A0597203 AI Business Applications

Introduction to Machine Learning

Introduction to Machine Learning

In this part of the course, we will cover 3 algorithms in Machine Learning:

1. Regression (Supervised Learning)
2. Classification (Supervised Learning)
3. Clustering (Unsupervised Learning)

What Is Linear Regression?

Purpose: Predict a continuous numeric outcome (dependent variable) using one or more independent variables.

Use Case Example:

Problem	Example Features
House Price Prediction	Location, size (sqft), number of bedrooms, age of property
Sales Forecasting	Advertising budget, seasonality, past sales, promotions
Stock Price Prediction	Trading volume, past prices, news sentiment, technical indicators
Temperature Prediction	Date, time of day, humidity, wind speed, cloud cover
Medical Cost Estimation	Age, BMI, smoking status, number of children, region

Simple Linear Regression

- In **simple** linear regression, we use one independent variable `X` to predict `Y`.
- **Model equation:**

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

where:

- Y: dependent variable
- X: independent variable (predictor)
- β_0: intercept
- β_1: slope (effect of X on Y)
- ε: error term (difference between actual and predicted values)

Goal of Regression

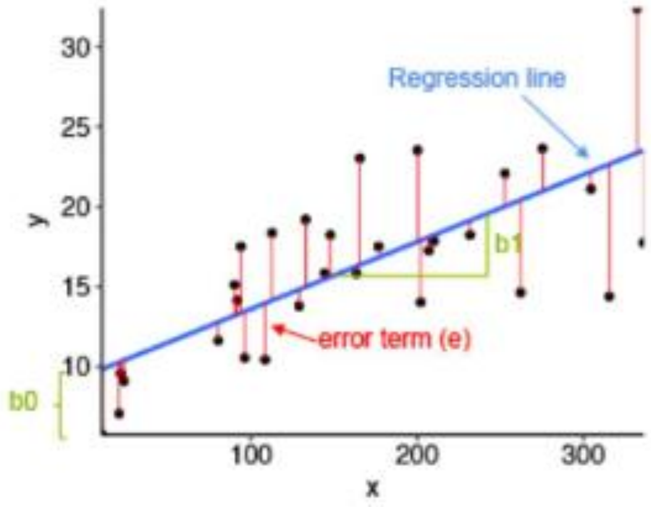
- Find values of β_0 and β_1 that minimize the **Sum of Squared Errors (SSE)**:

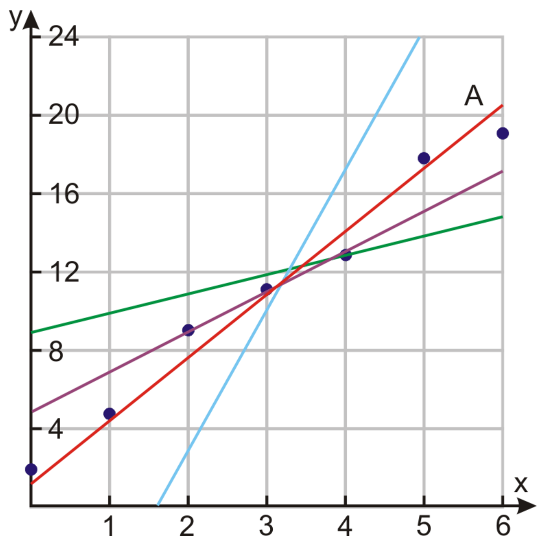
$$SSE = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Use **Ordinary Least Squares (OLS)** to estimate coefficients.

Interpreting Coefficients

- β_1: For every unit increase in X, Y is expected to increase by β_1, holding all else constant.





Solving Simple Linear Regression using the Closed Form Method

The closed-form solution for Simple Linear Regression (SLR) is a direct mathematical formula used to compute the slope ($\beta 1$) and intercept ($\beta 0$) of the best-fit line without iterative optimization.

1. **Slope (β_1)** is given by:

$$\beta_1 = \frac{\mathrm{Cov}(X, y)}{\mathrm{Var}(X)}$$

Where:

- $\mathrm{Cov}(X, y)$ is the covariance between X and y,
- $\mathrm{Var}(X)$ is the variance of X.

2. **Intercept (β_0)** is calculated as:

$$\beta_0 = \bar{y} - \beta_1 \bar{X}$$

Where:

- $\bar{y}$ is the mean of the dependent variable y,
- $\bar{X}$ is the mean of the independent variable X.

مبدأ اشتقاق المعادلة – للاطلاع فقط

We have $y_i = a + bx_i + \varepsilon_i$, with a, b chosen to minimize $S = \sum (y_i - a - bx_i)^2$.

From $\frac{\partial S}{\partial a} = 0$:
$-2\sum [y_i - a - bx_i] = 0 \;\Rightarrow\; \sum y_i - na - b\sum x_i = 0 \;\Rightarrow\; a = \bar{y} - b\bar{x}.$

From $\frac{\partial S}{\partial b} = 0$:
$-2\sum x_i[y_i - a - bx_i] = 0 \;\Rightarrow\; \sum x_i y_i - a\sum x_i - b\sum x_i^2 = 0.$

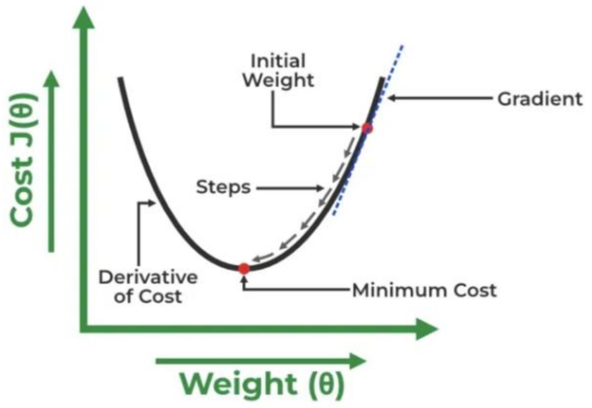
Substitute $a = \bar{y} - b\bar{x}$ and $\sum x_i = n\bar{x}$:
$\sum x_i y_i - n\bar{x}\bar{y} - b[\sum x_i^2 - n\bar{x}^2] = 0.$

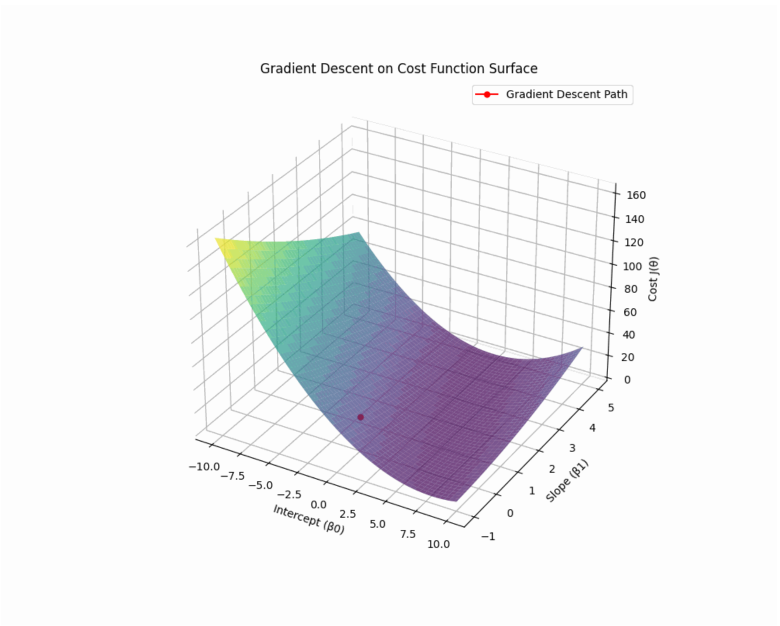
Note $\sum (x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n\bar{x}\bar{y}$ and $\sum (x_i - \bar{x})^2 = \sum x_i^2 - n\bar{x}^2$, so

$$b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}.$$

Solving Regression using the Gradient Descent Method

- A linear regression model can be trained using **gradient descent** method, which adjusts the model's parameters to minimize the mean squared error (MSE).
- To update the Intercept (Beta 0) and the Slope (Beta 1) and reduce the cost function (minimizing the RMSE).
- Gradient descent starts with random values for the Intercept and the Slope and iteratively improves them to find the best-fit line.
- A gradient is simply the derivative, showing how small changes in inputs affect the output.
- By moving in the direction of the Mean Squared Error negative gradient with respect to the coefficients, the coefficients can be changed.





Model Evaluation Metrics

1. Total Sum of Squares (SST)
Measures the total variance in the actual data.
Formula: $\mathbf{SST = \Sigma(y_i - \bar{y})^2}$
Interpretation: How much the actual values vary from their mean.

2. Sum of Squares for Error (SSE)
Measures the unexplained variance (residuals).
Formula: $\mathbf{SSE = \Sigma(y_i - \hat{y}_i)^2}$
Interpretation: How far the predictions are from the actual values.

3. Sum of Squares for Regression (SSR)
Measures the variance explained by the regression model.
Formula: $\mathbf{SSR = \Sigma(\hat{y}_i - \bar{y})^2}$

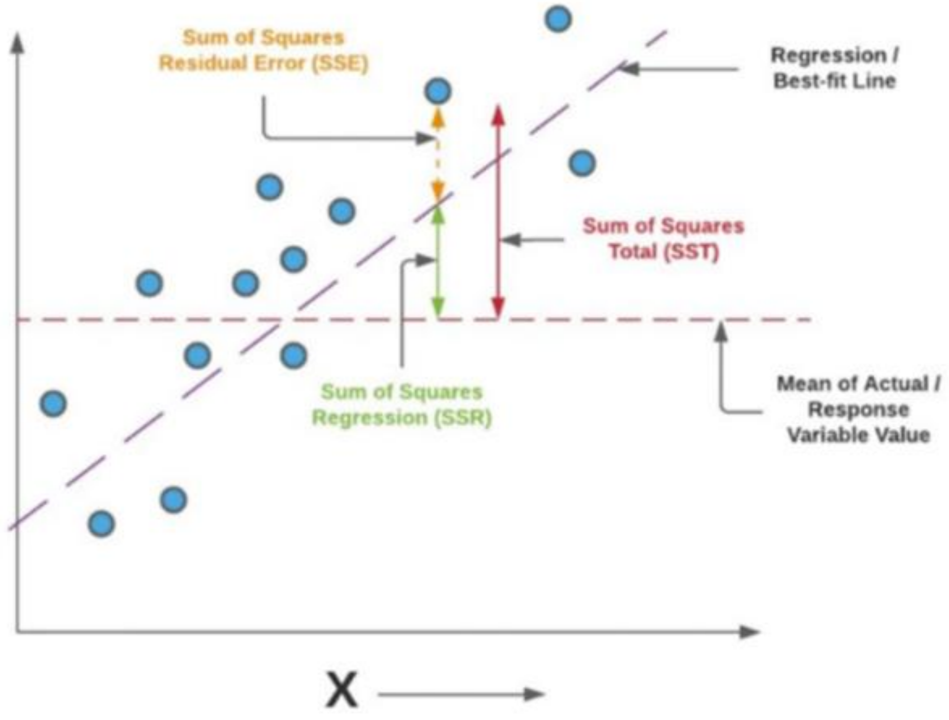
Interpretation: How much of the variation is captured by the model.

