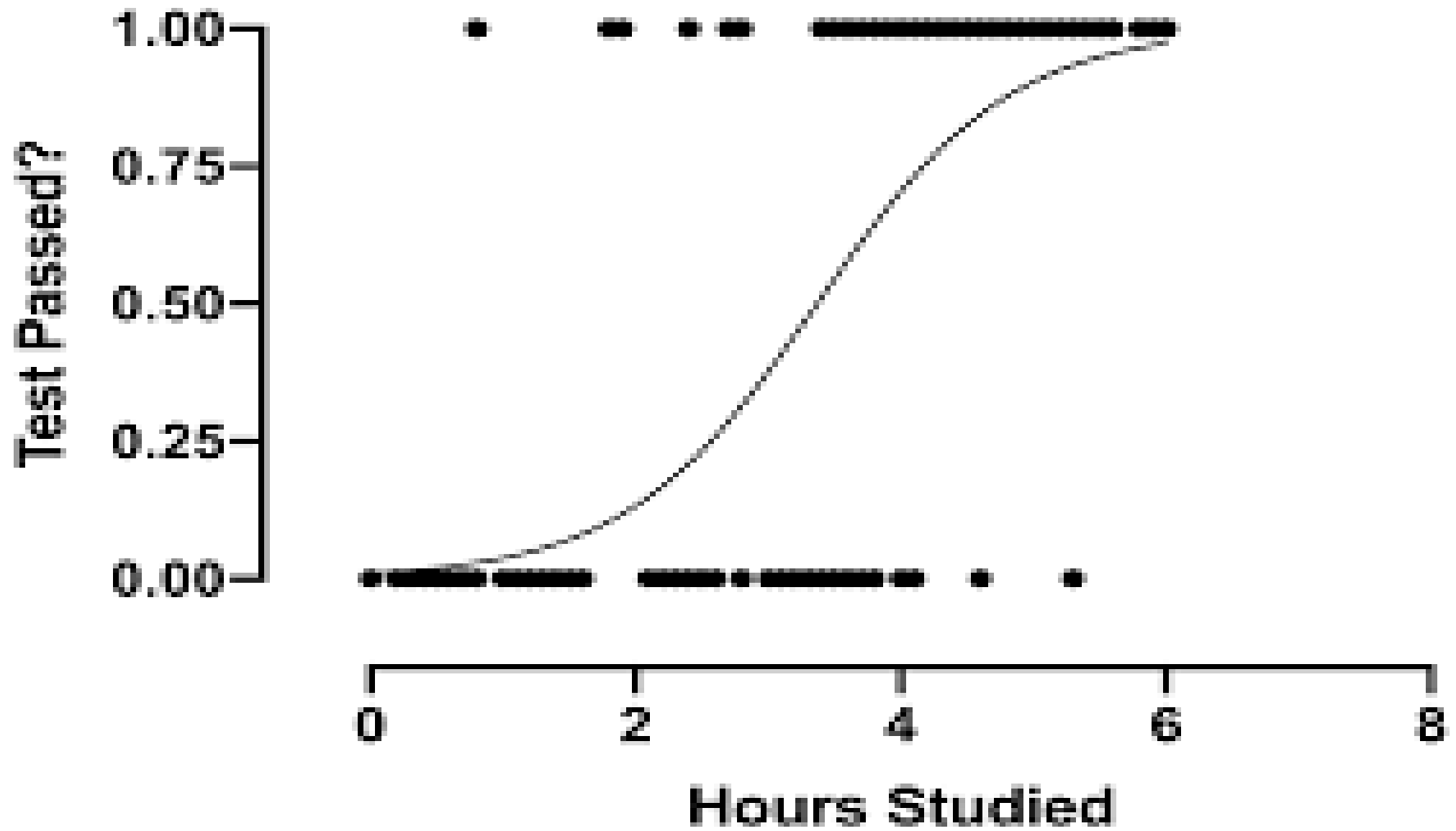
- **Type of Model:** A classification algorithm used to predict the probability of a binary outcome (i.e., two classes).
- **Goal:** Estimate the probability that an instance belongs to a certain class (e.g., 0 or 1).
- **Common Use Cases:**
  - Predicting whether a customer will churn (Yes/No).
  - Predicting whether a tumor is malignant or benign.
  - Fraud detection (Fraud/Not Fraud).
- **Mathematical Foundation**
  - **Binary Classification:** Logistic regression is used when the dependent variable (target) is binary.
  - **Logistic Function (Sigmoid Function):**
    - Converts the linear output (from a linear equation) into a probability score between 0 and 1.

