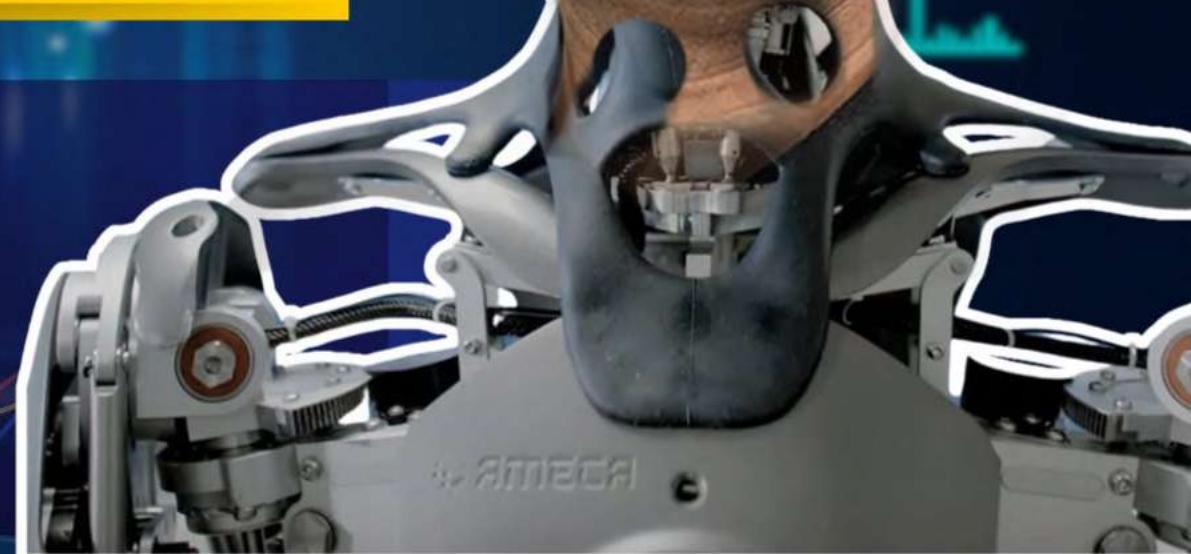


SEMESTER EXAM SERIES

MACHINE LEARNING

**IN
7 HOURS
+**
FREE NOTES



- **(UNIT-1 : INTRODUCTION)** Learning, Types of Learning, Well defined learning problems, Designing a Learning System, History of ML, Introduction of Machine Learning Approaches - (Artificial Neural Network, Clustering, Reinforcement Learning, Decision Tree Learning, Bayesian networks, Support Vector Machine, Genetic Algorithm), Issues in Machine Learning and Data Science Vs Machine Learning.
 - **(UNIT-2: REGRESSION & BAYESIAN LEARNING)** REGRESSION: Linear Regression and Logistic Regression. BAYESIAN LEARNING - Bayes theorem, Concept learning, Bayes Optimal Classifier, Naïve Bayes classifier, Bayesian belief networks, EM algorithm. SUPPORT VECTOR MACHINE: Introduction, Types of support vector kernel - (Linear kernel, polynomial kernel, and Gaussian kernel), Hyperplane - (Decision surface), Properties of SVM, and Issues in SVM.
 - **(UNIT-3: DECISION TREE LEARNING)** DECISION TREE LEARNING - Decision tree learning algorithm, Inductive bias, Inductive inference with decision trees, Entropy and information theory, Information gain, ID-3 Algorithm, Issues in Decision tree learning. INSTANCE-BASED LEARNING - k-Nearest Neighbour Learning, Locally Weighted Regression, Radial basis function networks, Case-based learning.
 - **(UNIT-4: ARTIFICIAL NEURAL NETWORKS)** ARTIFICIAL NEURAL NETWORKS - Perceptron's, Multilayer perceptron, Gradient descent & the Delta rule, Multilayer networks, Derivation of Backpropagation Algorithm, Generalization, Unsupervised Learning - SOM Algorithm and its variant; DEEP LEARNING - Introduction, concept of convolutional neural network, Types of layers - (Convolutional Layers, Activation function, pooling, fully connected), Concept of Convolution (1D and 2D) layers, Training of network, Case study of CNN for eg on Diabetic Retinopathy, Building a smart speaker, Self-driving car etc.
 - **(UNIT-5: REINFORCEMENT LEARNING)** REINFORCEMENT LEARNING-Introduction to Reinforcement Learning, Learning Task, Example of Reinforcement Learning in Practice, Learning Models for Reinforcement - (Markov Decision process, Q Learning - Q Learning function, @ Learning Algorithm), Application of Reinforcement Learning, Introduction to Deep Q Learning. GENETIC ALGORITHMS: Introduction, Components, GA cycle of reproduction, Crossover, Mutation, Genetic Programming, Models of Evolution and Learning, Applications.
- <http://www.knowledgigate.in/gate>