Relationship Between the Three
SST = SSR + SSE

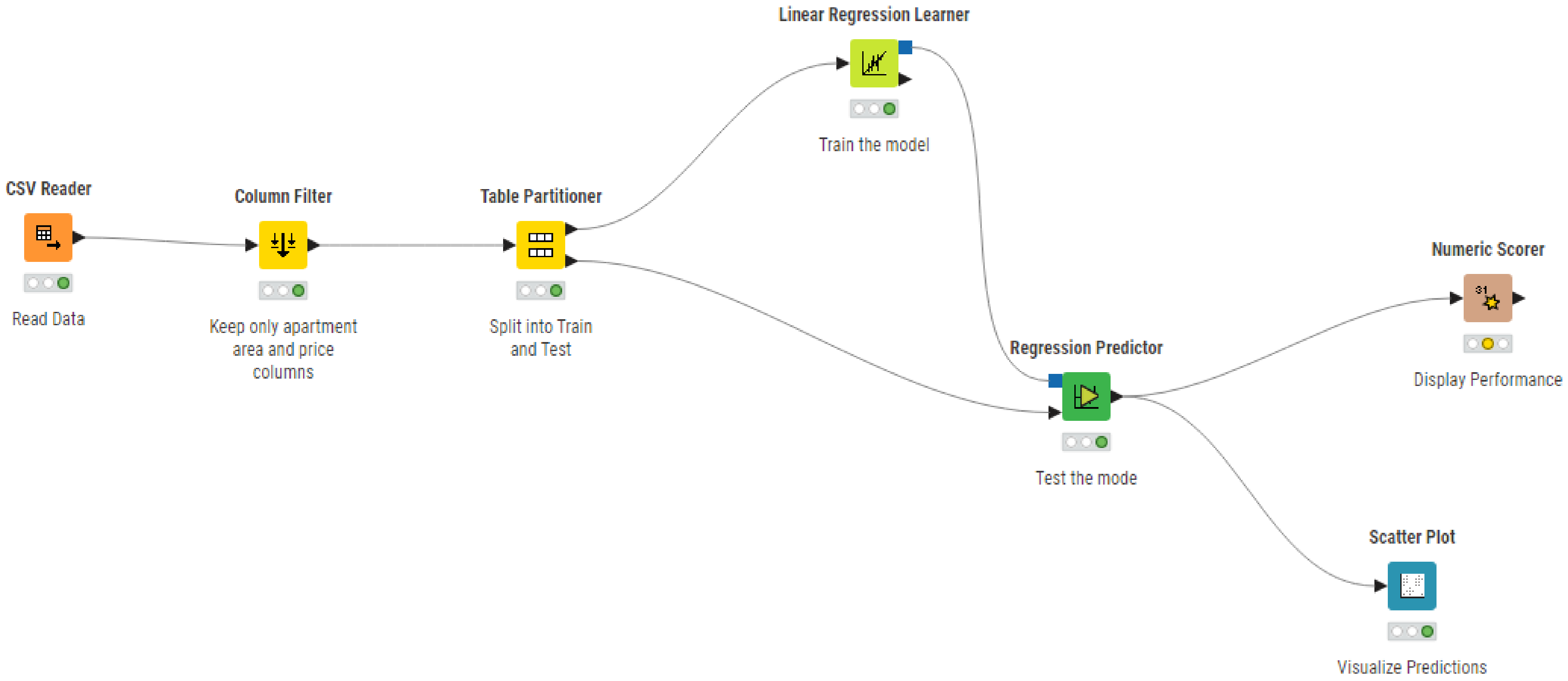
4. R-squared (R^2)
Proportion of total variance explained by the model.
Formula: $\mathbf{R^2 = 1 - (SSE / SST)}$
Range: 0 to 1. Higher R^2 indicates better fit.

5. Mean Squared Error (MSE)
Average of the squared residuals.
Formula: **MSE = SSE / n**
Used to assess the model's prediction error.

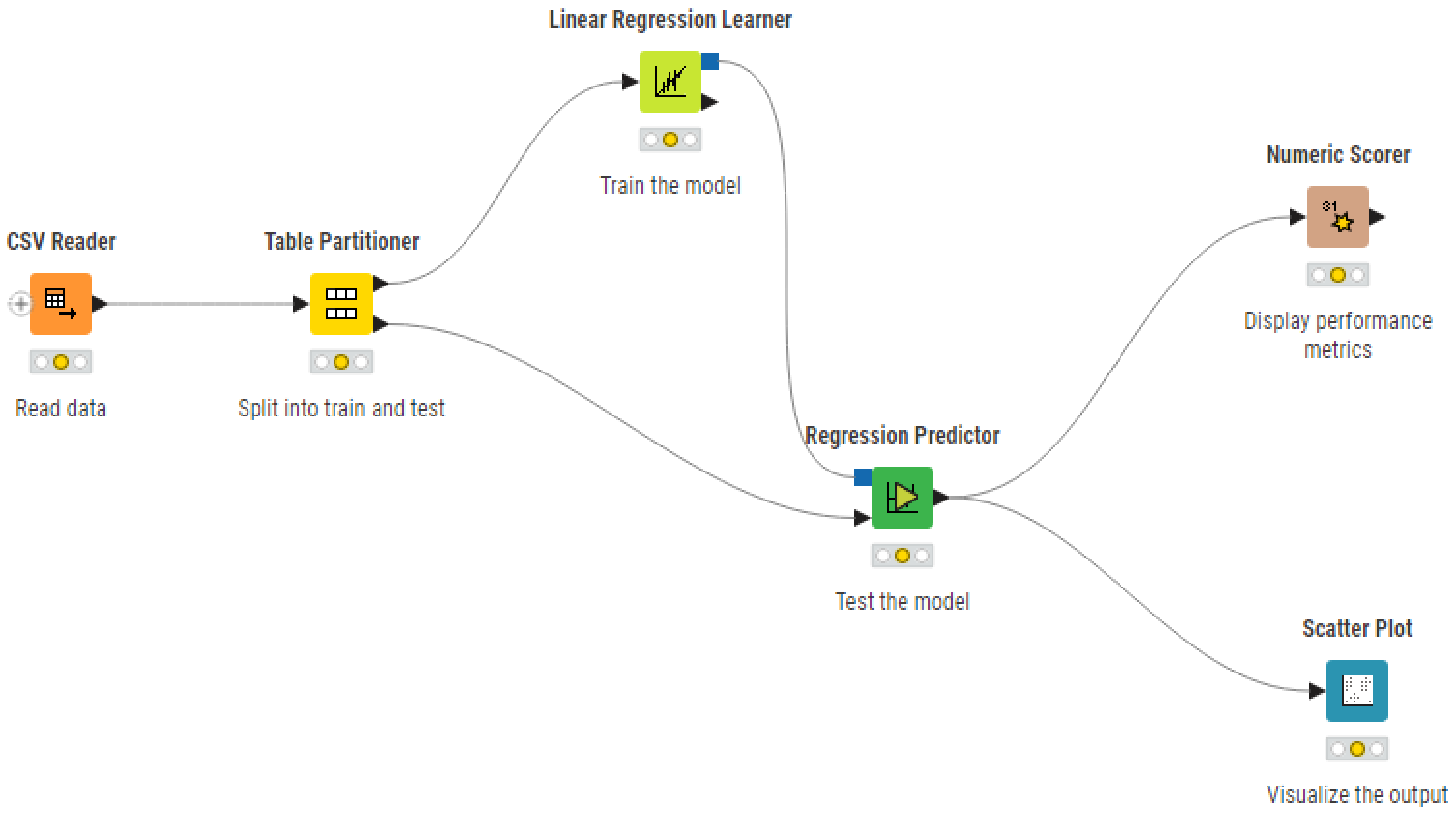
6. Root Mean Squared Error (RMSE)
Square root of MSE.
Formula: $\mathbf{RMSE = \sqrt{MSE}}$



Implementation in KNIME – Simple Linear Regression



Implementation in KNIME – Multiple Linear Regression



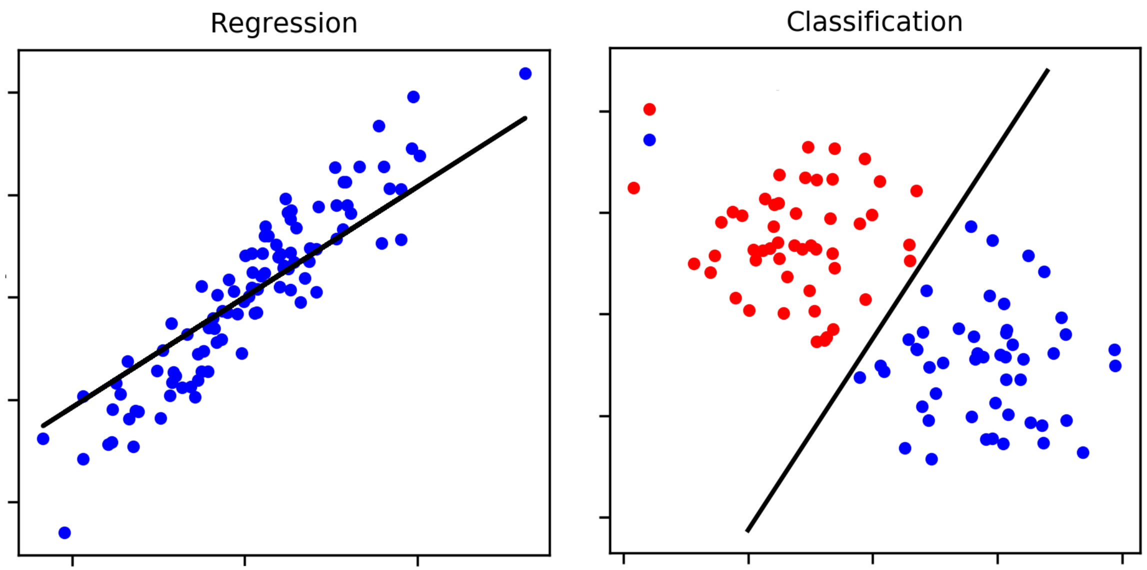
Classification

What is Classification?

Classification is a type of supervised learning where the goal is to predict a **categorical label** (like "yes" or "no") instead of a continuous value.

Examples:

- Predicting if an email is spam or not
- Determining if a customer will make a purchase
- Medical diagnosis



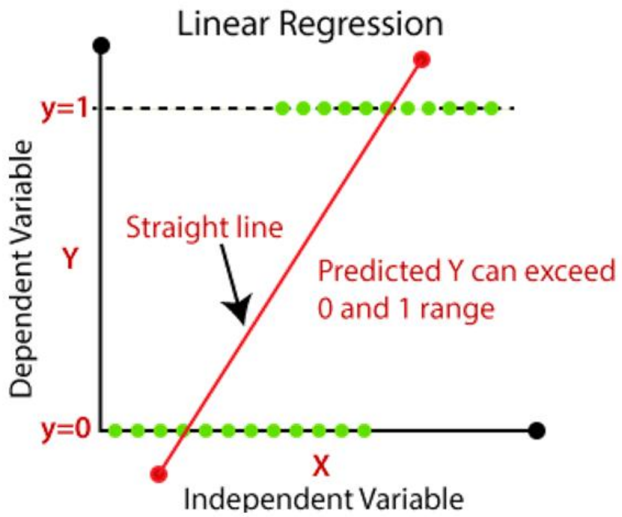
Binary Classification Examples

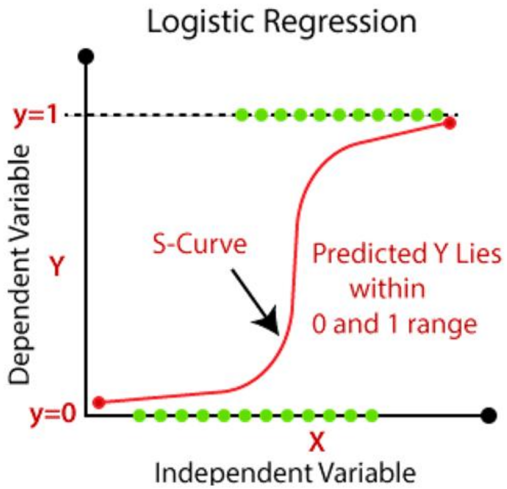
Problem	Example Features
Churn Prediction	Customer tenure, monthly charges, contract type, support calls
Spam Detection	Email subject length, sender reputation, word frequency
Loan Approval	Income, credit score, loan amount, employment status
Fraud Detection	Transaction amount, location, card usage frequency
Disease Diagnosis	Age, symptoms, test results, exposure history

Why Not Linear Regression?

Linear Regression Problems:

- Great for predicting continuous numbers (like prices)
- **Struggles with binary outcomes** (yes/no, spam/not spam)
- Can predict values outside 0-1 range
- Doesn't handle categorical data well
- **Solution:** Logistic Regression





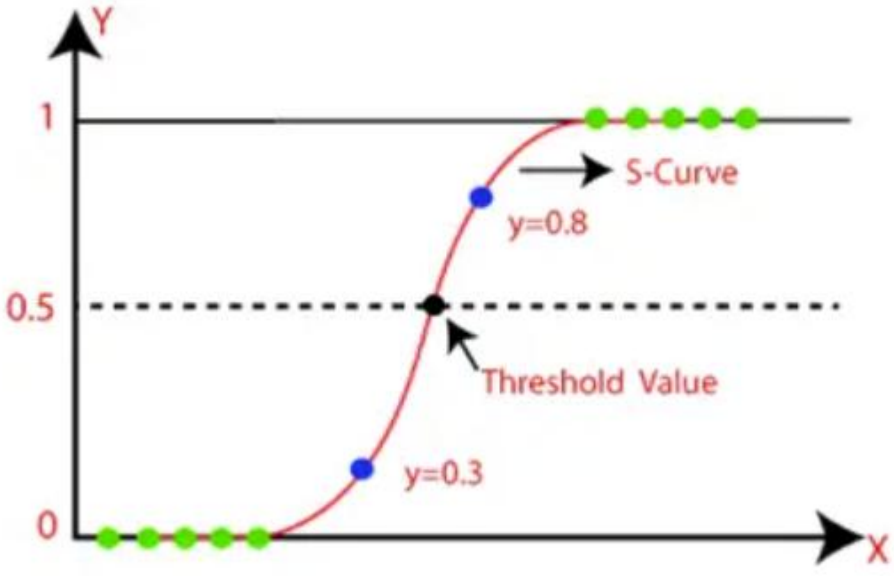
What is Logistic Regression?