## Sigmoid Function



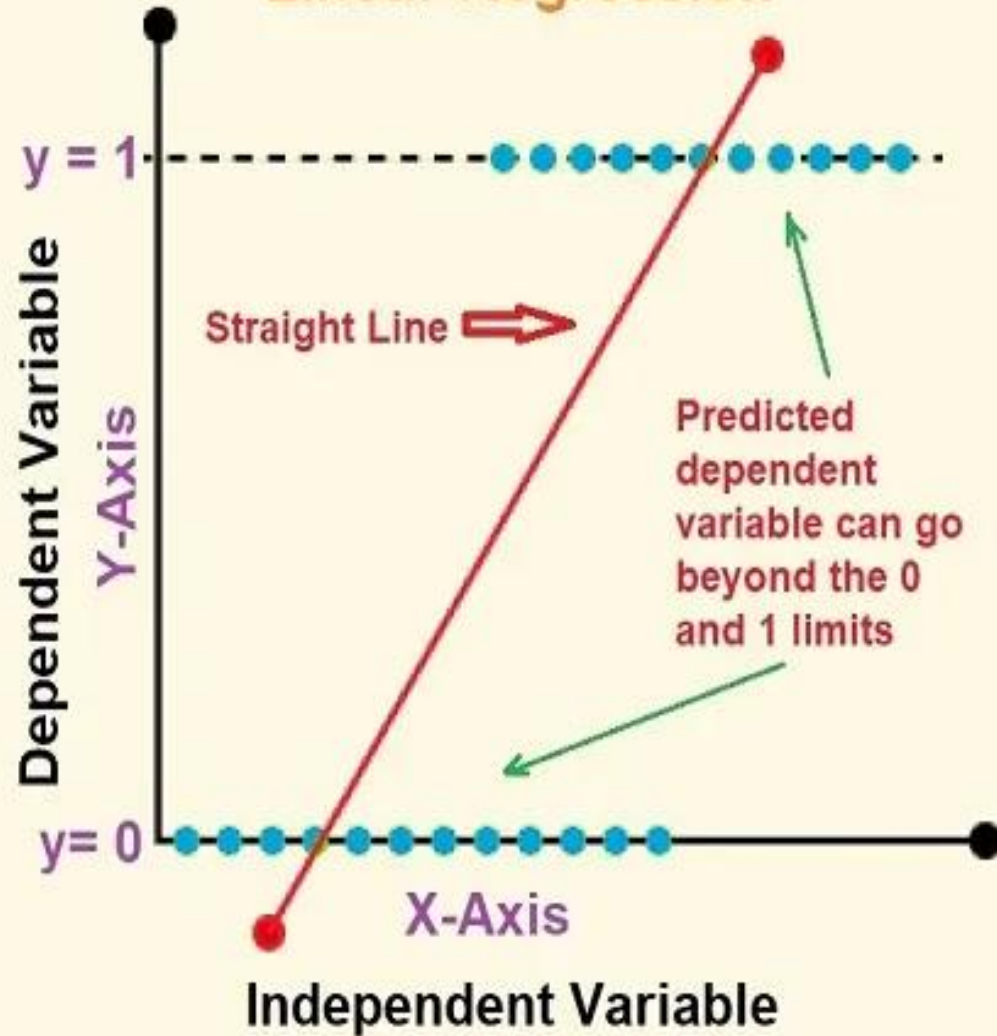
Sigmoid Activation Function



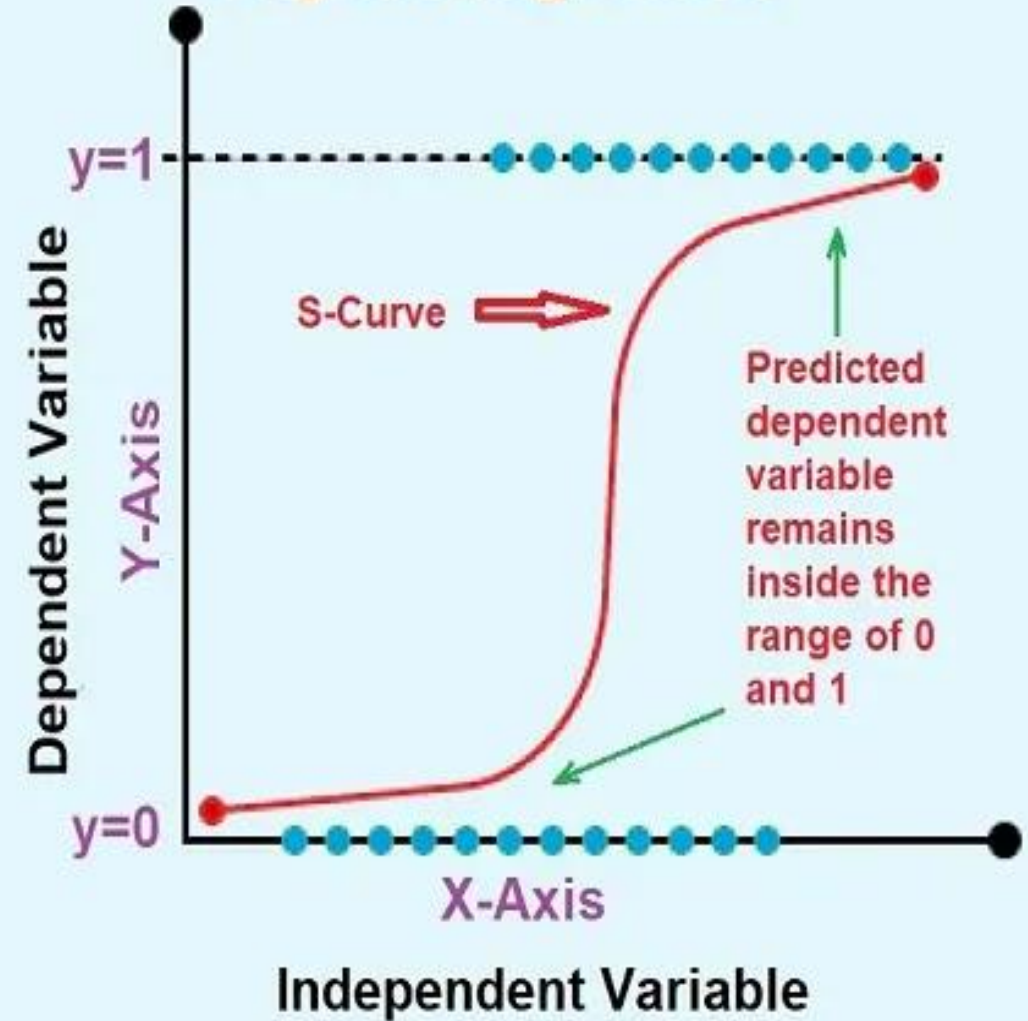




## Linear Regression



## Logistic Regression

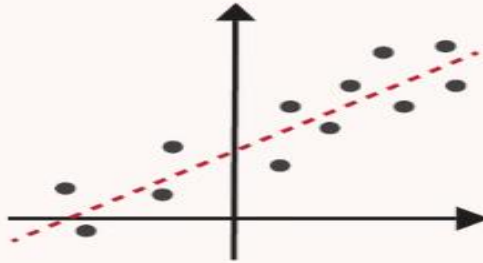




# Linear regression VS Logistic regression

## Linear regression

- Econometric modeling
- Marketing mix model
- Customer lifetime value



Continuous > Continuous

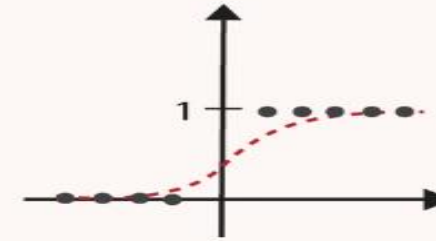
$$y = a_0 + \sum_{i=1}^N a_i x_i$$

`lm(y~x1 + x2, data)`

1 unit increase in  
x increases y by  $a$

## Logistic regression

- Customer choice model
- Click-through rate
- Conversion rate
- Credit scoring



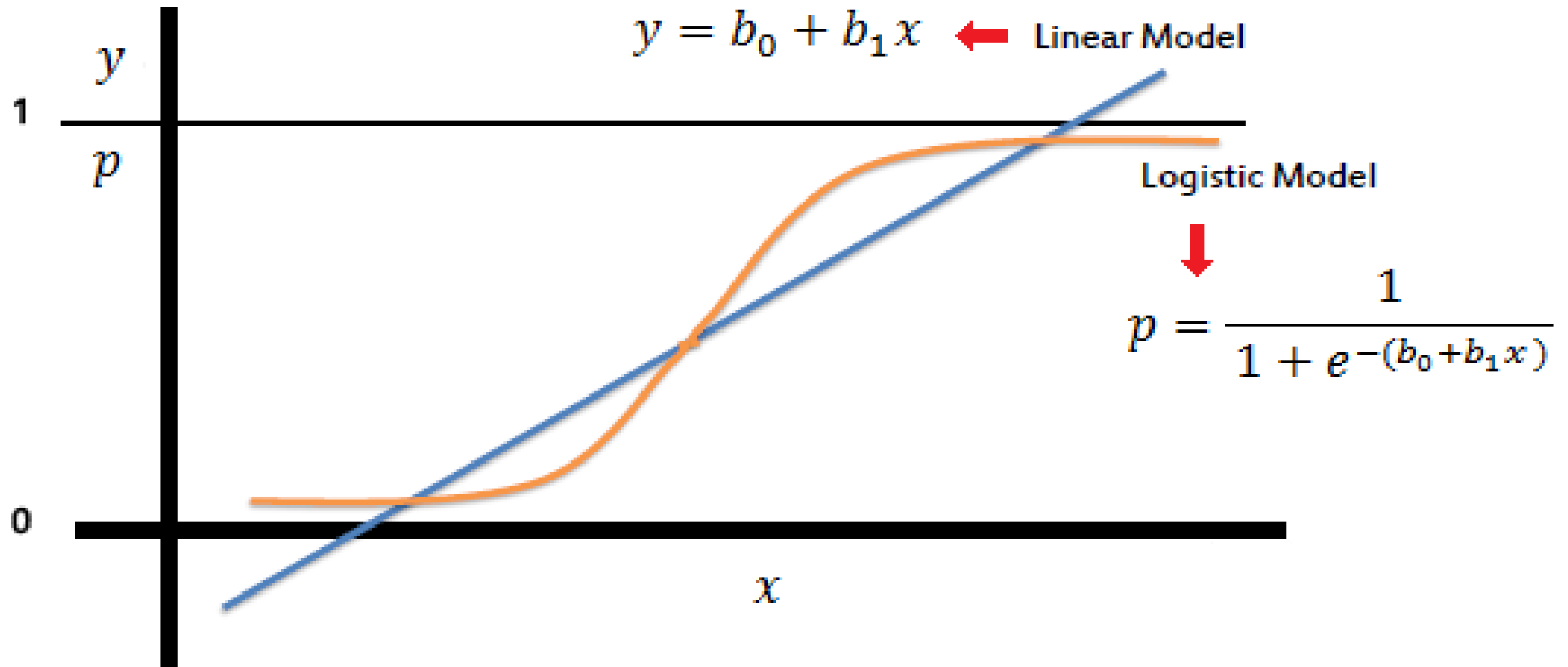
Continuous > True/False

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

$$z = a_0 + \sum_{i=1}^N a_i x_i$$

`glm(y~x1 + x2, data), family = binomial()`

1 unit increase in x  
increases log odds by  $a$



```
> data <- read.csv("credit_scoring.csv")
> fit <- glm (DEFAULT~., family=binomial(logit), data=data )
> summary(fit)
```



	Coefficient	SE	Z	P(z)
(Intercept)	-5.706	1.157	-4.933	0.000
BUSAGE	0.008	0.004	2.219	0.027
DAYSDELQ	0.102	0.020	5.106	0.000

$$y' = b_0 + b_1x_1 + b_2x_2$$

if  $p < 0.5$  then N else Y



BUSAGE	DAYSDELQ	DEFAULT	$y'$	$p$	Prediction
87	2	N	-4.811	0.008	N
89	2	N	-4.795	0.008	N
100	26	Y	-2.261	0.094	N
275	54	Y	1.983	0.879	Y
42	57	Y	0.437	0.608	Y
88	53	N	0.395	0.597	Y

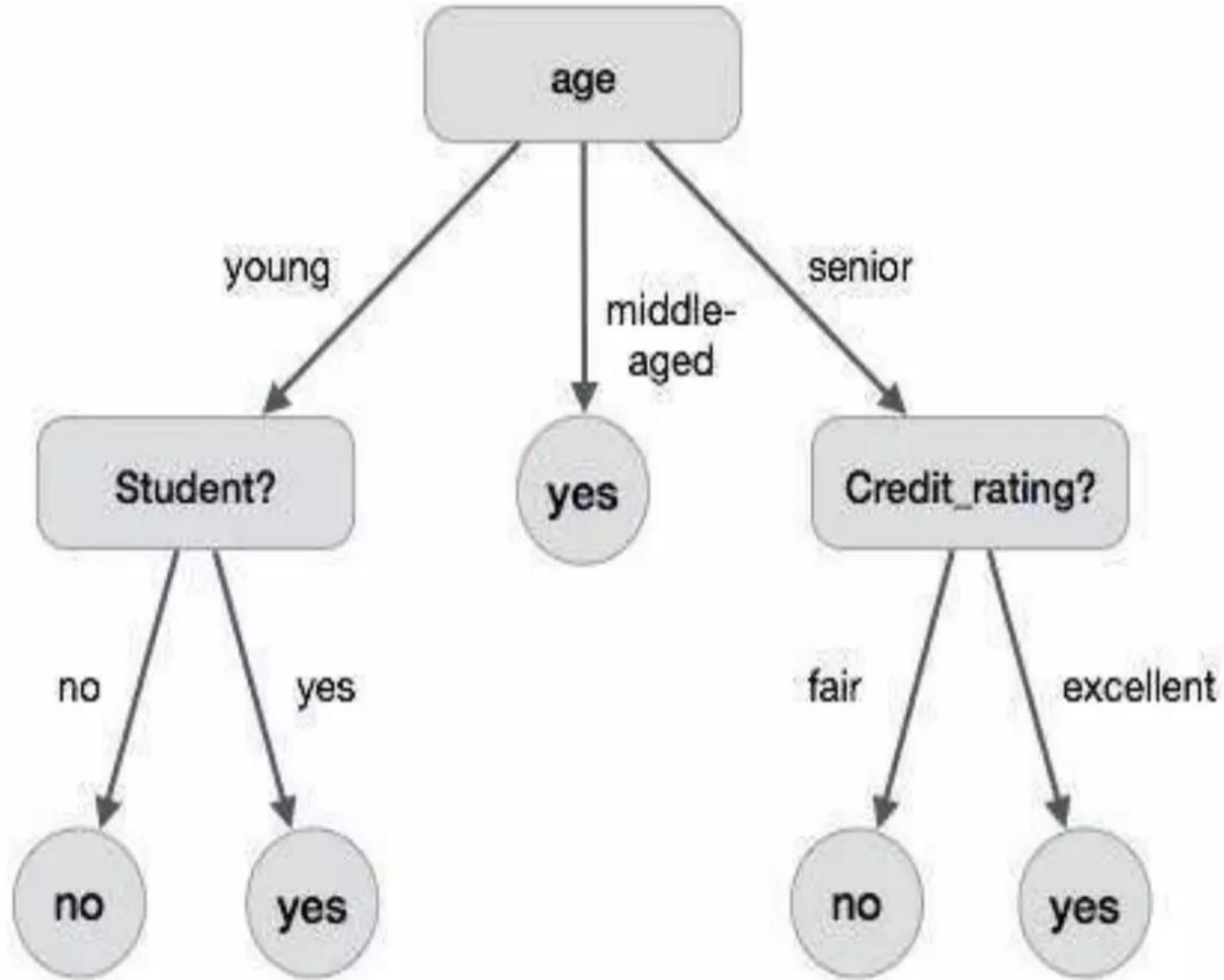
$$y' = -5.706 + 0.008 \times 87 + 0.102 \times 2 = -4.811$$

$$p = \frac{1}{1 + e^{-(b_0 + b_1x_1 + b_2x_2)}} = \frac{1}{1 + e^{-y'}}$$

$$p = \frac{1}{1 + e^{4.811}} = 0.008 \xrightarrow{<0.5} \boxed{N}$$

# Decision Tree

- Decision tree is a simple but powerful learning Paradigm.
- It is a type of classification algorithm for supervised learning.
- A decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision.
- A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes).



- **Decision Trees**
- **ID3 Algorithm**
- **C 4.5 Algorithm**
- **Random Forest**

# Construction of Decision Tree

- 1) First test all attributes and select the one that will function as the best root.
- 2) Break-up the training set into subsets based on the branches of the root node.
- 3) Test the remaining attributes to see which ones fit best underneath the branches of the root node.
- 4) Continue this process for all other branches until

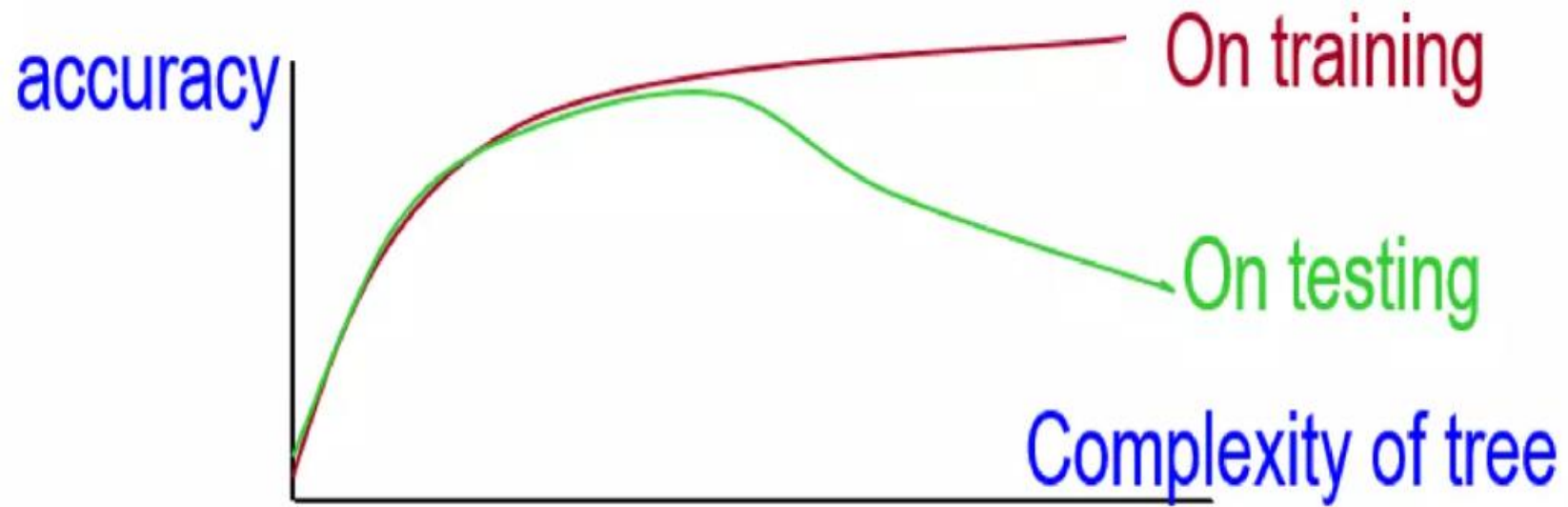


# Limitation of Decision Trees

- Learning a tree that classifies the training data perfectly may not lead to the tree with the **best generalization performance**.
  - There may be **noise** in the training data the tree is fitting
  - The algorithm might be making decisions based on **very little data**.
- A hypothesis  **$h$**  is said to overfit the training data if there is another hypothesis,  **$h'$** , such that ' **$h$** ' has smaller error than  **$h'$**  on the training data but  **$h$**  has larger error on the test data than  **$h'$** .