(UNIT-1 : INTRODUCTION)

- Learning, Types of Learning
- Well defined learning problems, Designing a Learning System.
- History of ML
- Introduction of Machine Learning Approaches –
 - Artificial Neural Network
 - Clustering
 - Reinforcement Learning
 - Decision Tree Learning
 - Bayesian networks
 - Support Vector Machine
 - Genetic Algorithm),
- Issues in Machine Learning and
- Data Science Vs Machine Learning. <http://www.knowledgegate.in/gate>

Definition of Learning

- Learning involves changes in behaviour due to experiences, focusing on adapting rather than relying on instinct or temporary states.

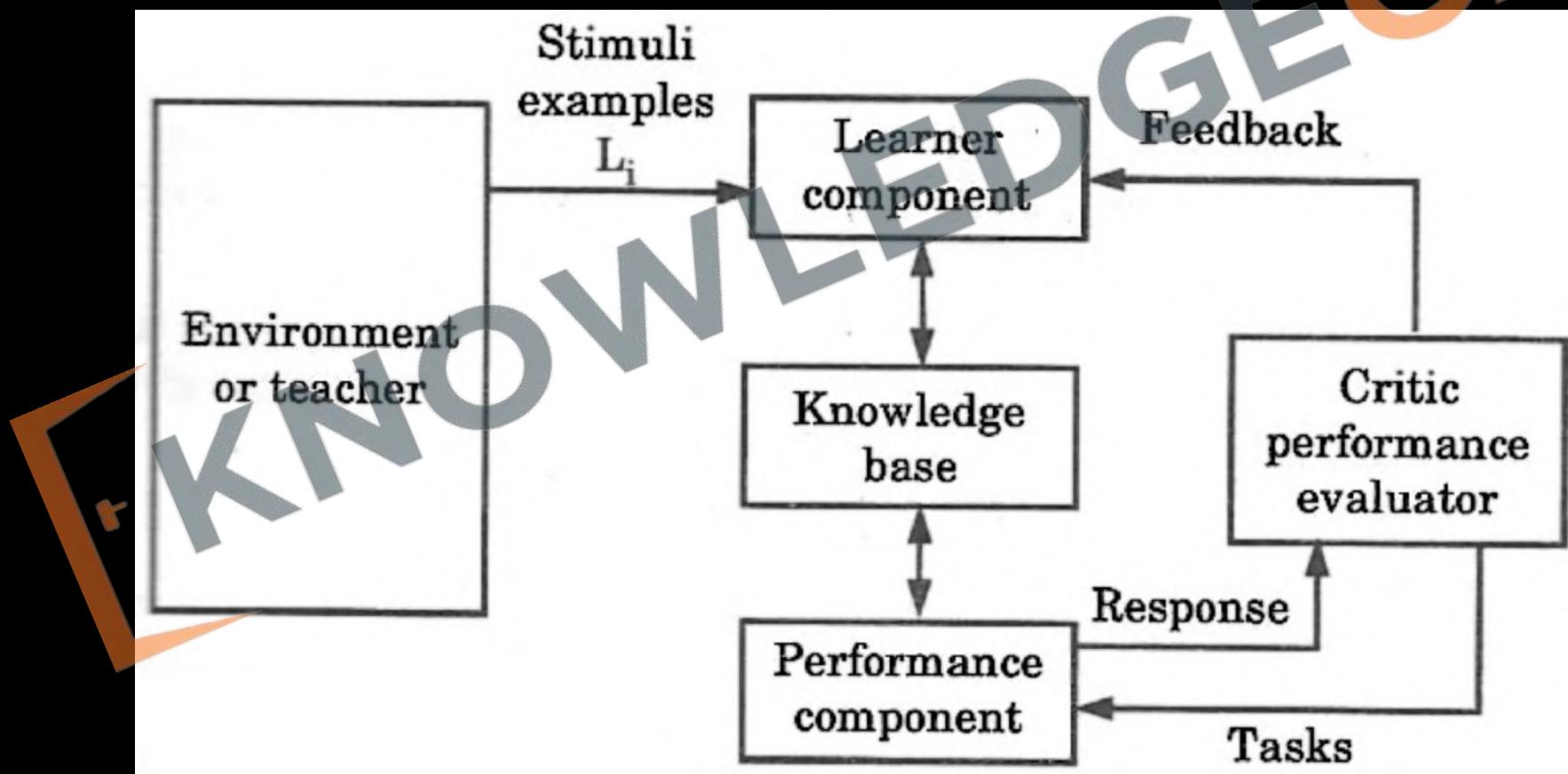


- **Components of a Learning System:**

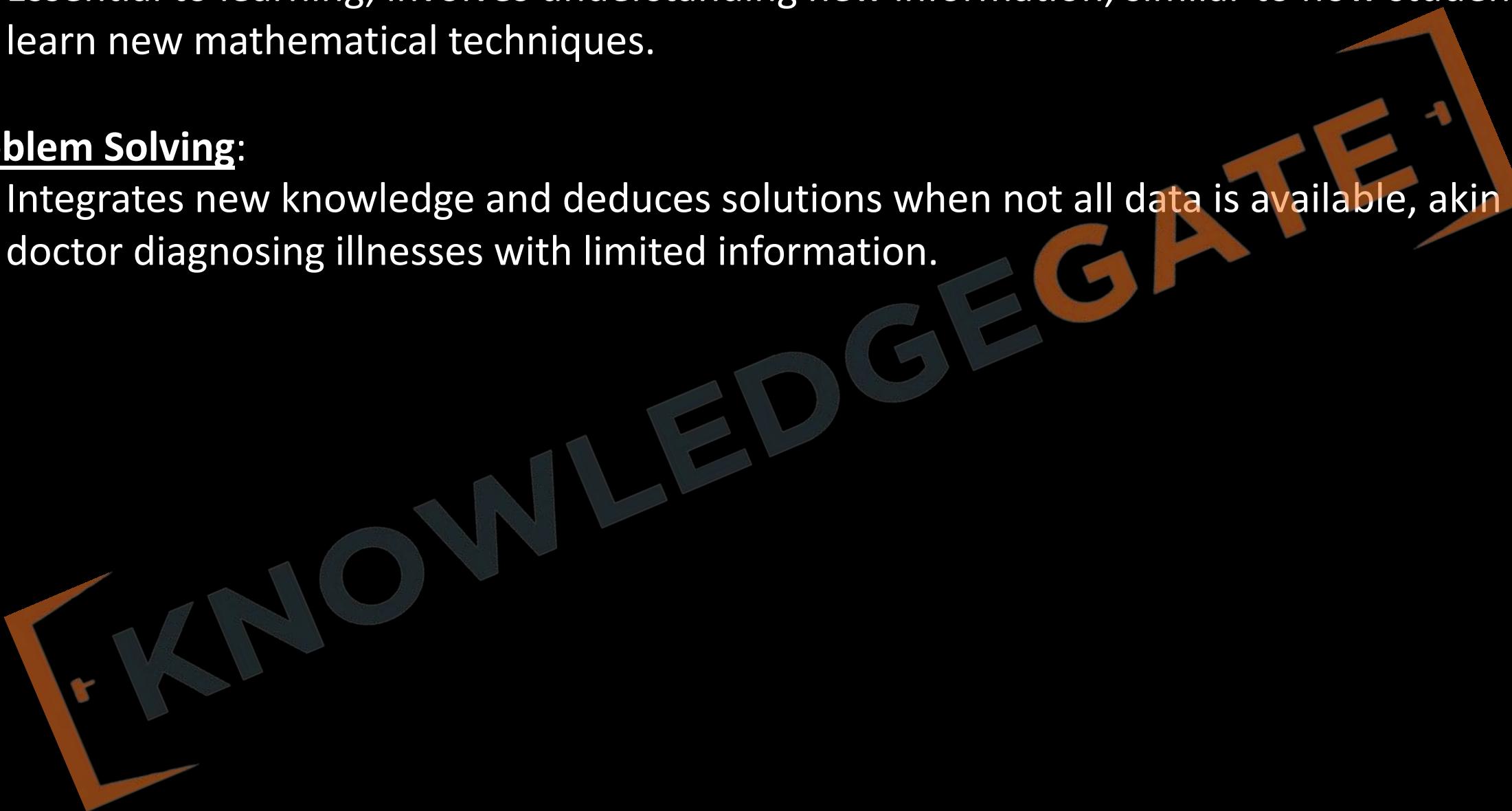
- **Performance Element:** Determines actions based on existing strategies.
- **Learning Element:** Improves the performance element by analyzing past outcomes.