- **Logistic regression** predicts the **probability** that an instance belongs to a certain class.
- **Key Component: Sigmoid Function**

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

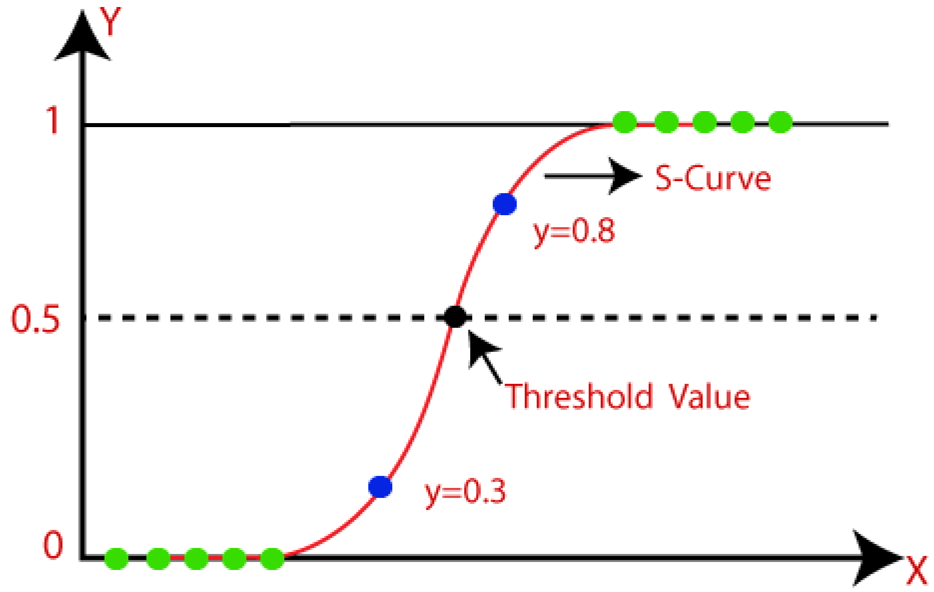
How it works:

- Takes any input value
- Squeezes it between 0 and 1
- Perfect for probability predictions!



Decision Making

- **Classification Rules:**
- If probability > 0.5 → Predict "Yes" (Class 1)
- If probability ≤ 0.5 → Predict "No" (Class 0)
- **Visual Comparison:**
- Linear Regression: Straight line, can go beyond 0-1
- Logistic Regression: S-curve, bounded between 0-1



Solving Logistic Regression

We solve logistic regression using **numerical optimization techniques**:

A. Gradient Descent (or variants like SGD, mini-batch GD)

We minimize the **negative log-likelihood** (i.e., cross-entropy loss):

$$\text{Loss}(\beta) = -\sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

Gradient of the loss w.r.t. β is:

$$\nabla_\beta = X^T(\hat{y} - y)$$

Then you update the parameters using:

$$\beta \leftarrow \beta - \eta \cdot \nabla_\beta$$

Confusion Matrix

Use these four statistics to calculate other evaluation metrics, such as overall accuracy, true positive rate, and false positive rate

	Predicted class positive	Predicted class negative
True class positive	TRUE POSITIVE	FALSE NEGATIVE
True class negative	FALSE POSITIVE	TRUE NEGATIVE

- TRUE POSITIVE (TP): Actual and predicted class is positive
- TRUE NEGATIVE (TN): Actual and predicted class is negative
- FALSE NEGATIVE (FN): Actual class is positive and predicted negative
- FALSE POSITIVE (FP): Actual class is negative and predicted positive

Overall Accuracy

- Definition:

$$\boldsymbol{Overall\ accuracy} = \frac{\boldsymbol{\#\ Correct\ classifications\ (test\ set)}}{\boldsymbol{\#\ All\ events\ (test\ set)}}$$

- The proportion of correct classifications
- Downsides:
  - Only considers the performance in general and not for the different classes
  - Therefore, not informative when the class distribution is unbalanced

Precision and Recall

Precision

- **Definition:** Of all items predicted as positive, the proportion that are actually positive.
- **Formula:**

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

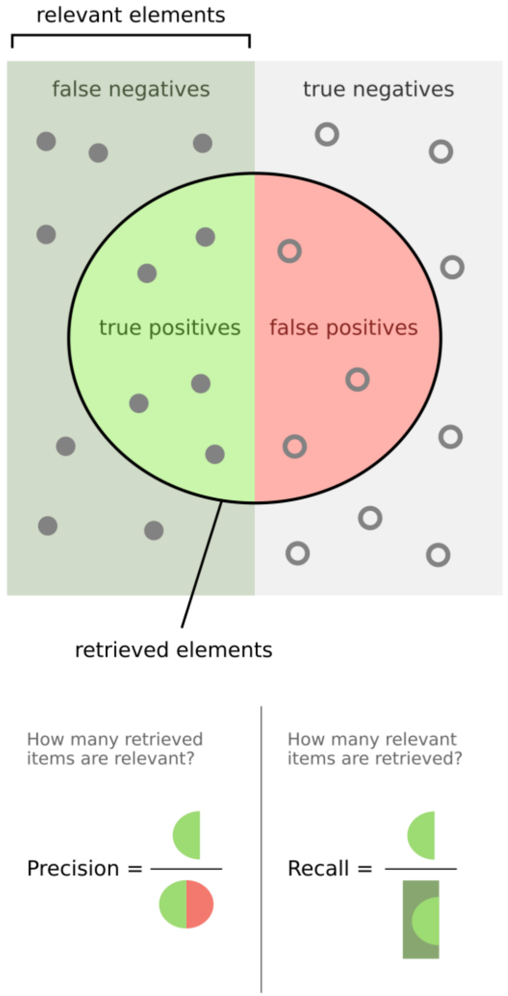
- **When it matters more:**
  - **Email Spam Detection** – avoid marking legitimate emails as spam.
  - **Final Stage Player Buying** – avoid signing players who aren't truly top talent.

Recall

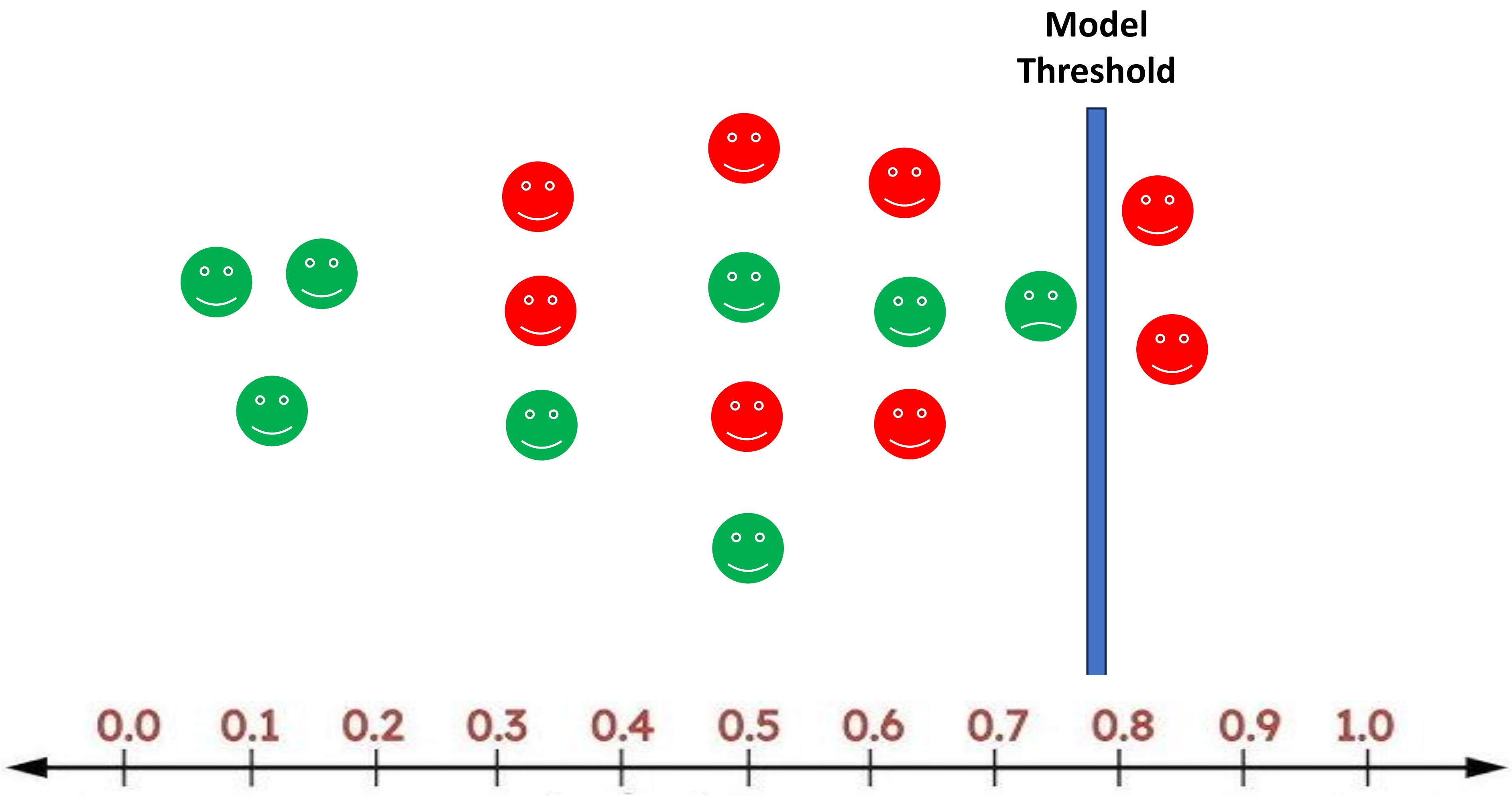
- **Definition:** Of all items that are actually positive, the proportion correctly identified as positive.
- **Formula:**

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

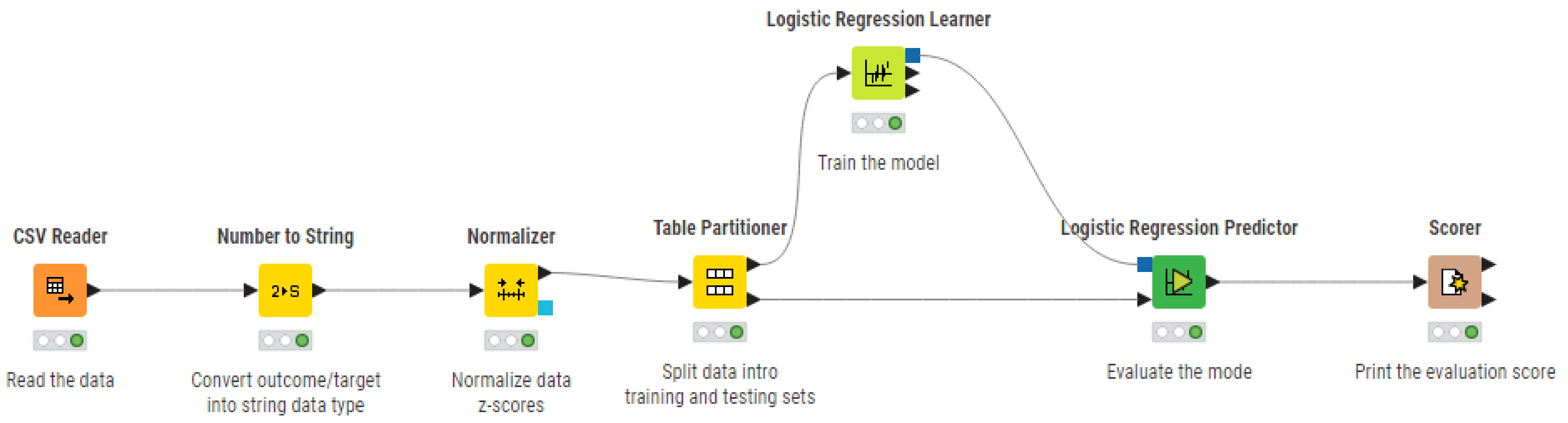
- **When it matters more:**
  - **Medical Screening** – avoid missing sick patients.
  - **Early Stage Player Scouting** – avoid missing potential stars, even if the list has some weaker players.