# Decision Trees

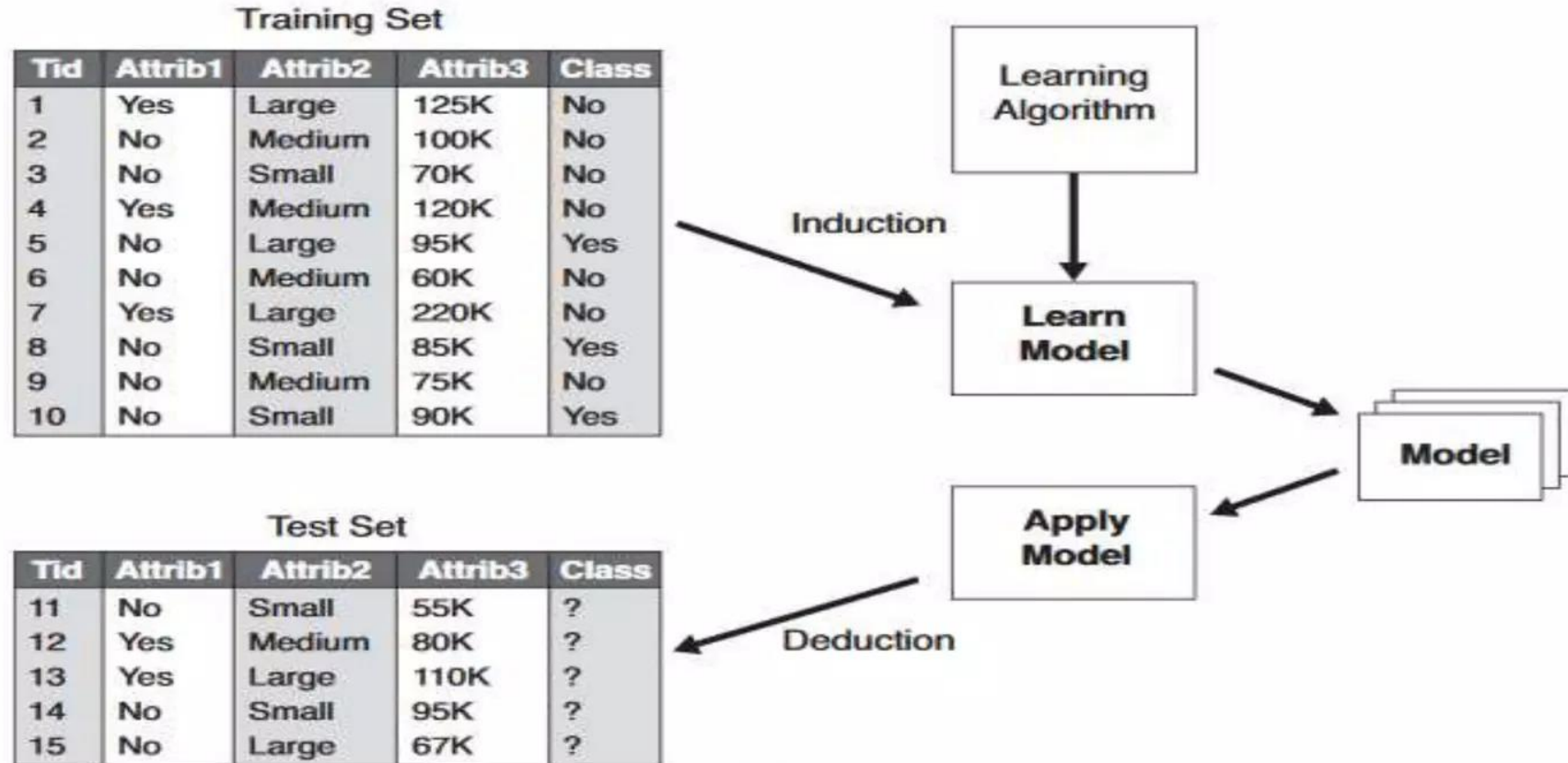


# Evaluation Methods for Decision Trees

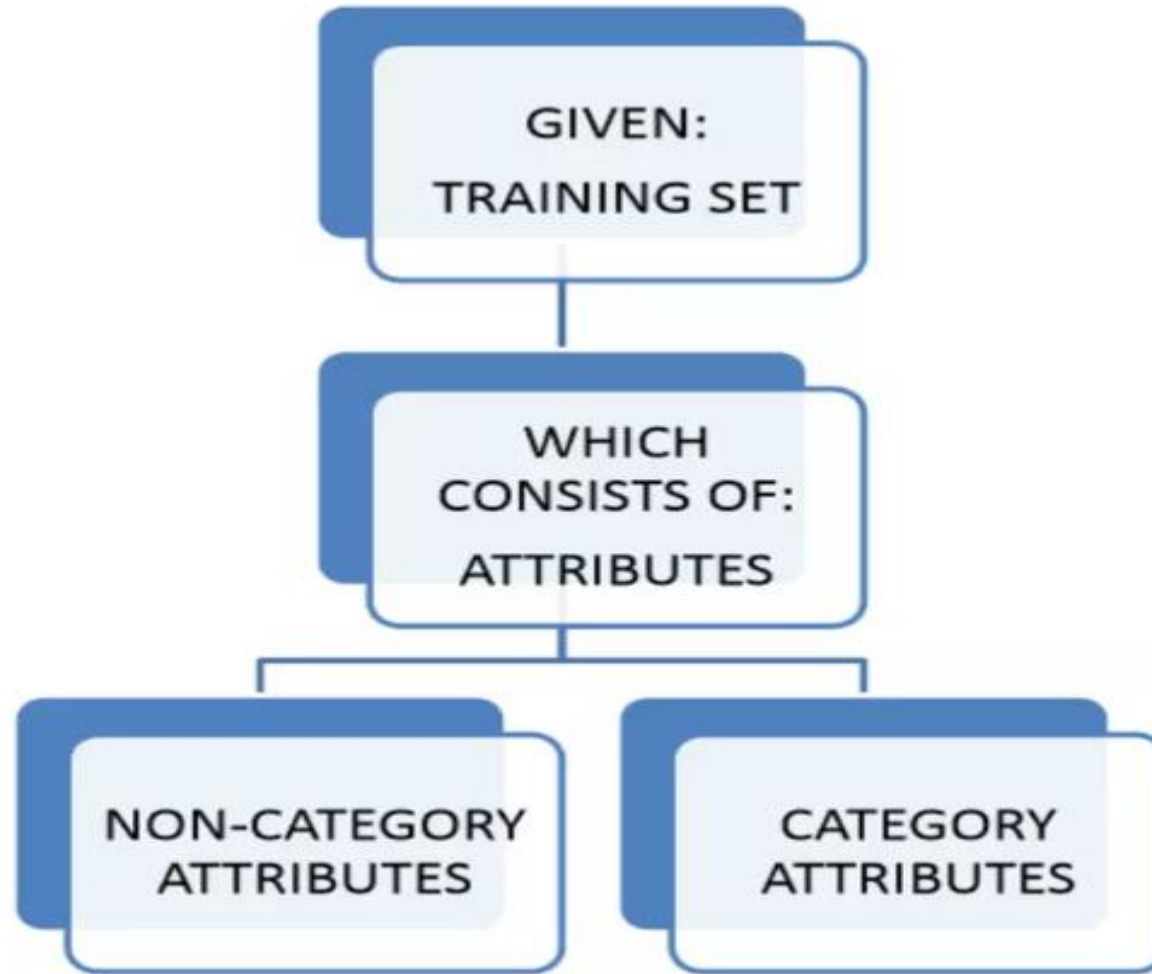
- Two basic approaches
  - **Pre-pruning:** Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - **Post-pruning:** Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Methods for evaluating subtrees to prune:
  - **Cross-validation:** Reserve hold-out set to evaluate utility.
  - **Statistical testing:** Test if the observed regularity can be dismissed as likely to be occur by chance.
  - **Minimum Description Length:** Is the additional complexity of the hypothesis smaller than remembering the exceptions ?

# ID3 (Iterative Dichtomiser 3)

- Invented by J.Ross Quinlan in 1975.
- Used to generate a decision tree from a given data set by employing a top-down, greedy search, to test each attribute at every node of the tree.
- The resulting tree is used to classify future samples.



**Figure 4.3.** General approach for building a classification model.



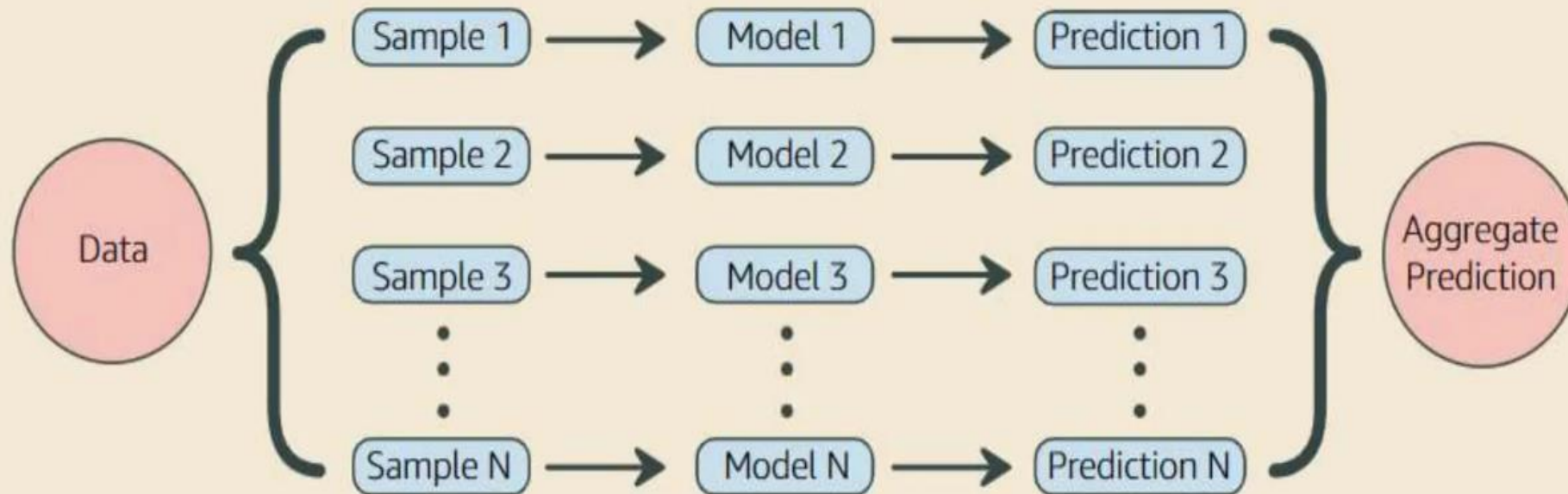


# Decision Tree Algorithm

Strengths	Weaknesses
<ul style="list-style-type: none"><li>• An all-purpose classifier that does well on most problems</li><li>• Highly-automatic learning process can handle numeric or nominal features, missing data</li><li>• Uses only the most important features</li><li>• Can be used on data with relatively few training examples or a very large number</li><li>• Results in a model that can be interpreted without a mathematical background (for relatively small trees)</li><li>• More efficient than other complex models</li></ul>	<ul style="list-style-type: none"><li>• Decision tree models are often biased toward splits on features having a large number of levels</li><li>• It is easy to overfit or underfit the model</li><li>• Can have trouble modeling some relationships due to reliance on axis-parallel splits</li><li>• Small changes in training data can result in large changes to decision logic</li><li>• Large trees can be difficult to interpret and the decisions they make may seem counterintuitive</li></ul>

# Ensemble Learning


Ensemble learning creates a stronger model by aggregating the predictions of multiple weak models. Random Forest is an example of ensemble learning where each model is a decision tree. The idea behind it is- the wisdom of the crowd. The majority vote aggregation can have better accuracy than the individual models.





# Data Preparation

## Dataset Preparation



### **Bootstrap Aggregating (Bagging)**

Creating a different training subset randomly from the original training dataset with replacement is called Bagging. With replacement refers to the bootstrap sample having duplicate elements. Reduces variance, helps to avoid overfitting.

### **Out Of Bag (OOB)**

The out-of-bag dataset represents the remaining elements which were not in the bootstrap dataset. Used for cross validation or to evaluate the performance.

### **Random Subspace Method (Feature Bagging)**

While building the tree, for splitting, a randomly selected subset of the features are used. So, the trees are more different having low correlation.

# Bagging (Bootstrap Aggregating)

- **Definition:**

- Bagging stands for Bootstrap Aggregating.
- It creates multiple versions of the same model using different subsets of the training data and aggregates their predictions (averaging for regression, majority vote for classification).
- Key Concepts of Bagging

- **Bootstrap Sampling:**

- Creates different subsets of data by randomly sampling the training data with replacement.
- Each subset (called a bootstrap sample) can have duplicate instances.

- **Parallel Learning:**

- Each model (usually a decision tree) is trained independently and in parallel on a different subset of data.

- **Final Prediction:**

- Classification: The final prediction is determined by majority voting across all models.
- Regression: The final prediction is the average of the predictions from all models.