Influences include:

- **Components of Performance:** Understanding existing capabilities.
- **Feedback Mechanism:** Using feedback to enhance performance.
- **Knowledge Representation:** How information is organized and accessed.



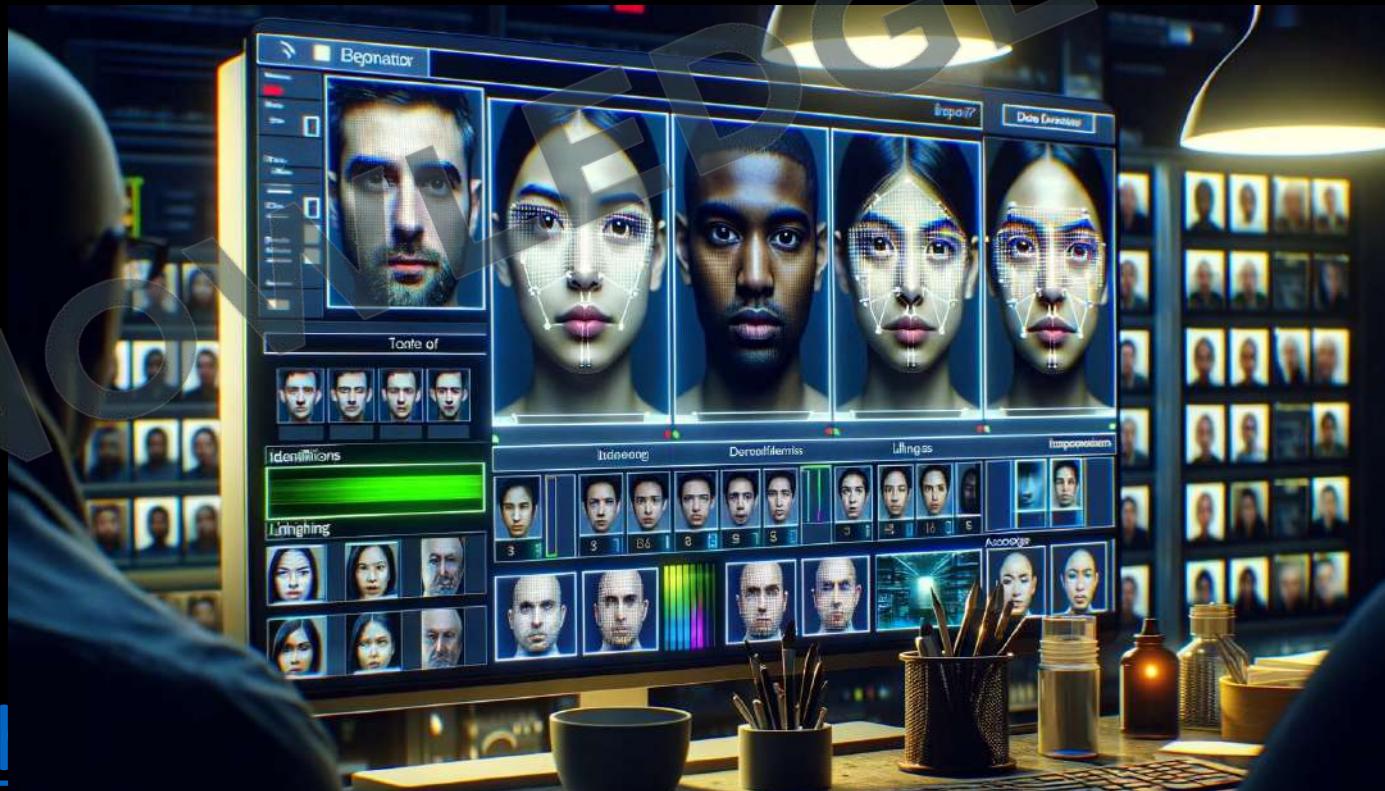
- **Acquisition of New Knowledge:**
 - Essential to learning; involves understanding new information, similar to how students learn new mathematical techniques.
- **Problem Solving:**
 - Integrates new knowledge and deduces solutions when not all data is available, akin to a doctor diagnosing illnesses with limited information.