Precision and Recall



Implementation in KNIME – Logistic Regression

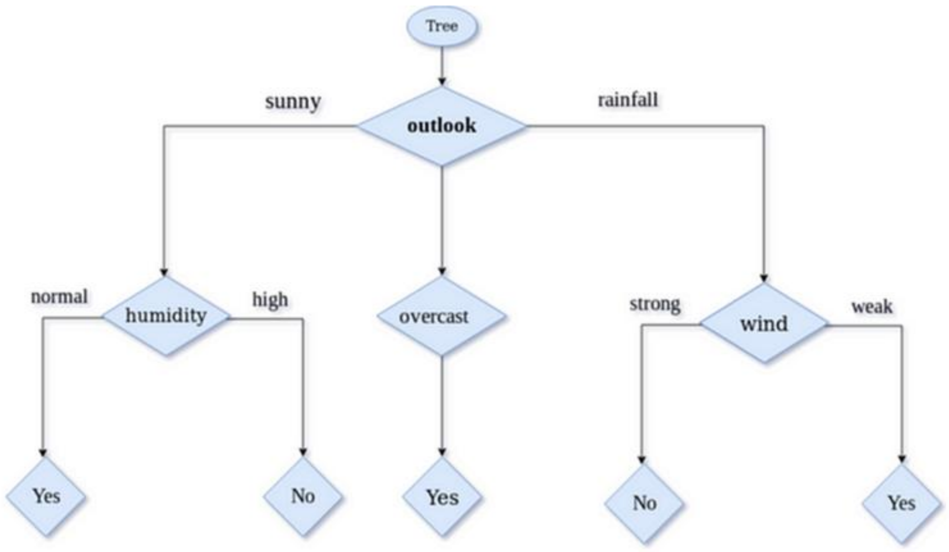


Decision Trees

- A **decision tree** is a popular supervised learning algorithm used for both classification and regression tasks.
- In classification, the goal is to predict the class label of an observation by splitting the data based on feature values in a tree-like structure.
- For example, to decision if we can paly tennis or not on a give day, we can use weather information to build a classification model to assist us in making the decision.

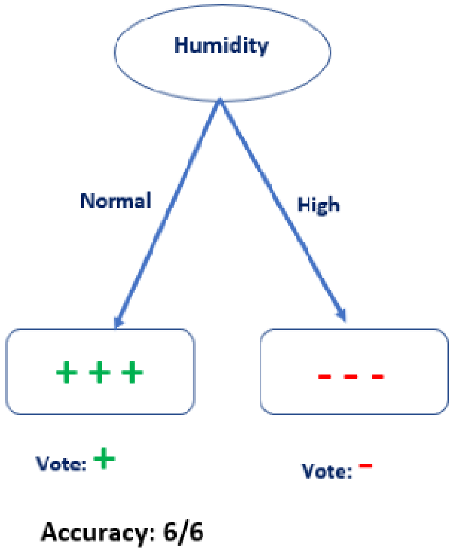
Play Tennis Dataset

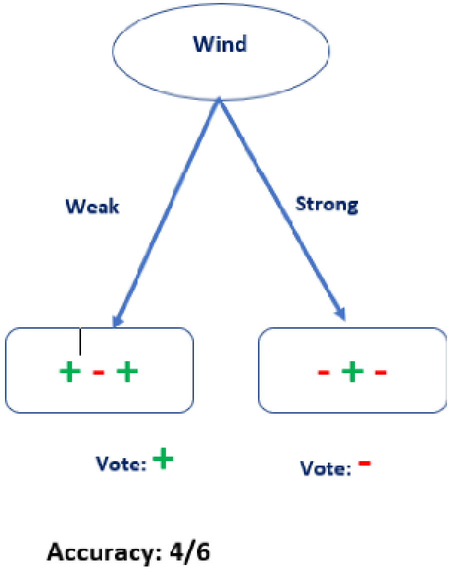
Day	outlook	temperature	humidity	wind	Decision
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rainfall	mild	high	weak	Yes
5	rainfall	cool	normal	weak	Yes
6	rainfall	cool	normal	strong	No
7	overcast	cool	normal	wtrong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rainfall	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	weak	Yes
14	rainfall	mild	high	strong	No



How can we split, which factor can we use?

Humidity	Wind	Decision
Normal	Weak	Yes
High	Weak	No
Normal	Strong	Yes
High	Strong	No
High	Strong	No
Normal	Weak	Yes





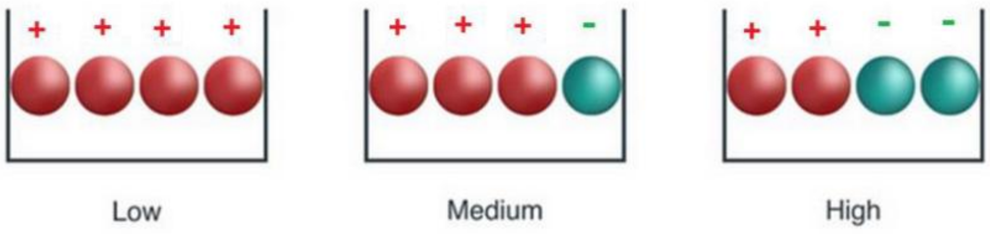
The Gini Index Method

The Gini index is a measure of inequality in sample. It has a value between 0 and 1.

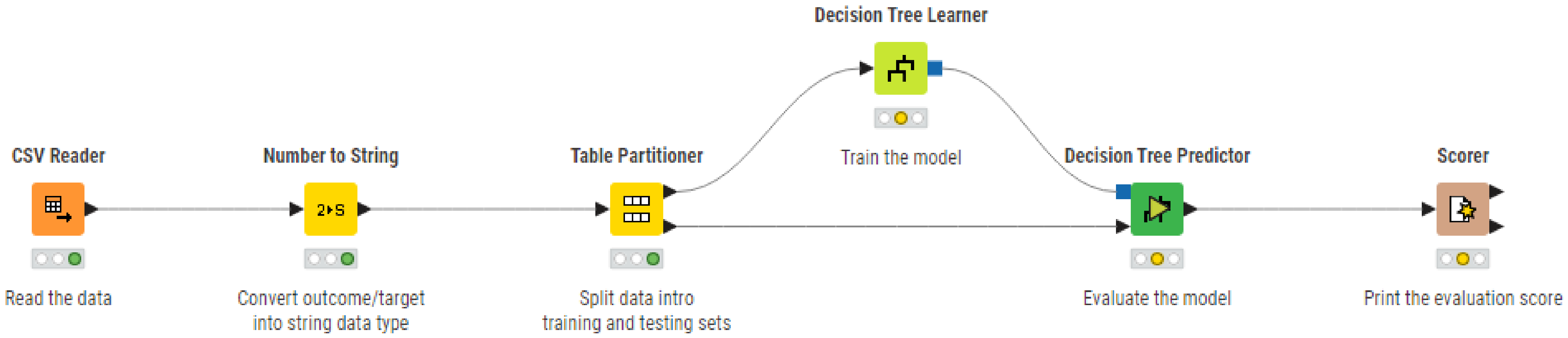
$$Gini\,index = 1 - \sum_{i=1}^{n} p_i^2$$

The Gini Index can be used to evaluate the split impurity when constructing classification trees.

Gini index of value 0 means sample is perfectly homogeneous, and all elements are similar, whereas Gini index of value 1 means maximal inequality among elements.



Implementation in KNIME – Decision Trees



Unsupervised Learning

Clustering

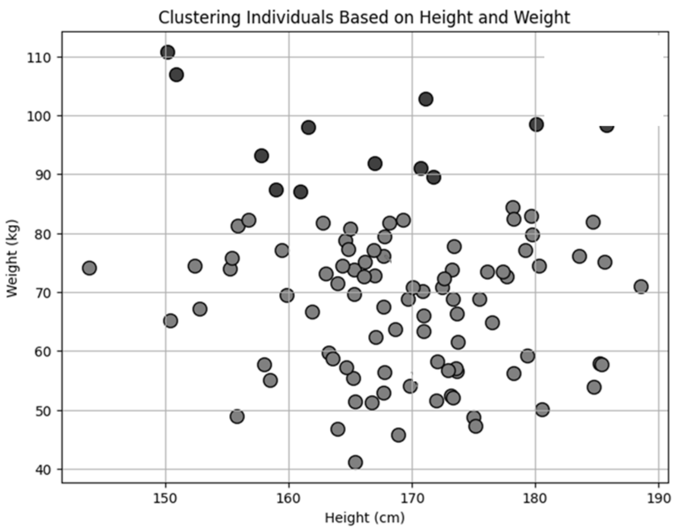
What Is Unsupervised Learning?

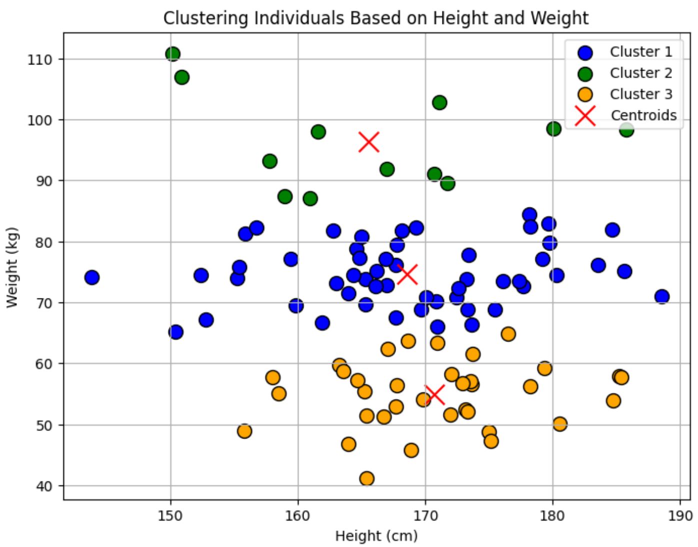
- **Definition**: Learning from data without labeled outcomes.
- **Goal**: Discover hidden patterns or structures.
- **Contrast with Supervised Learning**:
  - Supervised: Has labels (e.g., spam / not spam)
  - Unsupervised: No labels (e.g., group customers)

Clustering: Overview and Algorithms

What is Clustering?