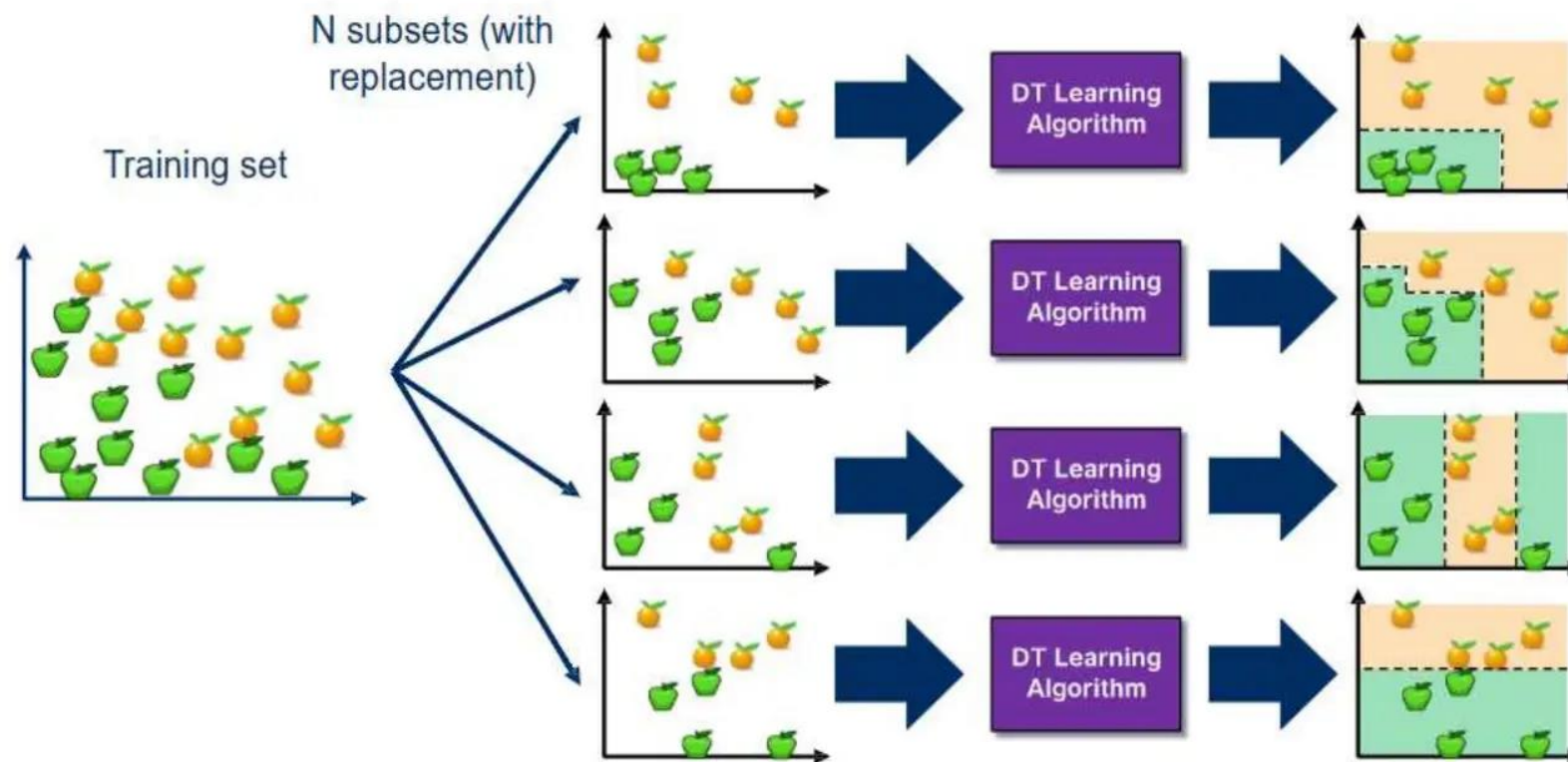
- **Reduces Variance:**

- Since decision trees are high-variance models (prone to overfitting), Bagging reduces the variance by averaging out the predictions from multiple trees.

# Steps in Bagging

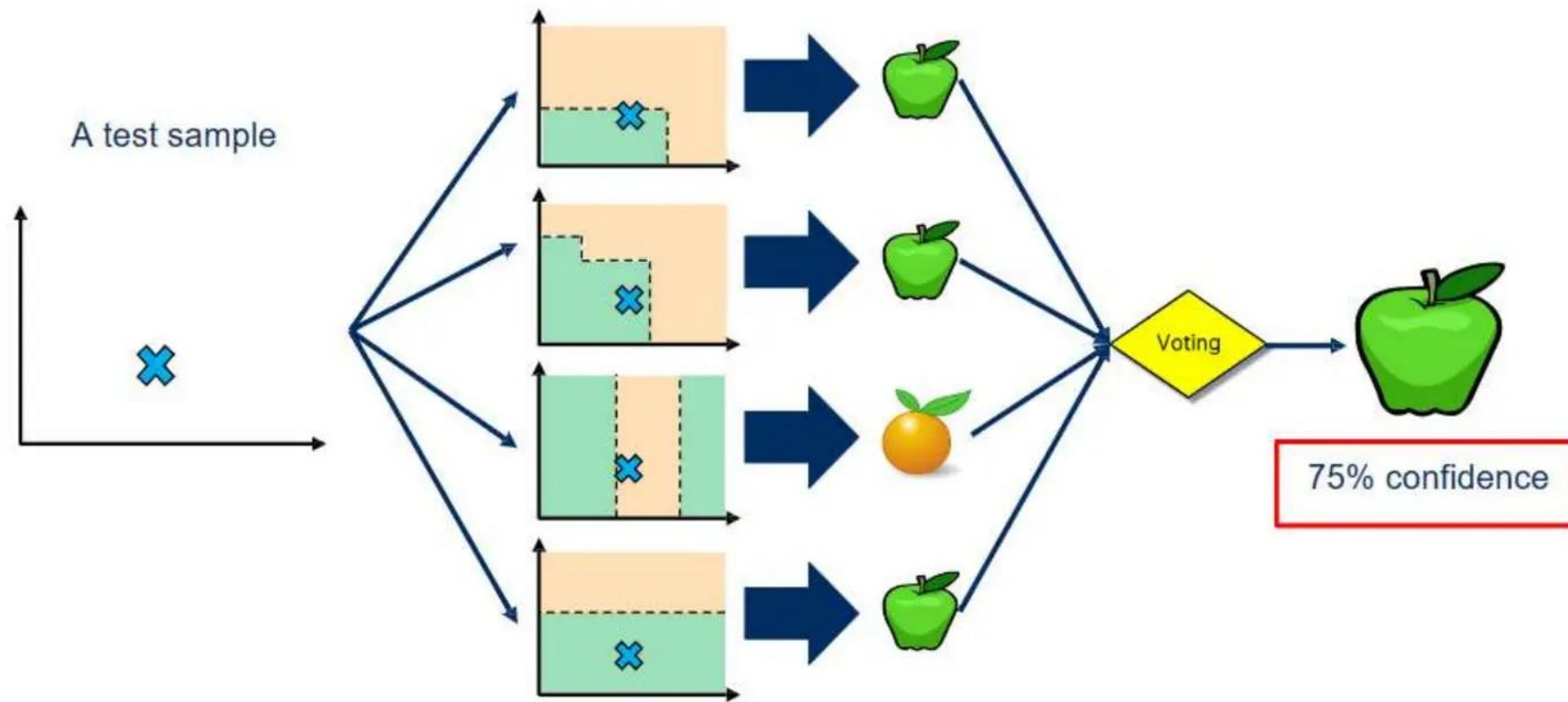
- **Bootstrap Sampling:** Randomly sample the training data with replacement to create multiple subsets (bootstrap samples).
- **Train Weak Learners:** Train a model (e.g., a decision tree) on each subset of the data.
- **Aggregate Predictions:**
  - For classification, use majority voting.
  - For regression, take the average of the predictions.
- **Advantages of Bagging**
  - **Reduces Overfitting:** Helps in reducing overfitting by averaging the results of multiple models, especially useful for high-variance models like decision trees.
  - **Improves Stability and Accuracy:** By aggregating multiple models, it improves the robustness and accuracy of the overall model.
  - **Parallelizable:** Since each model is trained independently, Bagging can be easily parallelized, speeding up training.
- **Disadvantages of Bagging**
  - **Not Effective for High Bias Models:** Bagging mainly reduces variance, so it doesn't help much when the base model has high bias (underfitting).
  - **Increased Computational Cost:** Since multiple models are trained, it requires more computational resources and memory.

# Bagging at training time



R A N D O M   F O R E S T

# Bagging at inference time



# Common Algorithms Using Bagging

- **Random Forest:**

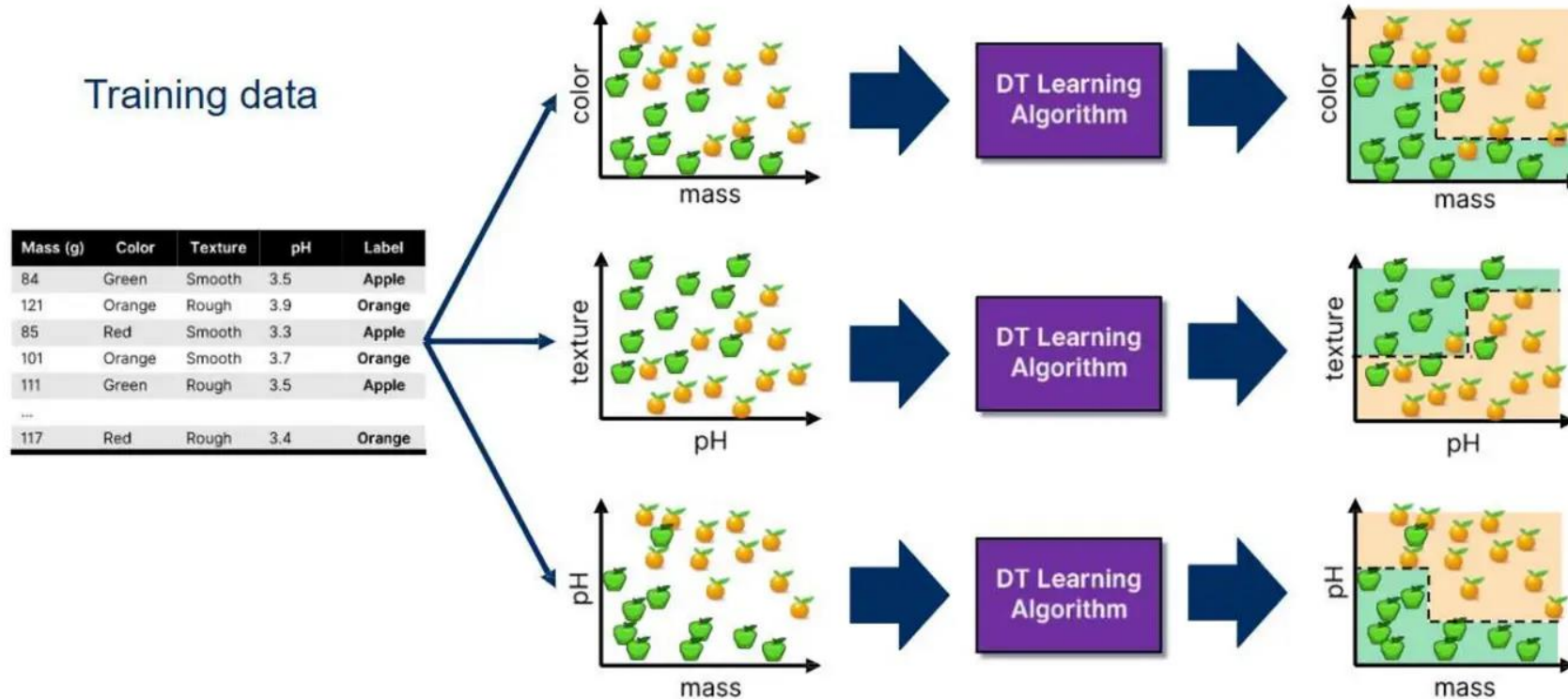
- A popular Bagging-based algorithm that builds multiple decision trees on different bootstrap samples and random subsets of features at each split.

- **Applications of Bagging**

- **Banking and Finance:** Fraud detection, credit scoring.
- **Healthcare:** Disease prediction and diagnosis support.
- **Marketing:** Customer segmentation and behavior prediction.