<http://www.knowledgegate.in/gate>

Performance measures for learning

- Generality
 - Generality refers to a machine learning model's ability to perform well across various datasets and environments, not just the one it was trained on. For instance, a facial recognition system that can accurately identify faces in diverse lighting conditions and angles demonstrates good generality.



- Efficiency
 - Efficiency in machine learning measures how quickly a model can learn from data. A spam detection algorithm that quickly adapts to new types of spam emails with minimal training data exhibits high efficiency.



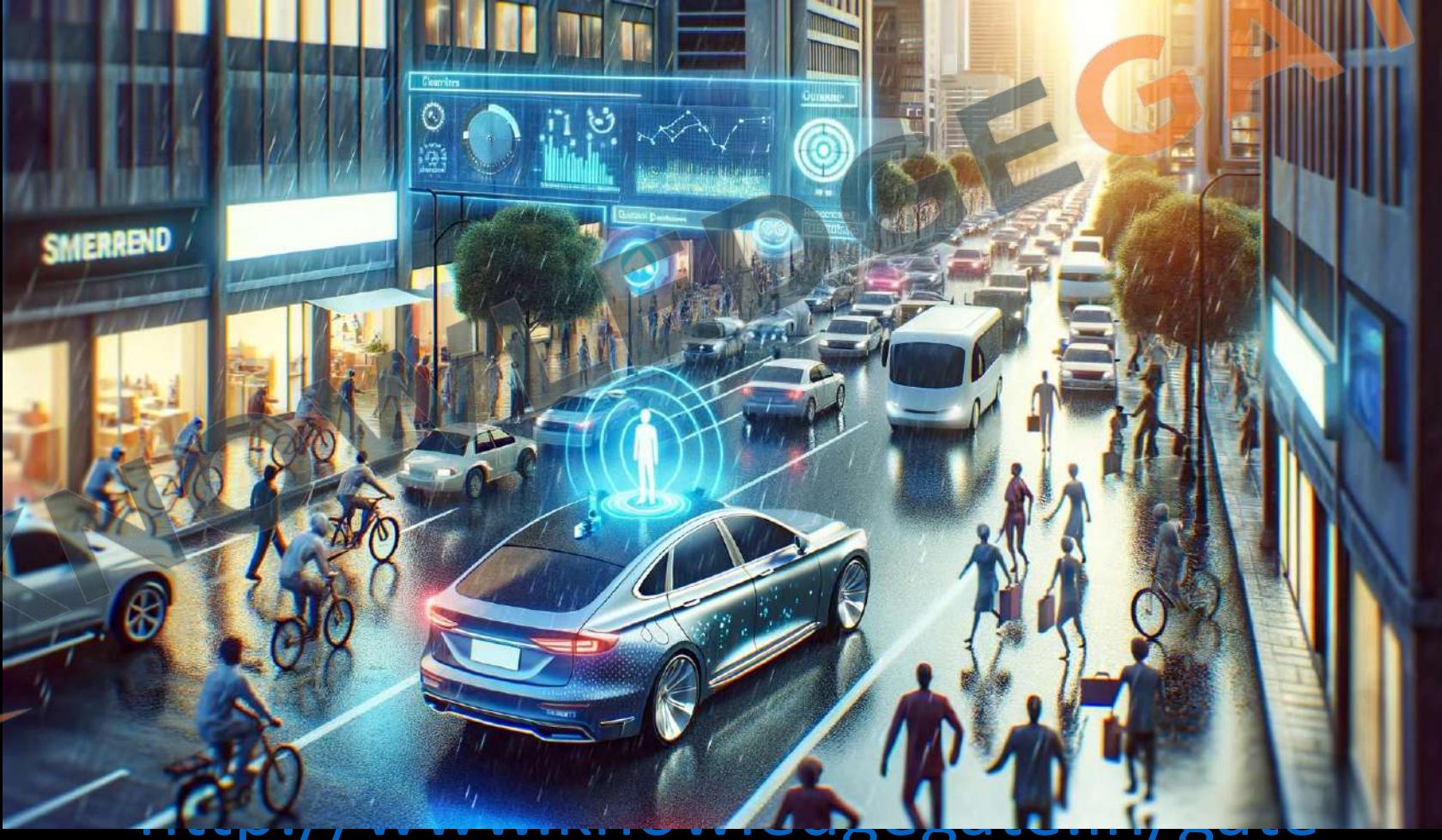
- **Robustness**

- Robustness is the ability of a model to handle errors, noise, and unexpected data without failing. A voice recognition system that can understand commands in a noisy room shows robustness.



- **Efficacy**

- Efficacy is the overall effectiveness of a machine learning model in performing its intended tasks. An autonomous driving system that safely navigates city traffic and avoids accidents under various conditions demonstrates high efficacy.



Ease of Implementation

- This measures how straightforward it is to develop and deploy a machine learning model. A recommendation system that can be integrated into an existing e-commerce platform using standard algorithms and software libraries highlights ease of implementation.



Supervised Learning

- Supervised learning involves training a machine learning model using labeled data, which means the data is already associated with the correct answer.
- **Example:** Consider teaching a child to identify fruits. You show them pictures of various fruits, like apples and bananas, while telling them, "This is an apple," and "This is a banana." Over time, the child learns to identify fruits correctly based on the examples given.



- Key Steps in Supervised Learning:
 - **Input and Output Pairing**: Each input (e.g., a fruit picture) is paired with its correct label (e.g., "apple").
 - **Training**: The model learns by comparing its prediction with the actual label and adjusting itself to improve accuracy.
 - **Error Correction**: If the model predicts incorrectly (e.g., calls an apple a banana), it adjusts its internal parameters to reduce the error.
 - **Outcome**: The model eventually learns to map inputs (fruit images) to the correct outputs (fruit names).

Unsupervised learning

- Unsupervised learning involves training a model without any labels, which means the model tries to identify patterns and data groupings on its own.
- Example: Imagine placing a mix of different coins on a table and asking a child to sort them. Without explaining any criteria, the child might start grouping the coins by size, color, or denomination on their own.