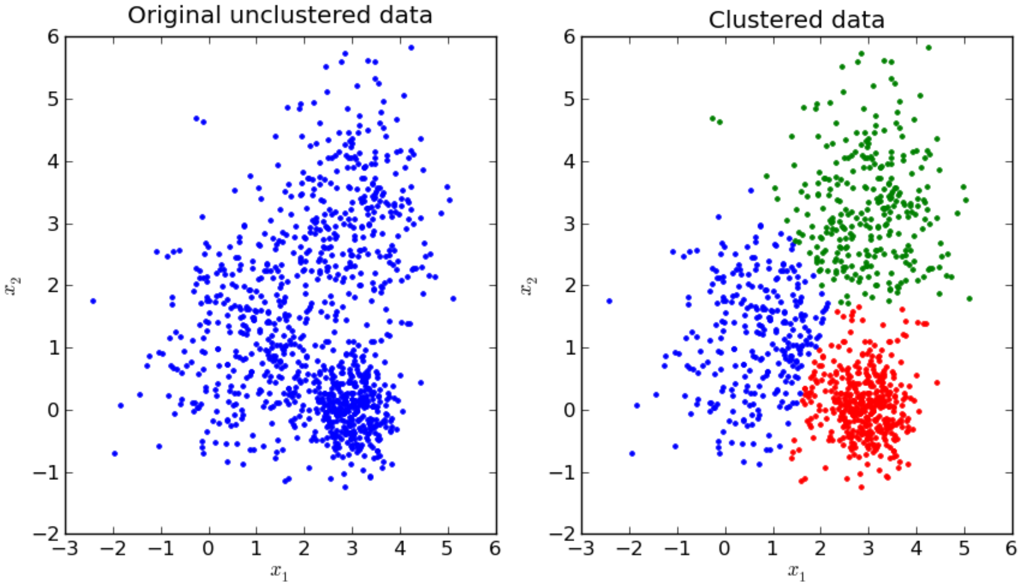
- Clustering is the process of finding groups (clusters) of similar data points in a dataset.
- It applies to **unlabeled** data, meaning no prior category or label is given.
- Clustering is a form of **unsupervised learning**.
- Goal: Segment data into groups where:
  - Points in the same cluster are **similar**
  - Points in different clusters are **dissimilar**

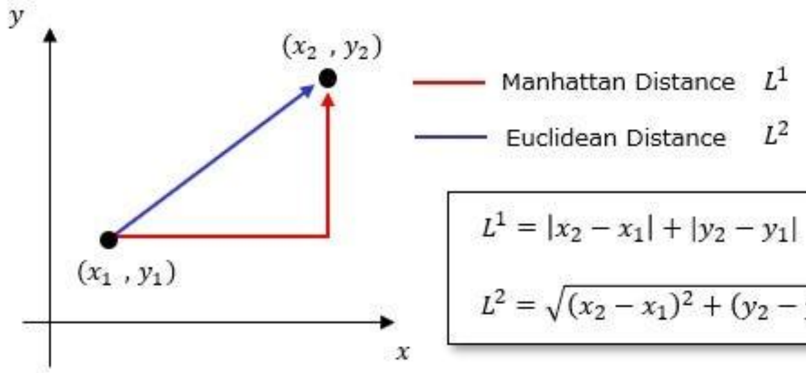




Clustering: Overview and Algorithms

- In a scatter plot, data points often concentrate into visually distinct groups.
- These groupings are what clustering algorithms aim to discover.
- Even with many features (high-dimensional space), similarity can be defined mathematically (e.g., Euclidean distance).





$$L^1 = |x_2 - x_1| + |y_2 - y_1|$$

$$L^2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

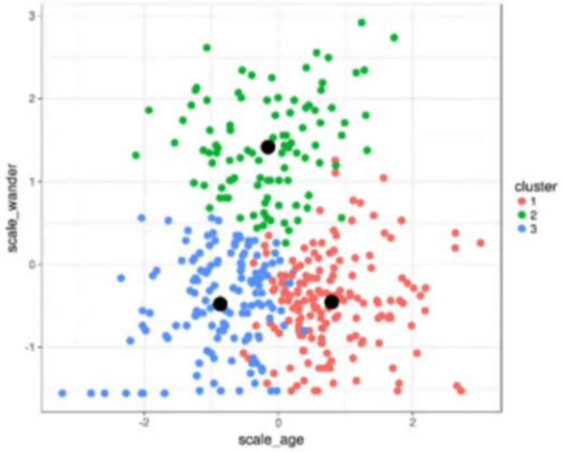
Properties of Clusters

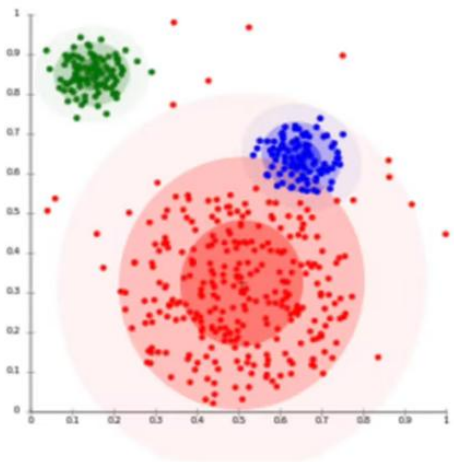
Clusters can differ in:

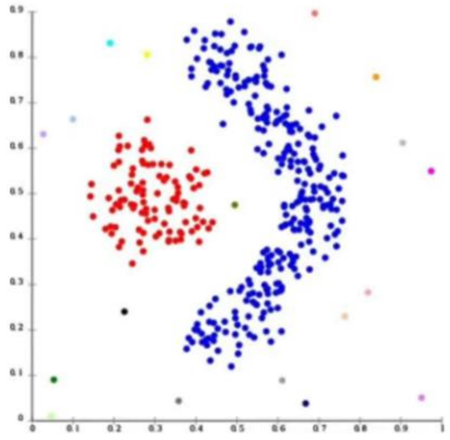
- **Shape** round, elongated, irregular
- **Size** small or large
- **Density**: tight or sparse

Clusters may:

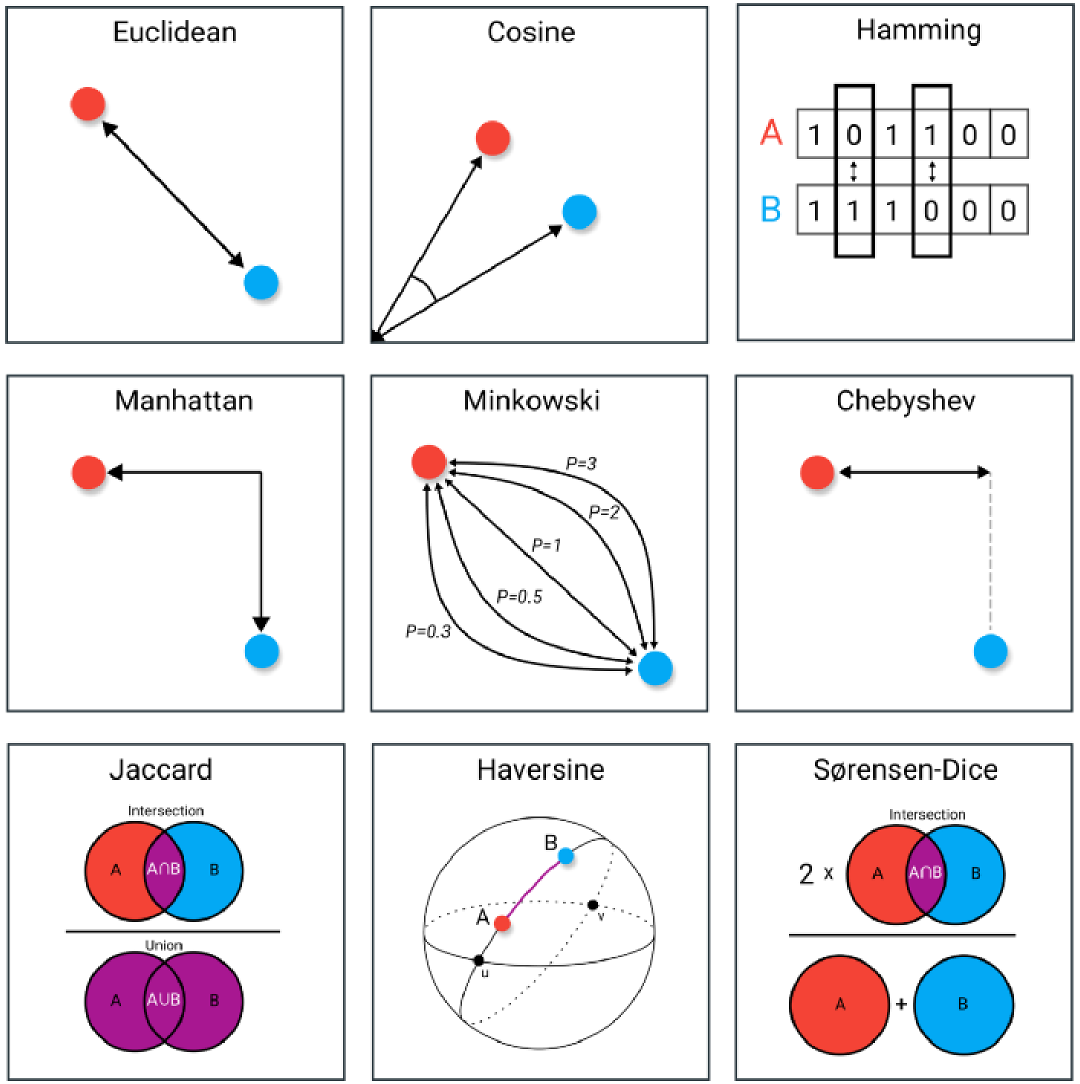
- Be **disjoint**, **touch**, or even **overlap**
- Be **flat** (partition-based) or form a **hierarchy** (tree-like structure)







Distance Measures



From Business to Vectors: Making Clustering Real

- **Business Data is Numeric at its Core**
- While business problems feel "real-world," the data behind them is **ultimately numeric**.
- Each data point in clustering is just a **vector of feature values**.
- These features are often business-relevant attributes.
- **Example: Customer Segmentation**
  Each customer can be represented as a point in multidimensional space:

Feature	Description
Age	Numeric (e.g., 34)
Annual Income	Numeric (e.g., $55,000)
Spending Score	Numeric (behavioral score)
Number of Purchases	Numeric (e.g., 18 purchases)
Loyalty Points	Numeric (e.g., 1200 points)

- This becomes a vector like: **[34, 55000, 78, 18, 1200]**
- Algorithms don't care about *what* the numbers mean—only how close or far apart they are.
- This allows us to **abstract real-world entities** (like customers or products) into a mathematical space where clustering makes sense.
- As long as we choose meaningful features, we can uncover **real, actionable business insights** through clustering.

Use Cases of Clustering

Business Case	Purpose of Clustering	Example Features
Customer Segmentation in Retail	Group customers into distinct profiles for targeted marketing	Purchase frequency, average basket size, product categories bought, time since last purchase, store visits per month
Market Segmentation for Subscription Services	Identify different usage patterns to tailor pricing or packages	Login frequency, time spent per session, number of active days per month, feature usage counts
Product Recommendation Optimization	Group similar products for cross-selling	Price range, category, material, seasonality, purchase co-occurrence
Fraud Pattern Detection in Banking	Identify unusual account clusters that might indicate fraud	Transaction amount distribution, transaction frequency, merchant type diversity, average geographic distance between transactions
Insurance Risk Profiling	Classify policyholders into risk groups	Age, claim frequency, claim amount, type of coverage, premium paid
Healthcare Patient Profiling	Identify patient types for preventive care programs	Age, BMI, blood pressure, cholesterol levels, visit frequency, medical conditions
Supply Chain Optimization	Group suppliers or logistics routes by performance or cost	Delivery time, delivery reliability, cost per unit, geographic location, order quantity
Churn Risk Grouping	Detect groups with higher likelihood of leaving	Subscription length, last interaction date, support tickets opened, payment delays, engagement score
Store Location Analysis	Group stores with similar sales/traffic patterns	Daily foot traffic, sales per square meter, region demographics, average transaction value
Social Media Community Detection	Group similar influencers or audiences	Follower count, engagement rate, content topics, posting frequency

K-Means Clustering

K-Means Clustering is especially useful when you want to quickly partition data into a fixed number of roughly spherical clusters, making it fast and effective for well-separated groups.