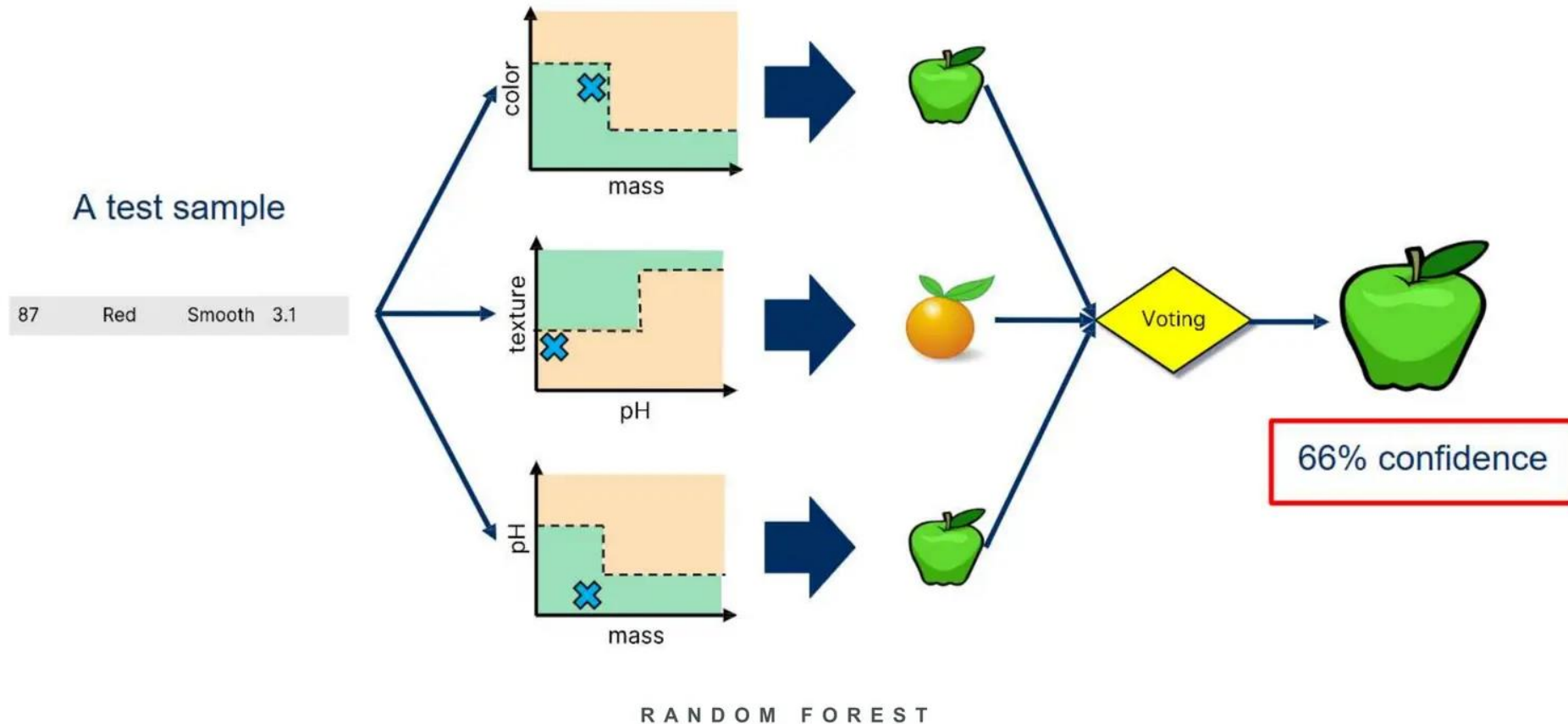
# Random Subspace Method at training time



R A N D O M F O R E S T



# Random Subspace Method at inference time

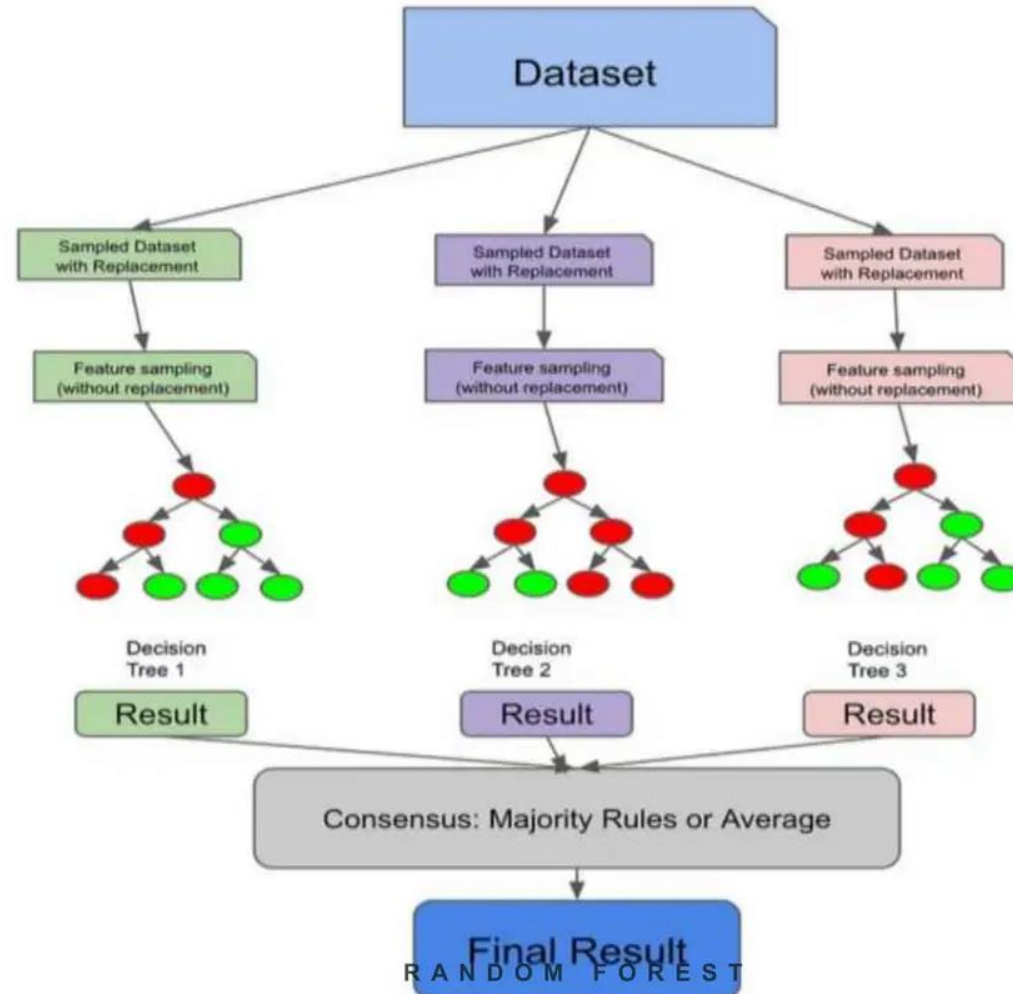


# Definition

---

**The Random Forest algorithm is an ensemble learning method consisting of many decision trees that are built by using bagging and feature bagging which helps to create an uncorrelated forest of trees whose combined prediction is more accurate than that of a single tree. For classification tasks, final prediction is done by taking majority votes and for regression tasks, average of all the individual trees.**

# Random Forest Model



# Why Use Random Forest

---

- Random forests are an effective tool in prediction.
- Forests give results competitive with boosting and adaptive bagging, yet do not progressively change the training set.
- Random inputs and random features produce good results in classification- less so in regression.
- For larger data sets, we can gain accuracy by combining random features with boosting.

# Advantages and Disadvantages

---

## ❖ Advantages

- Versatile uses
- Easy-to-understand hyperparameters
- Classifier doesn't overfit with enough trees

## ❖ Disadvantages

- Increased accuracy requires more trees
- More trees slow down model
- Can't describe relationships within data

# Random Forest Applications

---

- Detects reliable debtors and potential fraudsters in finance
- Verifies medicine components and patient data in healthcare
- Gauges whether customers will like products in e-commerce



# Boosting

- **Definition:**

- Boosting is an ensemble technique that builds models sequentially, where each new model focuses on correcting the errors made by the previous models.
- **Objective:** Reduce bias by converting weak learners into strong learners through weighted learning.

- **Key Concepts of Boosting**

- **Sequential Learning:**

- Models are trained sequentially, with each new model trying to correct the mistakes made by the previous model.

- **Adaptive Learning:**

- Boosting assigns higher weights to the incorrectly predicted instances, forcing the next model to focus more on these difficult cases.

- **Weighted Predictions:**

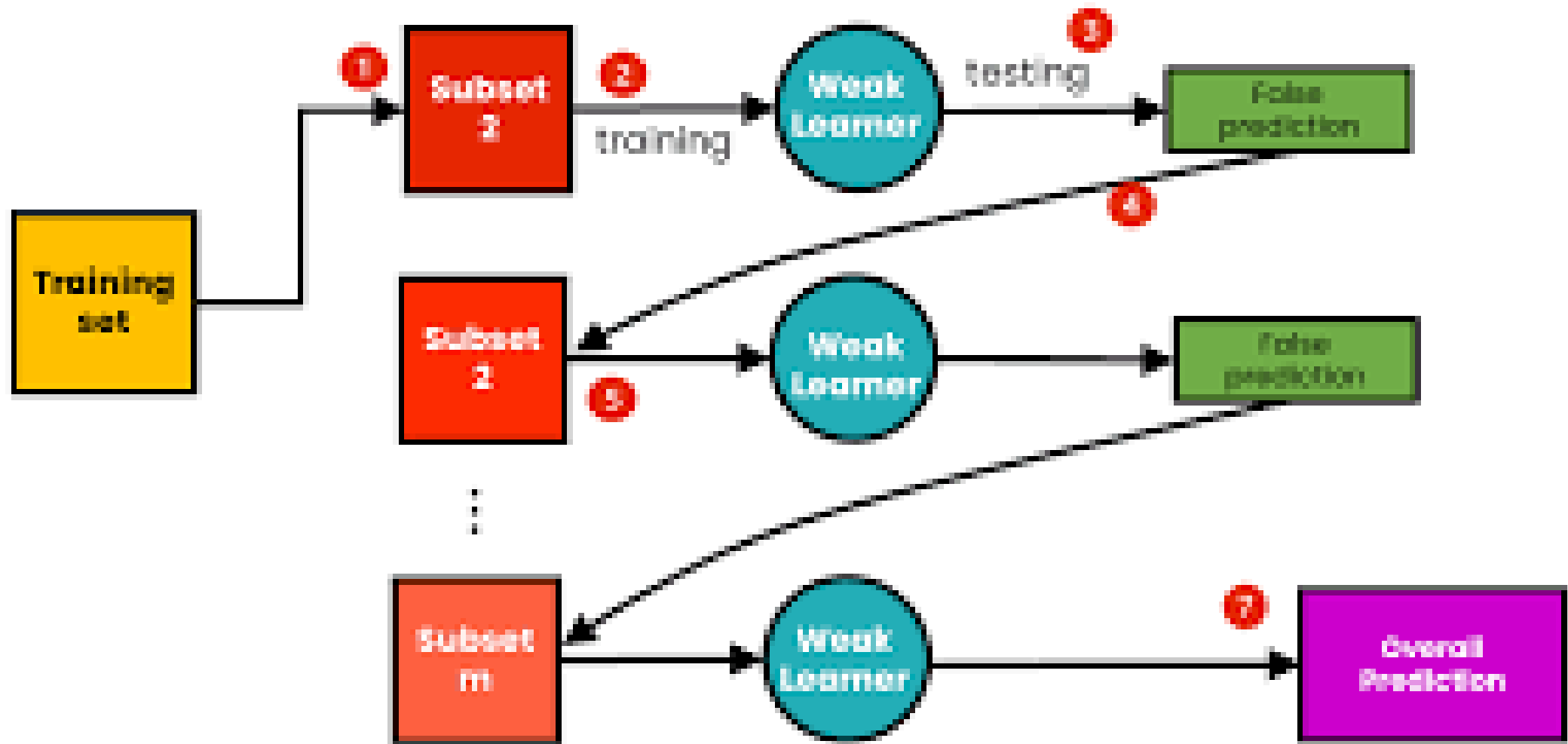
- Each model in the sequence contributes a weighted prediction, and the final prediction is a weighted combination of all models.

- **Reduces Bias:**

- By focusing on correcting mistakes, Boosting reduces the bias of the model, resulting in stronger learners.



# The Process of Boosting



# Steps in Boosting

- **Initialize Weights:** All instances are assigned equal weights initially.
  - **Train Weak Learner:** Train a weak model (e.g., a shallow decision tree) on the training data.
  - **Update Weights:** Increase the weights of misclassified instances to make them more important for the next model.
  - **Train Next Model:** Train the next weak learner on the updated data, giving more attention to previously misclassified instances.
  - **Aggregate Predictions:** Combine the predictions from all models using weighted voting or averaging.
- 
- **Advantages of Boosting**
    - **Reduces Bias and Variance:** Boosting improves model performance by addressing both bias and variance, making it more effective for complex problems.
    - **Improves Accuracy:** Boosting often results in highly accurate models by iteratively refining the predictions.
    - **Handles Imbalanced Data:** Boosting can be effective in handling imbalanced datasets, as it focuses on difficult-to-predict instances.
- 
- **Disadvantages of Boosting**
    - **Prone to Overfitting:** Boosting can overfit the training data, especially if the model is trained too long or without proper regularization.
    - **Sequential Learning (Slower Training):** Since models are trained one after another, Boosting can be slower compared to Bagging, which is parallelizable.
    - **Sensitive to Noisy Data:** Boosting models tend to be more sensitive to noise in the data, as they give higher importance to difficult cases, which may sometimes be noise.

# Types of Boosting Algorithms

- **AdaBoost (Adaptive Boosting):**
  - One of the earliest and simplest forms of Boosting.
  - Adjusts the weights of misclassified samples in each iteration, focusing on difficult cases.
- **Gradient Boosting:**
  - Models the residuals (errors) from the previous models, and the new model is trained to correct these residuals.
  - Gradient Boosted Decision Trees (GBDT) is a popular version.
- **XGBoost (Extreme Gradient Boosting):**
  - An optimized version of Gradient Boosting that includes regularization to prevent overfitting and provides faster performance.
- **LightGBM:**
  - A gradient boosting framework that builds trees based on a leaf-wise split instead of level-wise, making it faster for large datasets.
- **CatBoost:**
  - A gradient boosting algorithm optimized for categorical data and reducing the need for preprocessing.
- **Applications of Boosting**
  - **Financial Forecasting:** Predicting stock prices, credit scoring.
  - **Marketing:** Customer churn prediction, click-through rate (CTR) prediction.
  - **Healthcare:** Predicting patient outcomes, disease diagnosis.
  - **Image Recognition:** Object detection and classification.
  - **Natural Language Processing:** Sentiment analysis, spam detection.

# Comparision

Aspect	Bagging	Boosting
Goal	Reduce variance (overfitting)	Reduce bias (underfitting)
Learning Style	Parallel (models trained independently)	Sequential (models trained iteratively)
Model Focus	Each model is independent of others	Each model focuses on correcting errors
Model Complexity	Simple averaging or voting	Weighted voting or averaging of models
Overfitting Risk	Lower risk of overfitting	Higher risk of overfitting without control
Speed	Faster training (parallelizable)	Slower training (sequential)
Computational Cost	Lower computational cost	Higher computational cost due to sequential learning
Use Case	High-variance models (e.g., decision trees)	High-bias models or complex problems

# Dimensionality Reduction

- **Definition:**

- Dimensionality Reduction refers to the process of reducing the number of input features (dimensions) in a dataset while retaining as much relevant information as possible.
- Helps in simplifying models, reducing computation, and avoiding overfitting.

- **Reasons for Dimensionality Reduction:**

- **Curse of Dimensionality:** As the number of dimensions increases, the data becomes sparse and harder to interpret, which affects model performance.
- **Computational Efficiency:** Reducing dimensions speeds up training and testing of machine learning models.
- **Visualization:** High-dimensional data can be hard to visualize; dimensionality reduction allows for visualization in 2D or 3D.
- **Noise Reduction:** Removes redundant or irrelevant features, improving model performance and generalization.

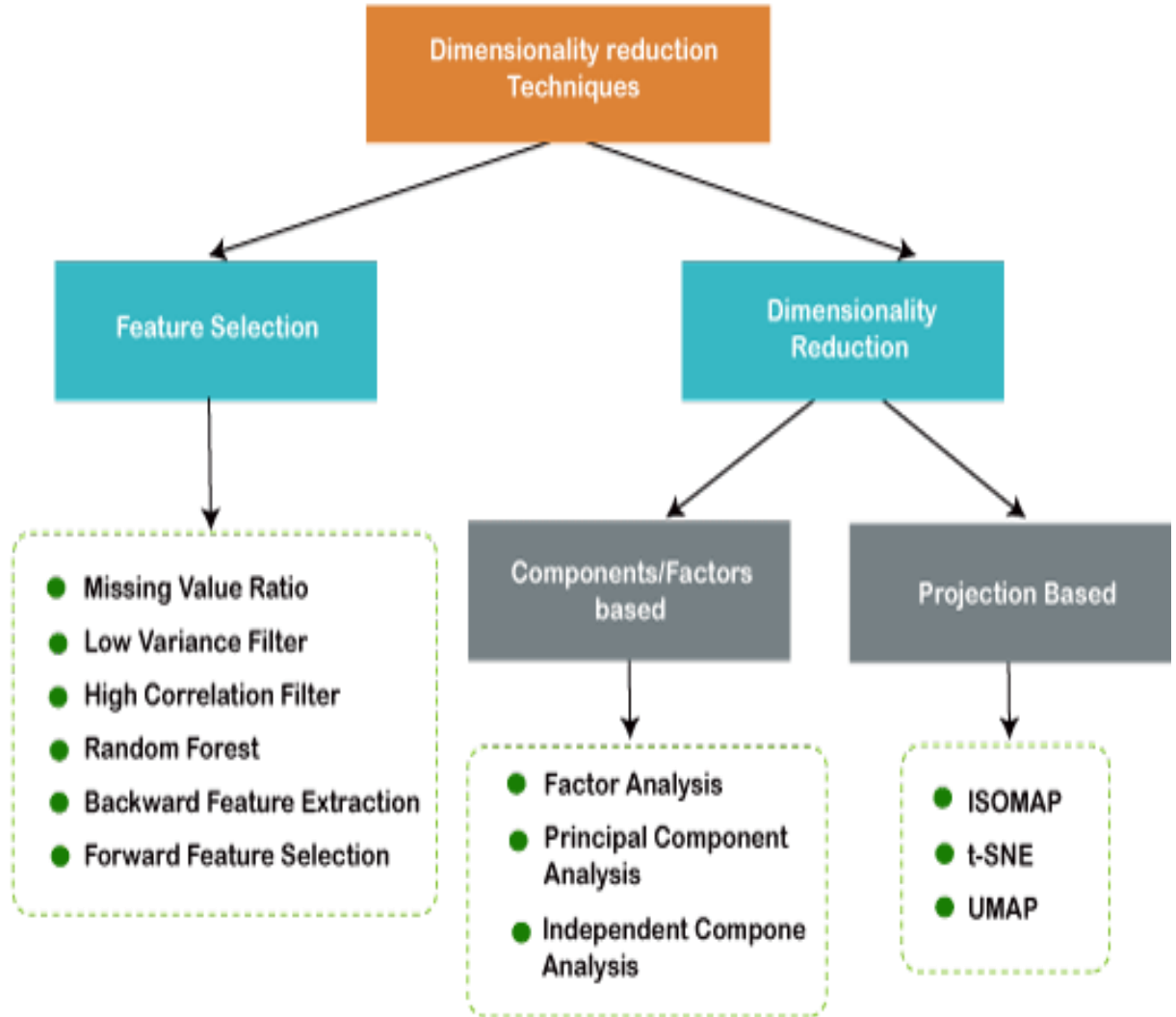
# Types of Dimensionality Reduction Techniques:

- **Feature Selection:**

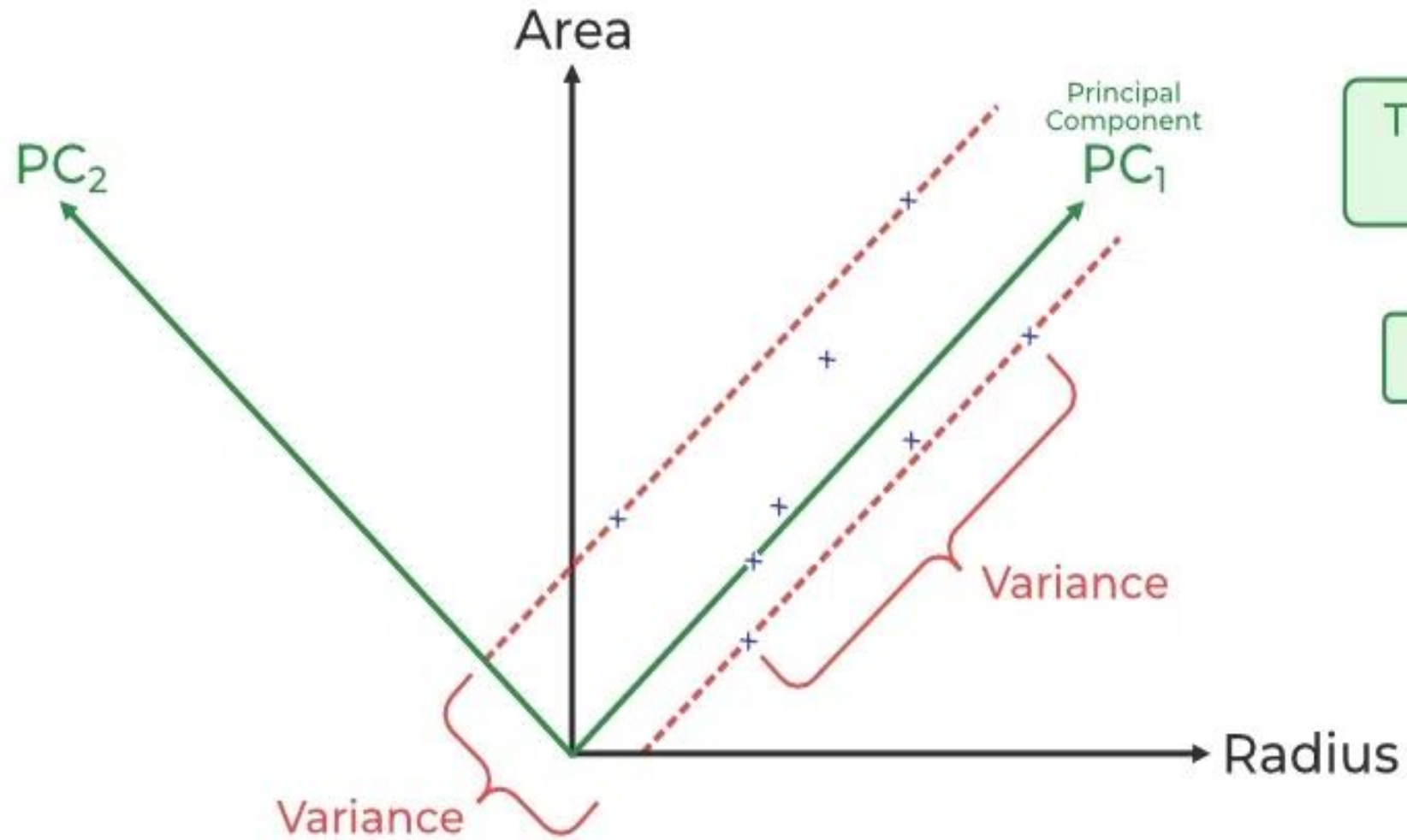
- Selecting the most important features based on their relevance.
- Examples: Recursive Feature Elimination (RFE), SelectKBest.

- **Feature Extraction:**

- Transforming the original features into a lower-dimensional space.
- Examples: **Principal Component Analysis (PCA)**, Linear Discriminant Analysis (LDA), t-SNE.