Type: Partition-based clustering

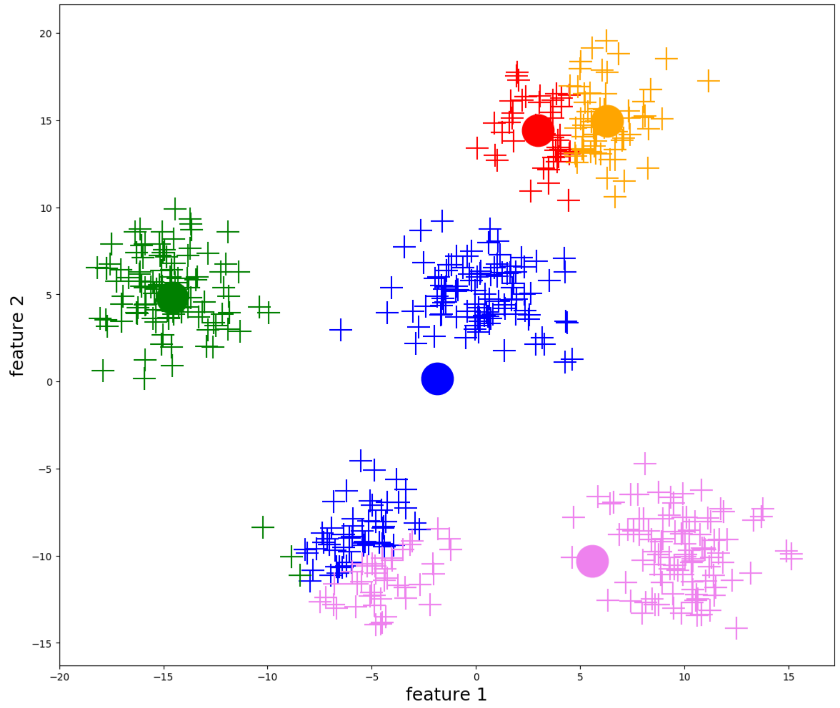
Objective: Divide data into **k** clusters

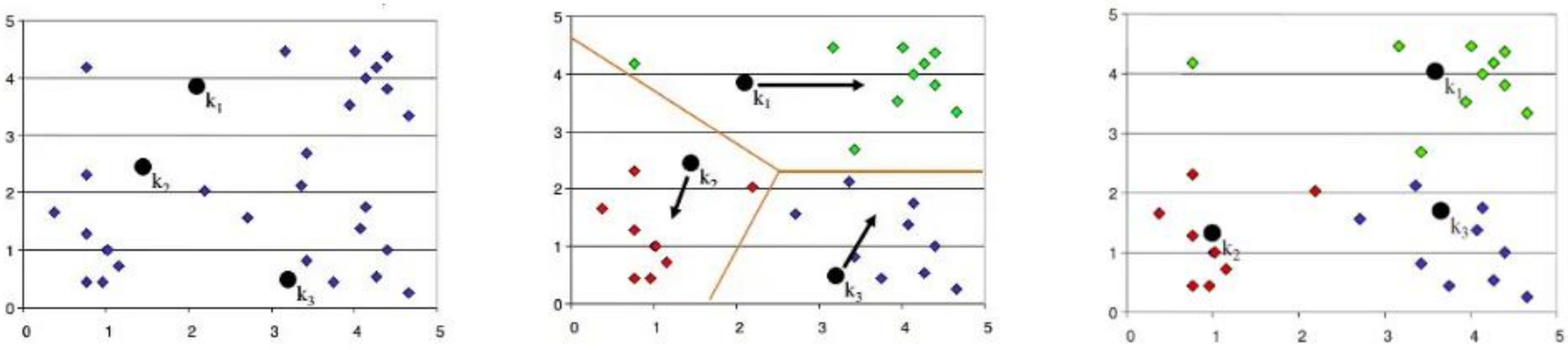
Procedure:

1. Randomly select **k** initial cluster centers (centroids)
2. Assign each data point to the nearest center
3. Recalculate each center as the mean of its assigned points
4. Repeat steps 2–3 until cluster centers stabilize (no longer move)

Characteristics:

- Results depend on the initial choice of centroids
- Sensitive to outliers
- Requires that we predefine the number of clusters





Practical Example – Iteration 1

Problem Setup

- **Data Points:** (x, y)
  - A1: (2, 10)
  - A2: (2, 5)
  - A3: (8, 4)
  - A4: (5, 8)
  - A5: (7, 5)
  - A6: (6, 4)
  - A7: (1, 2)
  - A8: (4, 9)
- **Initial Centroids:**
  - Mean 1: (2, 10)
  - Mean 2: (5, 8)
  - Mean 3: (1, 2)

Steps:

1. **Compute Distances**: Use the Euclidean distance formula to calculate the distance from each point to all centroids.
2. **Assign Clusters**: Each point is assigned to the cluster of the nearest centroid.

Point	Coordinates	Dist Mean 1 ($\sqrt{(2-x)^2 + (10-y)^2}$)	Dist Mean 2 ($\sqrt{(5-x)^2 + (8-y)^2}$)	Dist Mean 3 ($\sqrt{(1-x)^2 + (2-y)^2}$)	Cluster
A1	(2, 10)	$\sqrt{(2-2)^2 + (10-10)^2} = 0.00$	$\sqrt{(5-2)^2 + (8-10)^2} = 3.61$	$\sqrt{(1-2)^2 + (2-10)^2} = 8.25$	1
A2	(2, 5)	$\sqrt{(2-2)^2 + (10-5)^2} = 5.00$	$\sqrt{(5-2)^2 + (8-5)^2} = 3.61$	$\sqrt{(1-2)^2 + (2-5)^2} = 3.16$	3
A3	(8, 4)	$\sqrt{(2-8)^2 + (10-4)^2} = 9.43$	$\sqrt{(5-8)^2 + (8-4)^2} = 5.00$	$\sqrt{(1-8)^2 + (2-4)^2} = 7.28$	2
A4	(5, 8)	$\sqrt{(2-5)^2 + (10-8)^2} = 3.61$	$\sqrt{(5-5)^2 + (8-8)^2} = 0.00$	$\sqrt{(1-5)^2 + (2-8)^2} = 6.71$	2
A5	(7, 5)	$\sqrt{(2-7)^2 + (10-5)^2} = 8.06$	$\sqrt{(5-7)^2 + (8-5)^2} = 3.61$	$\sqrt{(1-7)^2 + (2-5)^2} = 6.08$	2
A6	(6, 4)	$\sqrt{(2-6)^2 + (10-4)^2} = 8.94$	$\sqrt{(5-6)^2 + (8-4)^2} = 4.47$	$\sqrt{(1-6)^2 + (2-4)^2} = 5.39$	2
A7	(1, 2)	$\sqrt{(2-1)^2 + (10-2)^2} = 8.00$	$\sqrt{(5-1)^2 + (8-2)^2} = 7.62$	$\sqrt{(1-1)^2 + (2-2)^2} = 1.00$	3
A8	(4, 9)	$\sqrt{(2-4)^2 + (10-9)^2} = 2.24$	$\sqrt{(5-4)^2 + (8-9)^2} = 1.41$	$\sqrt{(1-4)^2 + (2-9)^2} = 7.07$	2

Practical Example – Iteration 2

Recalculate Centroids: For each cluster, compute the new centroid as the mean of all points in that cluster.

For each cluster, the new centroid is calculated as the mean of the (x)- and (y)-coordinates of all points in the cluster:

$$\text{Centroid}_i = \left(\frac{\sum x_{\text{cluster}}}{\text{n}}, \frac{\sum y_{\text{cluster}}}{\text{n}}\right)$$

- **New Centroids**: $(2.0, 10.0)$, $(6.0, 6.0)$, $(1.5, 3.5)$

2. **Reassign Clusters**: Recompute distances to the updated centroids and assign points to the nearest cluster.

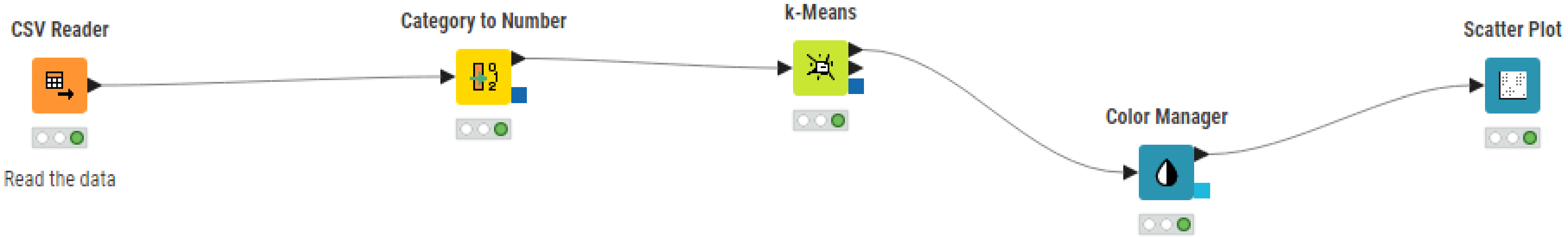
Point	Coordinates	Dist Mean 1 (√((x1−x)² + (y1−y)²))	Dist Mean 2 (√((x2−x)² + (y2−y)²))	Dist Mean 3 (√((x3−x)² + (y3−y)²))	Cluster
A1	(2, 10)	√((2−2)² + (10−10)²) = 0.00	√((6−2)² + (6−10)²) = 5.39	√((1.5−2)² + (3.5−10)²) = 8.83	1
A2	(2, 5)	√((2−2)² + (10−5)²) = 5.00	√((6−2)² + (6−5)²) = 4.47	√((1.5−2)² + (3.5−5)²) = 2.50	3
A3	(8, 4)	√((2−8)² + (10−4)²) = 9.22	√((6−8)² + (6−4)²) = 3.61	√((1.5−8)² + (3.5−4)²) = 6.86	2
A4	(5, 8)	√((2−5)² + (10−8)²) = 3.61	√((6−5)² + (6−8)²) = 2.24	√((1.5−5)² + (3.5−8)²) = 6.92	2
A5	(7, 5)	√((2−7)² + (10−5)²) = 8.06	√((6−7)² + (6−5)²) = 2.24	√((1.5−7)² + (3.5−5)²) = 5.92	2
A6	(6, 4)	√((2−6)² + (10−4)²) = 8.94	√((6−6)² + (6−4)²) = 2.83	√((1.5−6)² + (3.5−4)²) = 5.22	2
A7	(1, 2)	√((2−1)² + (10−2)²) = 8.06	√((6−1)² + (6−2)²) = 7.21	√((1.5−1)² + (3.5−2)²) = 1.80	3
A8	(4, 9)	√((2−4)² + (10−9)²) = 2.24	√((6−4)² + (6−9)²) = 3.61	√((1.5−4)² + (3.5−9)²) = 6.80	1

Comments:

- Notice how centroids shift toward the center of their clusters.
- Some points might change clusters as centroids update.

Implementation in KNIME – K-Means

Download and examine the customers segmentation workflow.



Hierarchical Clustering

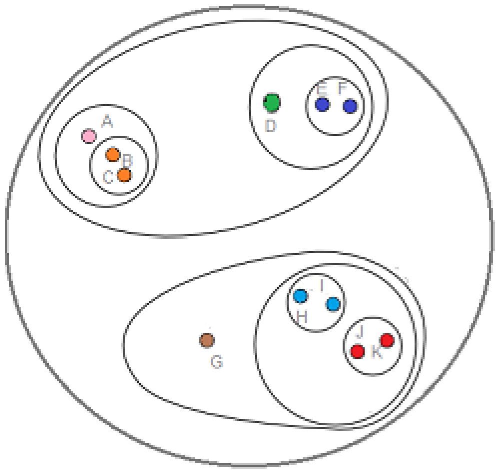
Hierarchical clustering is especially useful when you want more than just a flat set of clusters — you want to understand the relationships between clusters at different levels of granularity.

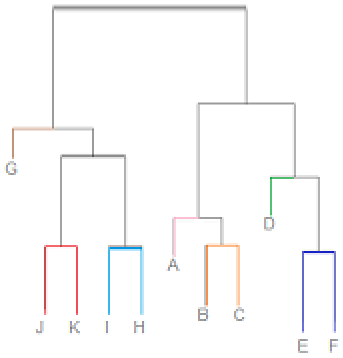
Type: Agglomerative (bottom-up) hierarchical clustering

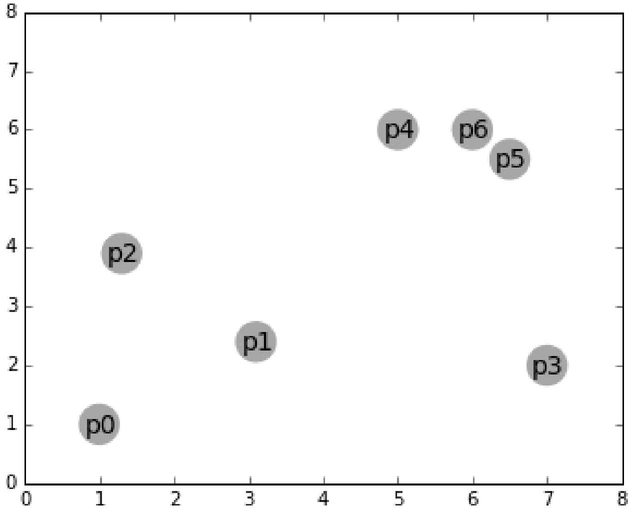
Procedure:

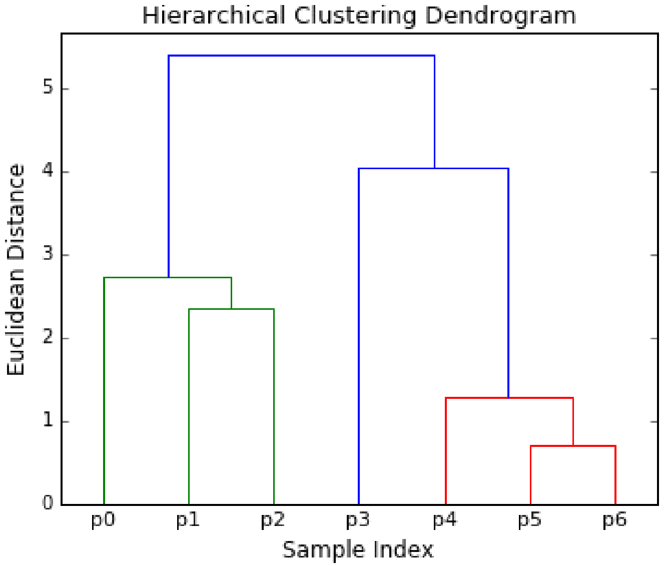
1. Start with each data point as its own cluster
2. Iteratively merge the two closest points/clusters
3. Continue until all data points are in one single cluster