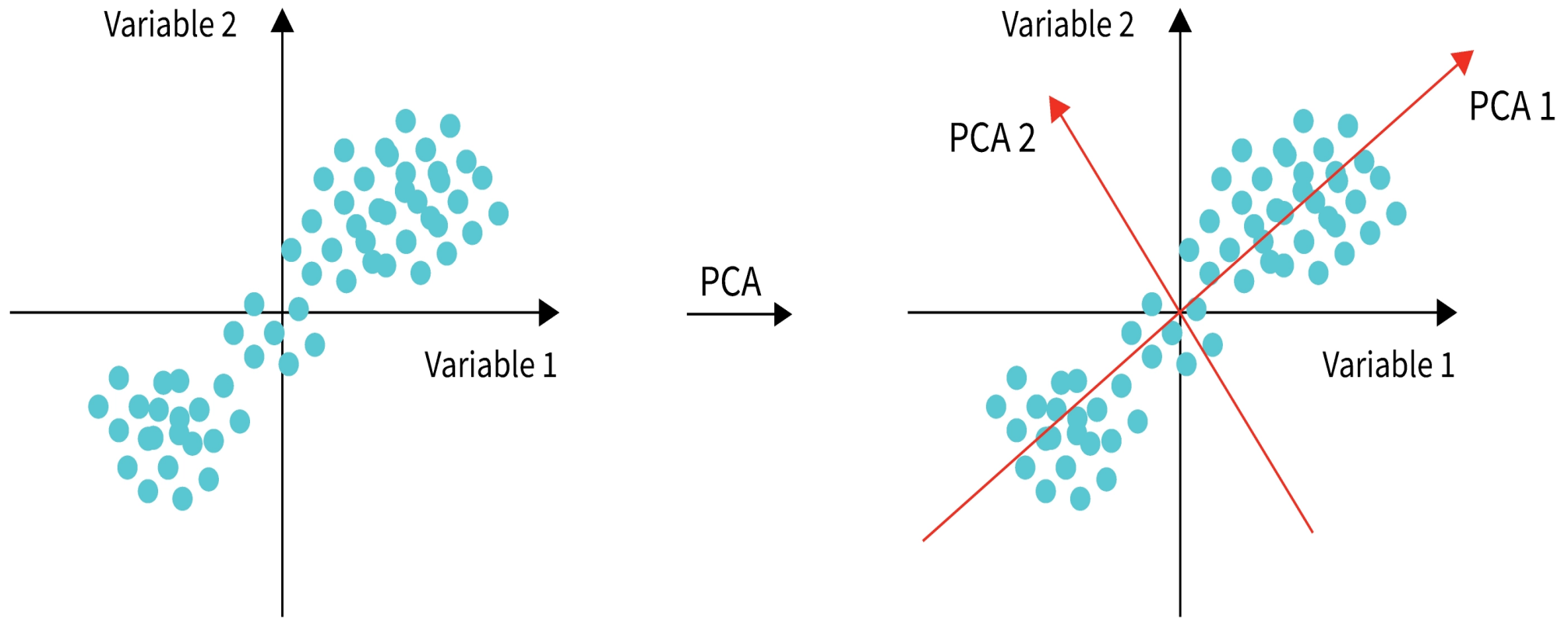






Transformation  
2D → 1D

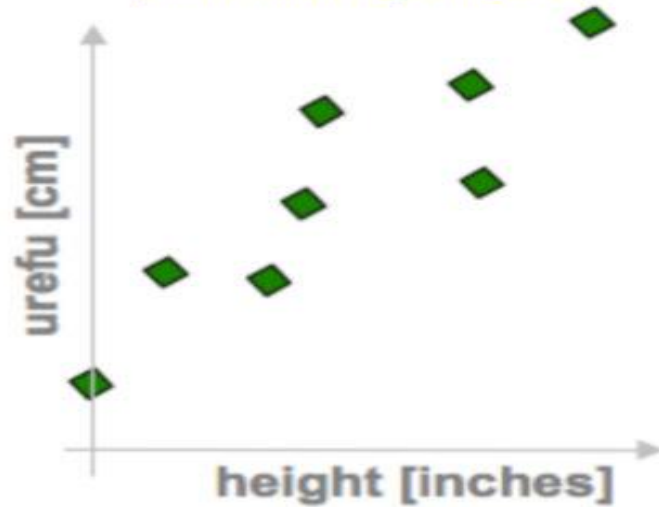
$PC_1 > PC_2$



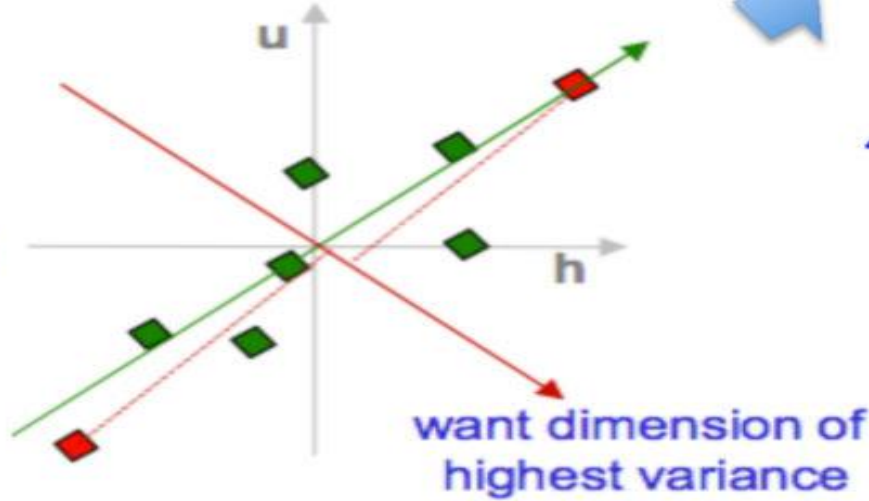
# PCA in a nutshell

## 1. correlated hi-d data

("urefu" means "height" in Swahili)



## 2. center the points



## 3. compute covariance matrix

$$\begin{matrix} & h & u \\ h & \begin{pmatrix} 2.0 & 0.8 \end{pmatrix} \\ u & \begin{pmatrix} 0.8 & 0.6 \end{pmatrix} \end{matrix} \rightarrow \text{cov}(h,u) = \frac{1}{n} \sum_{i=1}^n h_i u_i$$

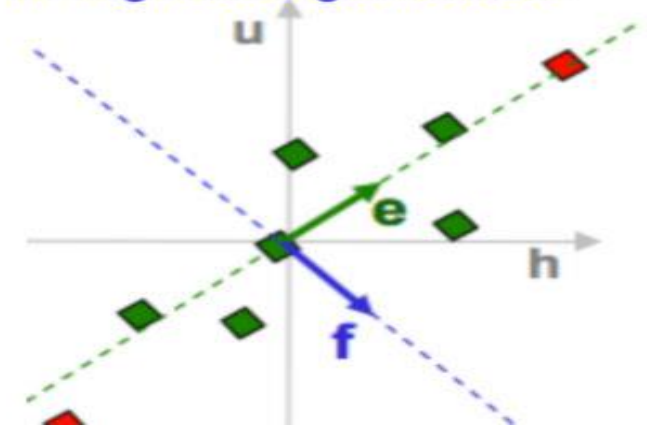
## 4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

$\text{eig}(\text{cov}(\text{data}))$

## 5. pick $m < d$ eigenvectors w. highest eigenvalues

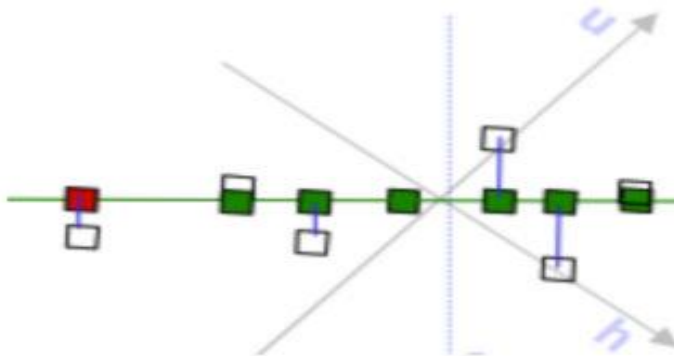


## 6. project data points to those eigenvectors

$$x'_e = x^T e = \sum_{j=1}^a x_{ij} e_j$$

A diagram showing a data point (red circle) being projected onto the first principal component 'e' (green line). The projection is shown as a red arrow from the point to the line. Other axes are labeled with blue dashed lines and labels like +1, +2, +3.

## 7. uncorrelated low-d data



# Principal Component Analysis (PCA)

- **Definition:**

- PCA is a widely used technique for feature extraction that transforms high-dimensional data into a lower-dimensional space by finding the principal components.
- Principal components are linear combinations of the original features that capture the maximum variance in the data.

- **Key Concepts of PCA:**

- **Principal Components:**

- Principal Components (PCs) are new features (axes) that are linear combinations of the original features.
- The first principal component captures the largest amount of variance in the data, the second captures the next highest, and so on.

- **Orthogonal Axes:**

- All principal components are orthogonal (perpendicular) to each other, ensuring that each component captures unique information.

- **Variance Explained:**

- Each principal component captures a specific amount of variance from the original dataset.
- The goal is to retain as much variance as possible while reducing dimensions.

- **Covariance Matrix:**

- PCA identifies directions (principal components) where the data has the highest variance by calculating the covariance matrix of the features.
- It then computes the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the direction, and the eigenvalues represent the magnitude of variance along each direction.

- **Dimensionality Reduction:**

- Once the principal components are calculated, PCA selects a subset of them to form a reduced feature set.
- The number of selected components is based on the amount of variance one wants to retain (often 95%).

# Steps in PCA:

- **Standardize the Data:**
  - It is essential to standardize the features (mean = 0, variance = 1) since PCA is affected by the scale of the data.
- **Compute the Covariance Matrix:**
  - Calculate the covariance matrix of the standardized data to understand the relationships between features.
- **Eigenvalues and Eigenvectors:**
  - Compute the eigenvalues and eigenvectors of the covariance matrix.
  - Eigenvectors represent the directions of maximum variance (principal components).
  - Eigenvalues represent the magnitude of the variance captured by each eigenvector.
- **Select Principal Components:**
  - Rank the principal components based on their eigenvalues and select the top components that capture most of the variance (e.g., 95% of total variance).
- **Project Data:**
  - Transform the original data into the new space defined by the selected principal components.

- **Advantages of PCA:**

- **Reduces Overfitting:** By reducing the number of features, PCA reduces the risk of overfitting, especially when working with high-dimensional data.
- **Improves Performance:** PCA simplifies the model and speeds up training and testing times.
- **Captures Maximum Variance:** PCA selects components that capture the maximum variance in the data, leading to more meaningful projections.
- **Helps with Visualization:** PCA can reduce data to two or three dimensions, making it easier to visualize complex datasets.

- **Disadvantages of PCA:**

- **Interpretability:** The new principal components are linear combinations of the original features, making them harder to interpret compared to the original features.
- **Linear Assumption:** PCA assumes that the relationships between variables are linear. It may not capture more complex, nonlinear structures.
- **Sensitive to Scaling:** PCA is sensitive to the scaling of features. Features with larger variance dominate, so it's crucial to standardize the data before applying PCA.



# Use Cases of PCA:

- **Image Compression:**

- PCA is commonly used in image processing to reduce the dimensionality of images while preserving important information.

- **Noise Reduction:**

- PCA can filter out noise from data by retaining only the most significant components, discarding less informative ones.

- **Visualization of High-Dimensional Data:**

- PCA helps visualize data in 2D or 3D, useful in exploratory data analysis.

- **Gene Expression Analysis:**

- In bioinformatics, PCA is used to reduce the dimensionality of gene expression data for easier interpretation and visualization.

# Singular Value Decomposition (SVD)

- Definition:
  - Singular Value Decomposition (SVD) is a matrix factorization technique that decomposes a matrix into three smaller matrices:  $U$ ,  $\Sigma$  (Sigma), and  $V^T$ .
  - It is a generalization of the eigenvalue decomposition for rectangular matrices.
  - SVD is widely used in dimensionality reduction, noise reduction, and data compression.
- Mathematical Formulation:
  - For a given matrix  $A$  ( $m \times n$ ), the SVD is represented as:

$$A = U\Sigma V^T$$

- Where:
  - $A$ : The original  $m \times n$  matrix to be decomposed.
  - $U$ : An  $m \times m$  orthogonal matrix. The columns of  $U$  are called left singular vectors.
  - $\Sigma$ : An  $m \times n$  diagonal matrix containing singular values.
  - $V^T$ : The transpose of an  $n \times n$  orthogonal matrix. The columns of  $V$  are called right singular vectors.

- Components of SVD:
  - Matrix  $U$  (Left Singular Vectors):
    - The columns of  $U$  are orthogonal and form a basis for the column space of  $A$ .
    - Represents directions in the original space that are orthogonal to each other.
- Matrix  $\Sigma$  (Singular Values):
  - A diagonal matrix containing non-negative singular values.
  - The singular values represent the strength of each corresponding singular vector.
  - Larger singular values capture more important information in the data.
- Matrix  $V^T$  (Right Singular Vectors):
- The rows of  $V^T$  (or the columns of  $V$ ) are orthogonal and form a basis for the row space of  $A$ .

# Applications of SVD:

- **Dimensionality Reduction:**

- SVD can reduce the dimensions of high-dimensional data by keeping only the top singular values and corresponding singular vectors. This is known as Truncated SVD.
- Commonly used in Principal Component Analysis (PCA), where PCA is essentially the application of SVD to centered data.

- **Latent Semantic Analysis (LSA):**

- Used in Natural Language Processing (NLP) for topic modeling and information retrieval. SVD is applied to a term-document matrix to capture underlying patterns (latent topics).

- **Image Compression:**

- SVD is used to compress images by keeping only the largest singular values, which preserve the most important visual information, while discarding smaller singular values (representing less important details).

- **Noise Reduction:**

- SVD can be used to remove noise from data by eliminating small singular values associated with noisy components.

- **Collaborative Filtering in Recommender Systems:**

- SVD is applied to user-item matrices to predict missing values and make recommendations based on the latent factors extracted from user-item interactions.

# Manifold Learning

- **Definition:**

- Manifold learning is a type of dimensionality reduction technique that assumes high-dimensional data lie on a lower-dimensional manifold.

- **Purpose:**

- To find a lower-dimensional representation of data that preserves its intrinsic structure.

- **Key Concepts**

- **Manifold:** A topological space that locally resembles Euclidean space. In data analysis, it's the space on which high-dimensional data points are assumed to lie.
- **Dimensionality Reduction:** The process of reducing the number of random variables under consideration, aiming to simplify the model without losing significant information.

# Techniques

- **Principal Component Analysis (PCA)**
  - Linear technique
  - Finds directions (principal components) that maximize variance in data
- **Isomap**
  - Extension of Multi-Dimensional Scaling (MDS)
  - Preserves global geometric properties by using geodesic distances
- **Locally Linear Embedding (LLE)**
  - Preserves local relationships
  - Based on linear reconstruction of local neighborhoods
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**
  - Non-linear technique
  - Preserves pairwise similarities, emphasizing local structure
- **Laplacian Eigenmaps**
  - Preserves local structure through graph-based approach
  - Utilizes eigenvectors of Laplacian matrix

## Applications

- Data visualization
- Noise reduction
- Feature extraction
- Image and speech processing



# Diffusion Maps

- **Definition:**

- A non-linear dimensionality reduction technique based on the diffusion process over a graph constructed from the data.

- **Purpose:**

- To reveal the intrinsic geometric structure of the data by analyzing the diffusion of data points in a graph.

- **Key Concepts**

- **Diffusion Process:**

- Simulates the spreading of heat or particles over the data graph.

- **Diffusion Operator:**

- Describes how a quantity diffuses over time.

- **Diffusion Coordinates:**

- Reduced-dimensional representation that captures the diffusion behavior.

- **Advantages**

- Captures global and local data structure.
- Effective for non-linear dimensionality reduction.

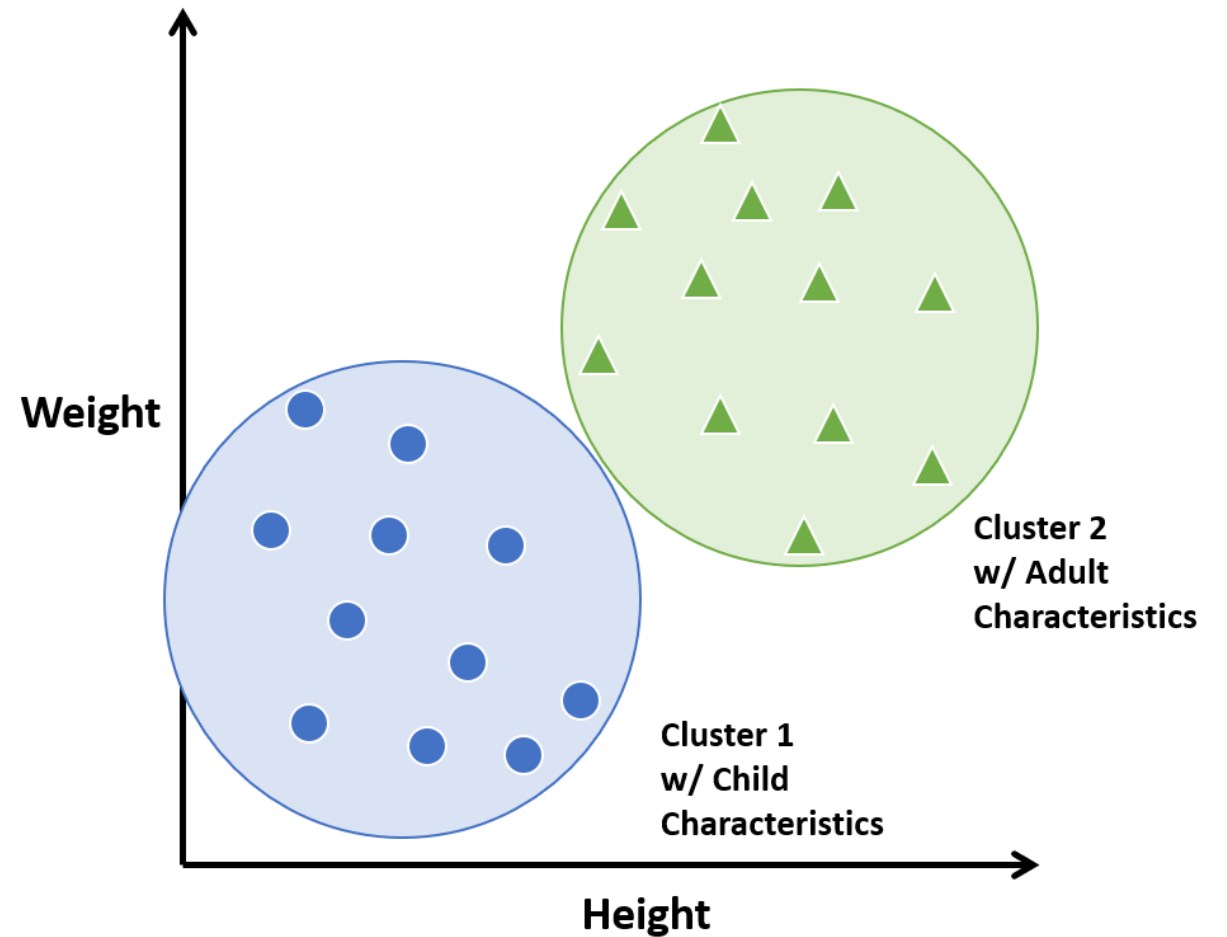
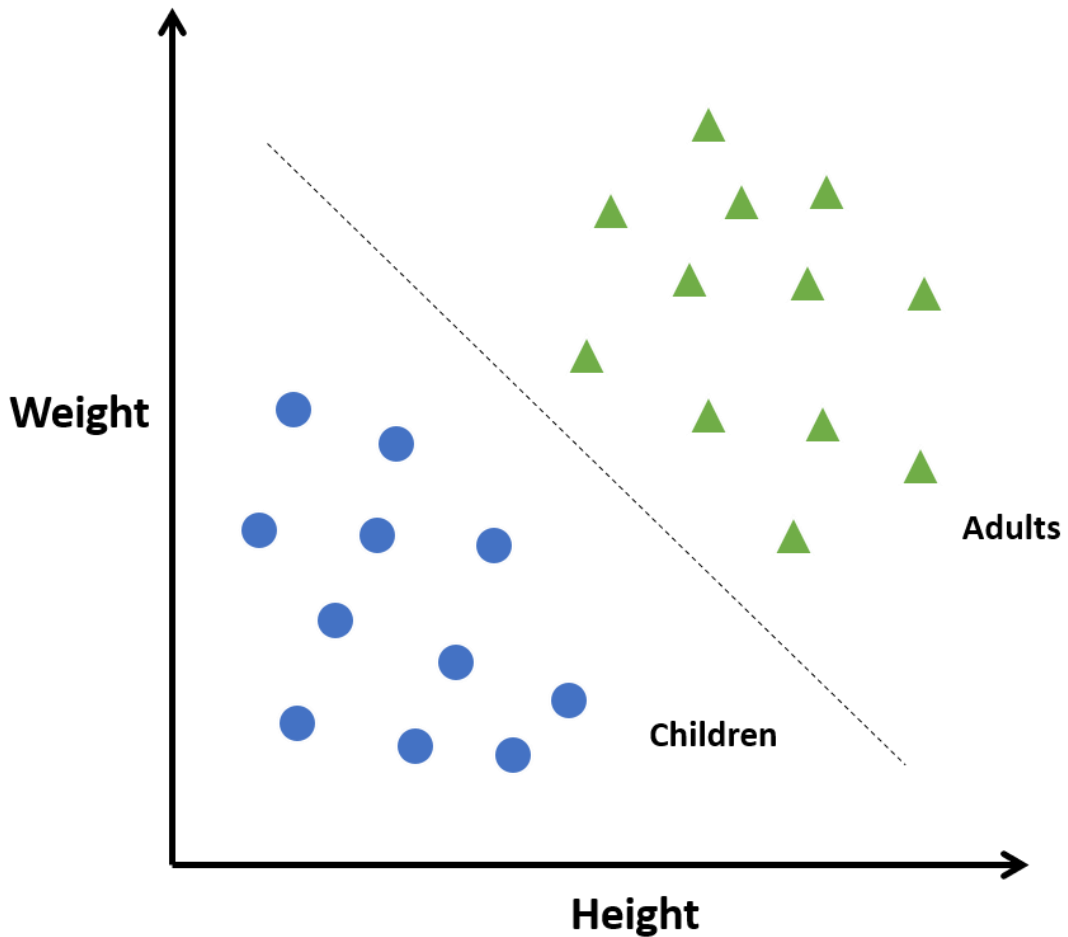
- **Applications**

- Data visualization
- Image processing
- Manifold learning
- Clustering

# Classification

vs

# Clustering



# Spectral Clustering

- **Definition:**

- Spectral clustering is a technique that uses the eigenvalues of a similarity matrix to reduce dimensionality before applying a clustering algorithm like K-means.

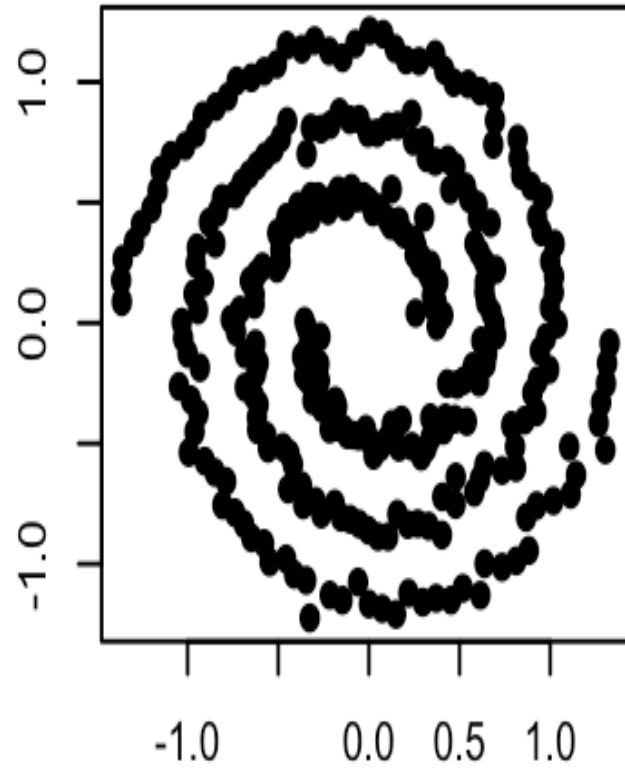
- **Purpose:**

- To group data into clusters by capturing the global structure of the data through the eigenstructure of the graph Laplacian.

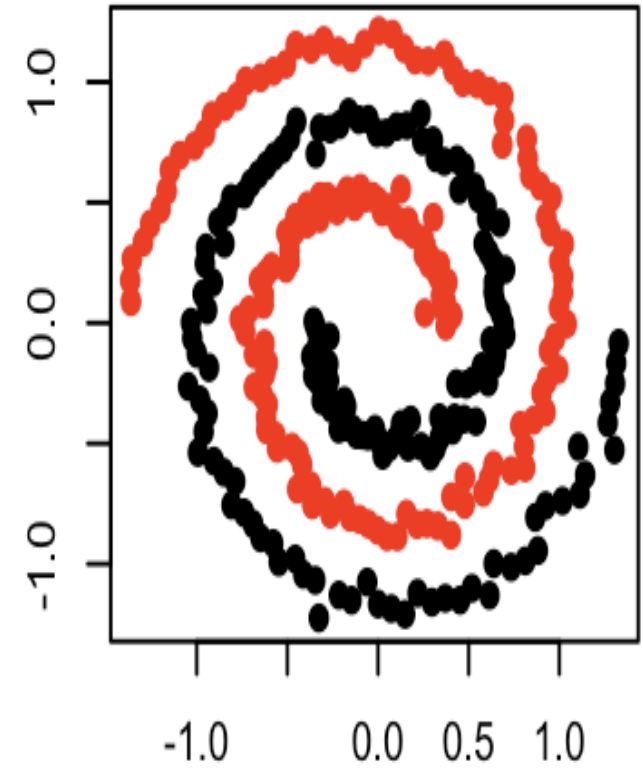
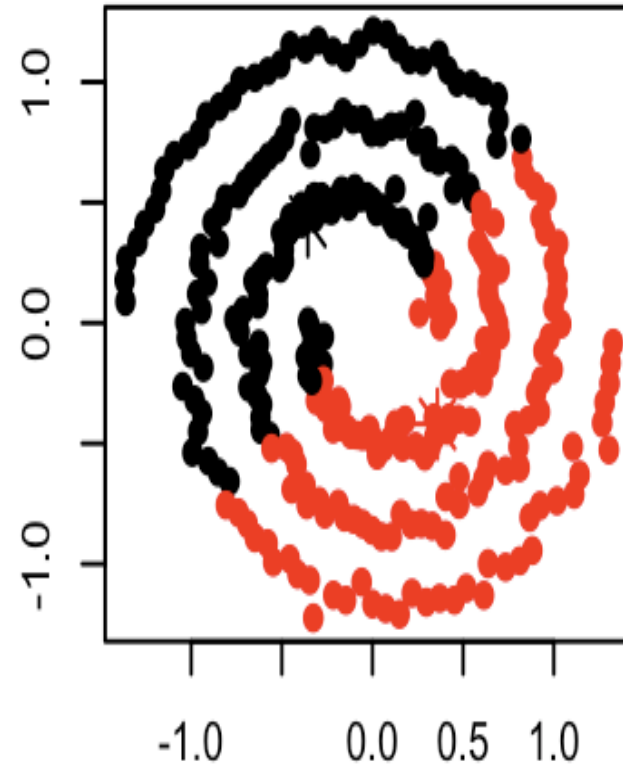
- **Key Concepts**

- **Similarity Matrix:** A matrix that represents the pairwise similarity between data points.
- **Graph Laplacian:** A matrix derived from the similarity matrix that reflects the structure of the graph formed by data points.
- **Eigenvalues and Eigenvectors:** Used to capture the structure of the graph and transform the data into a lower-dimensional space.

**K-means**



**Spectral clustering**



- **Advantages**
- **Captures Complex Structures:** Effective for identifying clusters with complex shapes.
- **Robust to Noise:** More robust to noise and outliers compared to traditional clustering methods.
- **Disadvantages**
- **Computationally Intensive:** Eigen decomposition can be expensive for large datasets.
- **Choice of Parameters:** Sensitivity to the choice of similarity function and parameters like (  $\sigma$  ) in the Gaussian kernel.
- **Applications**
- Image segmentation
- Social network analysis
- Biological data analysis
- Document clustering