- The result is a **dendrogram** (tree diagram showing cluster hierarchy)
- The y-axis of the dendrogram shows the DISTANCE between points.
- To extract a fixed number of clusters, apply a **cut-off threshold** on the dendrogram

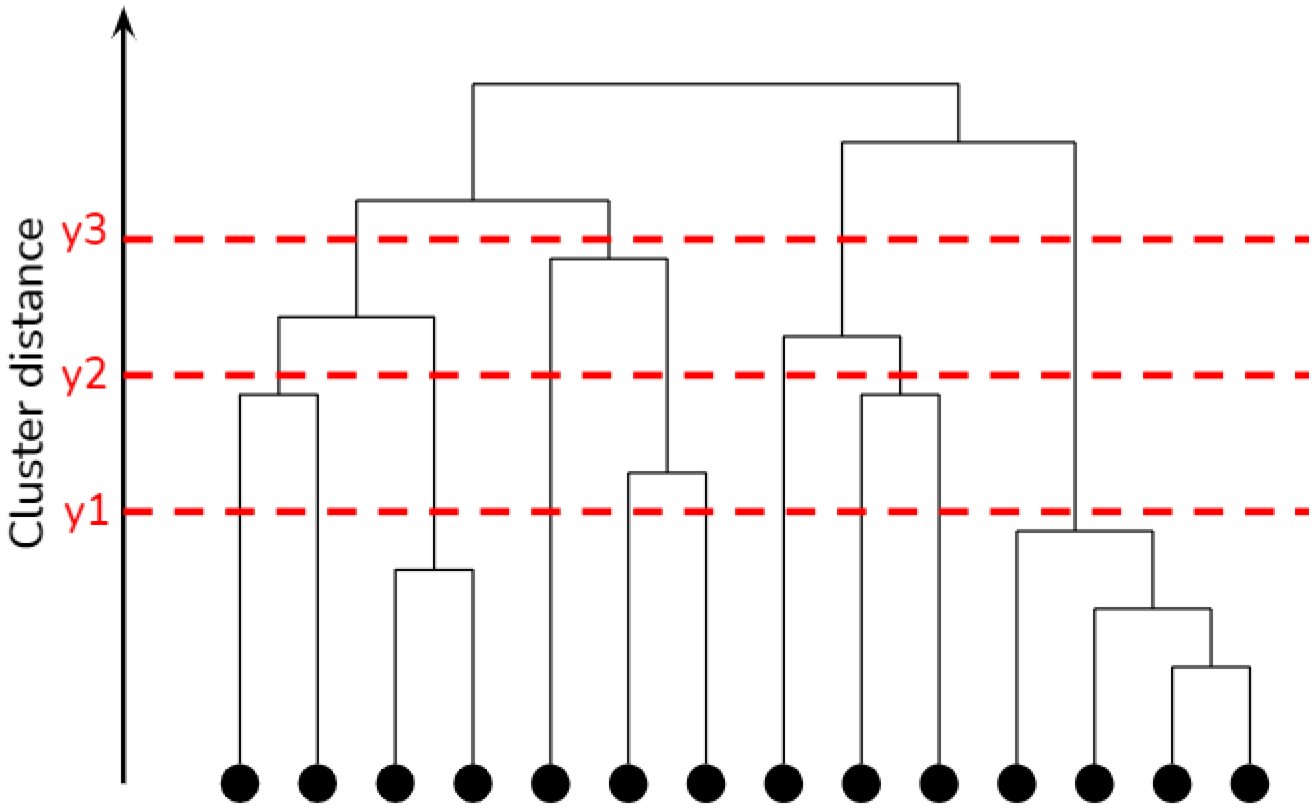






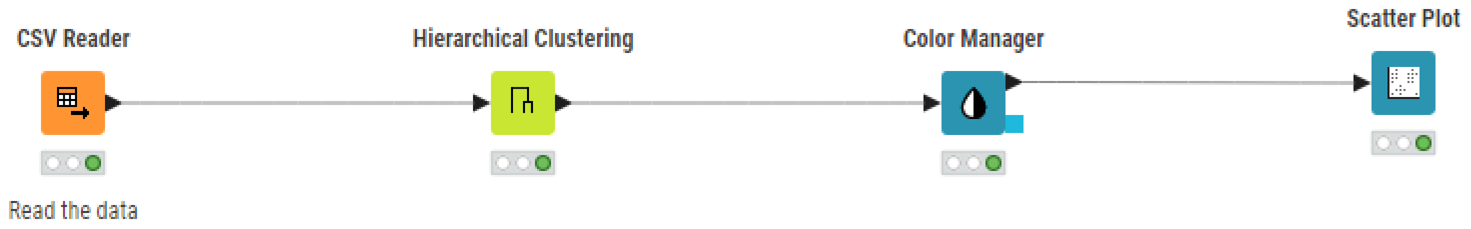


Distance Cut Points Result in Different Clusters



Implementation in KNIME - Hierarchal

Download and examine the customers segmentation workflow.



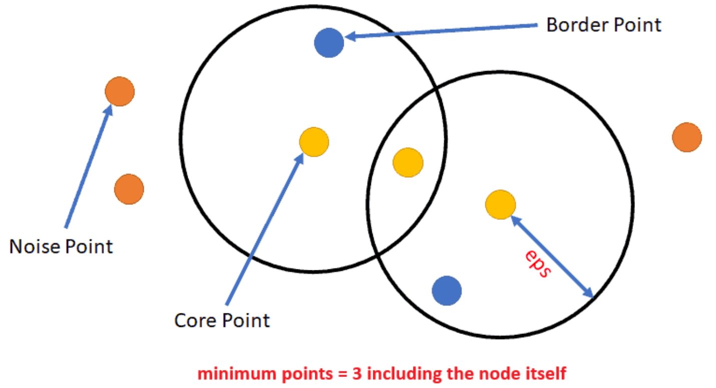
DBSCAN Clustering

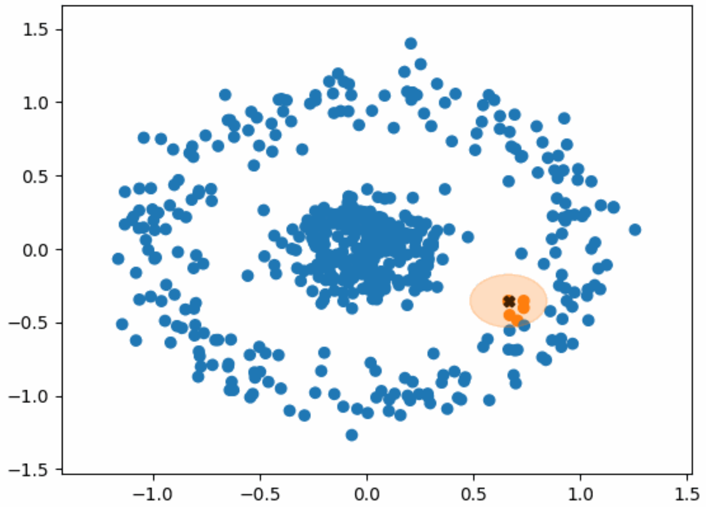
DBSCAN is especially useful when you want to discover clusters of arbitrary shape and identify noise points without having to predefine the number of clusters.

Type: Density-based clustering
Groups dense areas of data while treating sparse points as noise

Procedure:
1. Randomly pick a point
2. If enough neighboring points are found (within a defined radius), they form a cluster
3. Expand the cluster by checking neighbors of neighbors
4. Repeat for unvisited points

Characteristics:
- Can find clusters of arbitrary shape
- Automatically detects noise
- Does not require specifying the number of clusters beforehand
- “moons” datasets look artificial, but the idea they represent (non-convex, intertwined clusters) can appear in real business contexts.
- You just rarely see literal crescent shapes — it’s more about patterns that wrap around each other in feature space.





How does the Algorithm Work

Compute the distances between every two points

Points	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P1: (3,7)												
P2: (4,6)	1.41											
P3: (5,5)	2.83	1.41										
P4: (6,4)	4.24	2.83	1.41									
P5: (7,3)	5.66	4.24	2.83	1.41								
P6: (6,2)	5.83	4.47	3.16	2.24	1.00							
P7: (7,2)	6.40	5.00	3.61	2.24	1.00	1.00						
P8: (8,4)	5.83	4.47	3.16	2.24	1.41	2.24	1.41					
P9: (3,3)	5.00	4.24	3.16	3.16	4.12	3.00	3.16	4.47				
P10: (2,6)	1.41	2.00	3.16	4.24	5.39	5.83	6.40	6.32	3.00			
P11: (3,5)	2.00	1.41	2.00	3.16	4.24	4.47	5.00	4.47	2.24	1.00		
P12: (2,4)	3.16	2.83	3.16	4.00	5.10	5.39	5.83	5.10	2.00	2.00	1.41	0

Count the Number of Neighbors for Each Point

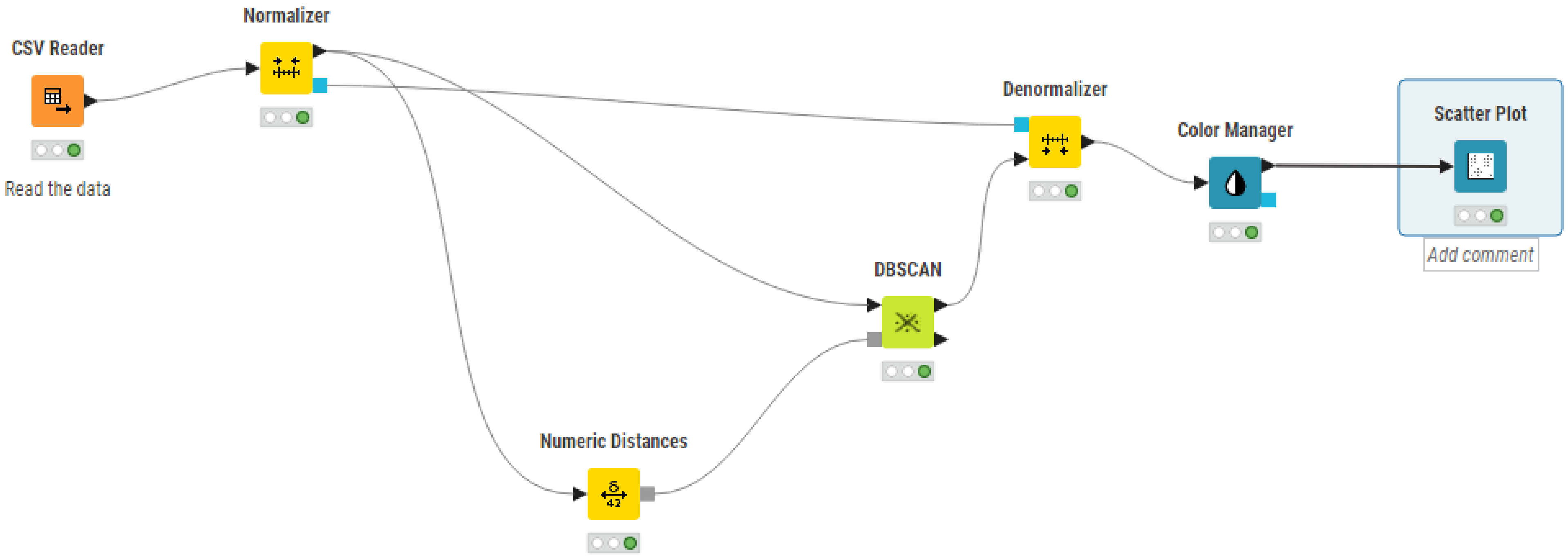
Point	Neighbors (ε=1.9)	# Neighbors
P1: (3,7)	P2, P10	2
P2: (4,6)	P1, P3, P11	3
P3: (5,5)	P2, P4	2
P4: (6,4)	P3, P5	2
P5: (7,3)	P4, P6, P7, P8	4
P6: (6,2)	P5, P7	2
P7: (7,2)	P5, P6	2
P8: (8,4)	P5	1
P9: (3,3)	-	0
P10: (2,6)	P1, P11	2
P11: (3,5)	P2, P10, P12	3
P12: (2,4)	P9, P11	2

Classify Points into Core, Border or Noise

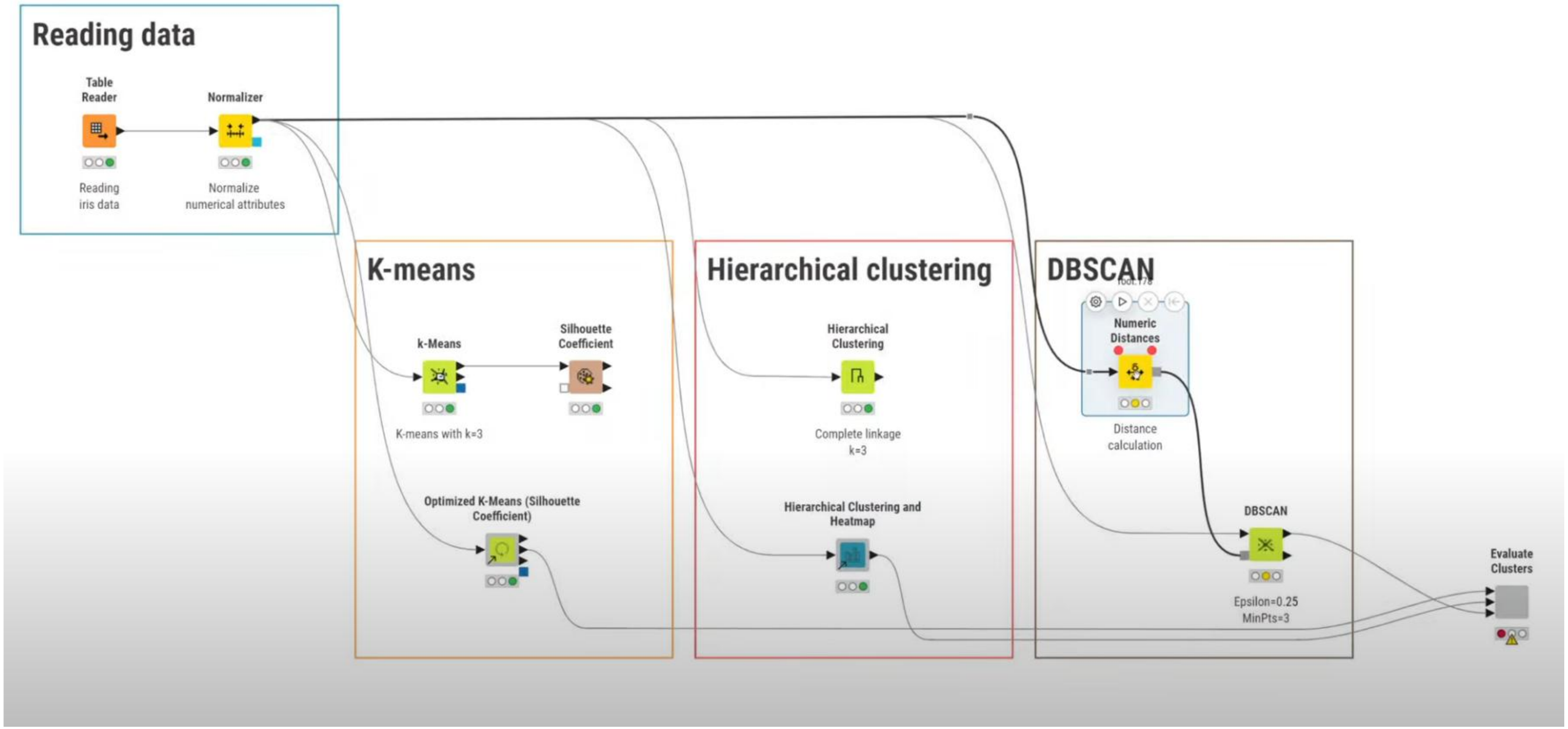
Point	Neighbors (ε=1.9)	Status
P1: (3,7)	P2, P10	Border
P2: (4,6)	P1, P3, P11	**Core**
P3: (5,5)	P2, P4	Border
P4: (6,4)	P3, P5	Border
P5: (7,3)	P4, P6, P7, P8	**Core**
P6: (6,2)	P5, P7	Border
P7: (7,2)	P5, P6	Border
P8: (8,4)	P5	Border
P9: (3,3)	-	Noise
P10: (2,6)	P1, P2, P11	Border
P11: (3,5)	P2, P10, P12	**Core**
P12: (2,4)	P9, P11	Border

Implementation in KNIME - DBSCAN

Download and examine the customers segmentation workflow.

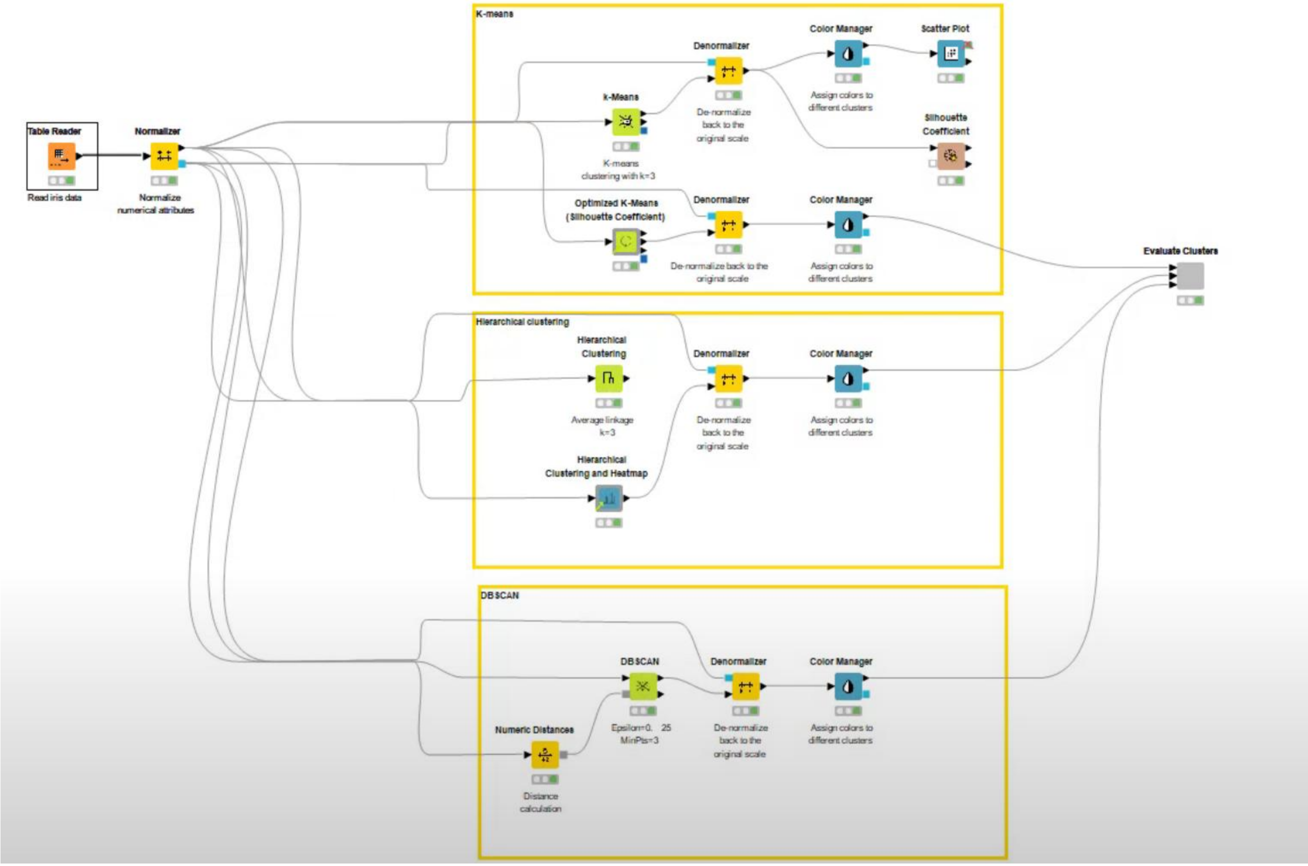


Clustering in KNIME



https://www.youtube.com/watch?v=i47dBwK8KfQ

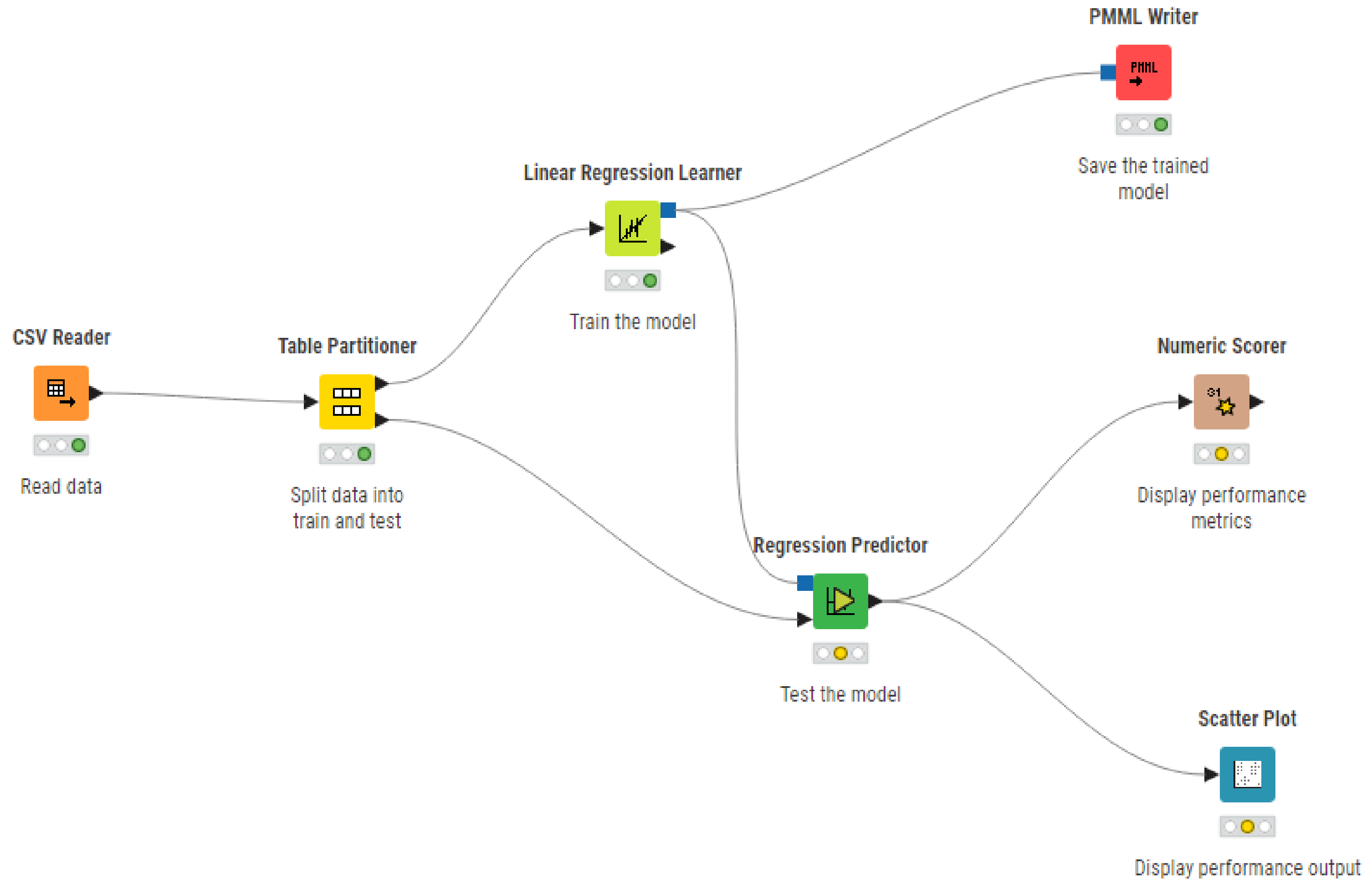
Clustering in KNIME



https://www.youtube.com/watch?v=7luMauX0KWM

Saving and Loading Trained Models

Save a Trained Model



Load and Use a Trained Model