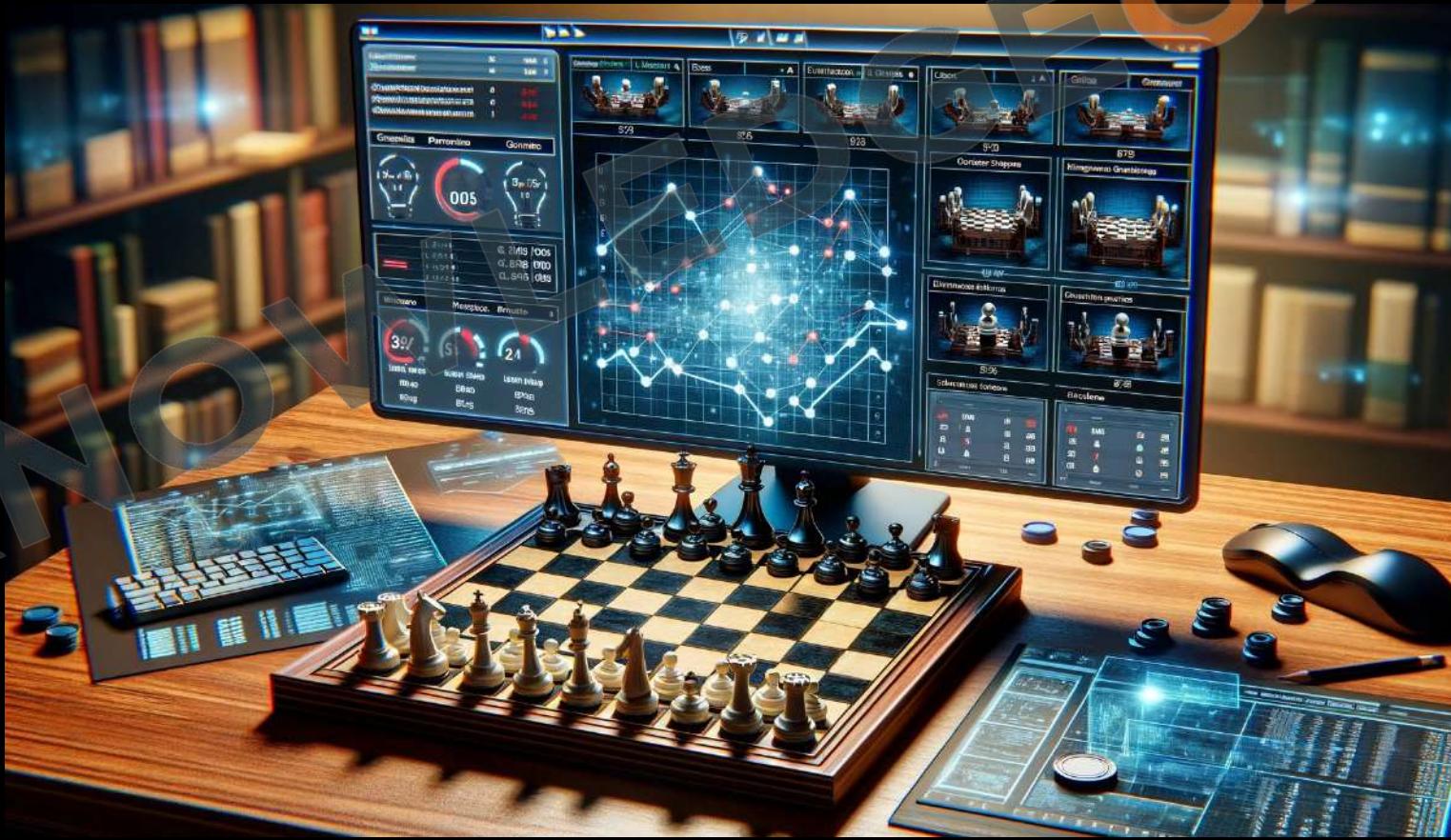
- Key Steps in Unsupervised Learning:
 - **Input Without Labels**: The model receives data without any explicit instructions on what to do with it.
 - **Pattern Recognition**: The model analyzes the data and tries to find any natural groupings or patterns (e.g., clustering coins based on size or color).
 - **Self-Organization**: The model organizes data into different categories based on the patterns it perceives.
 - **Outcome**: The model creates its own system of categorization without external guidance.

Well-defined learning problems

- A well-defined learning problem allows a computer program to improve at a specific task through experience. This is characterized by three key elements:
 - **Task (T)**: The specific activity or challenge the program is expected to perform.
 - **Performance Measure (P)**: The criteria used to gauge the program's effectiveness at the task.
 - **Experience (E)**: The data or interactions from which the program learns.

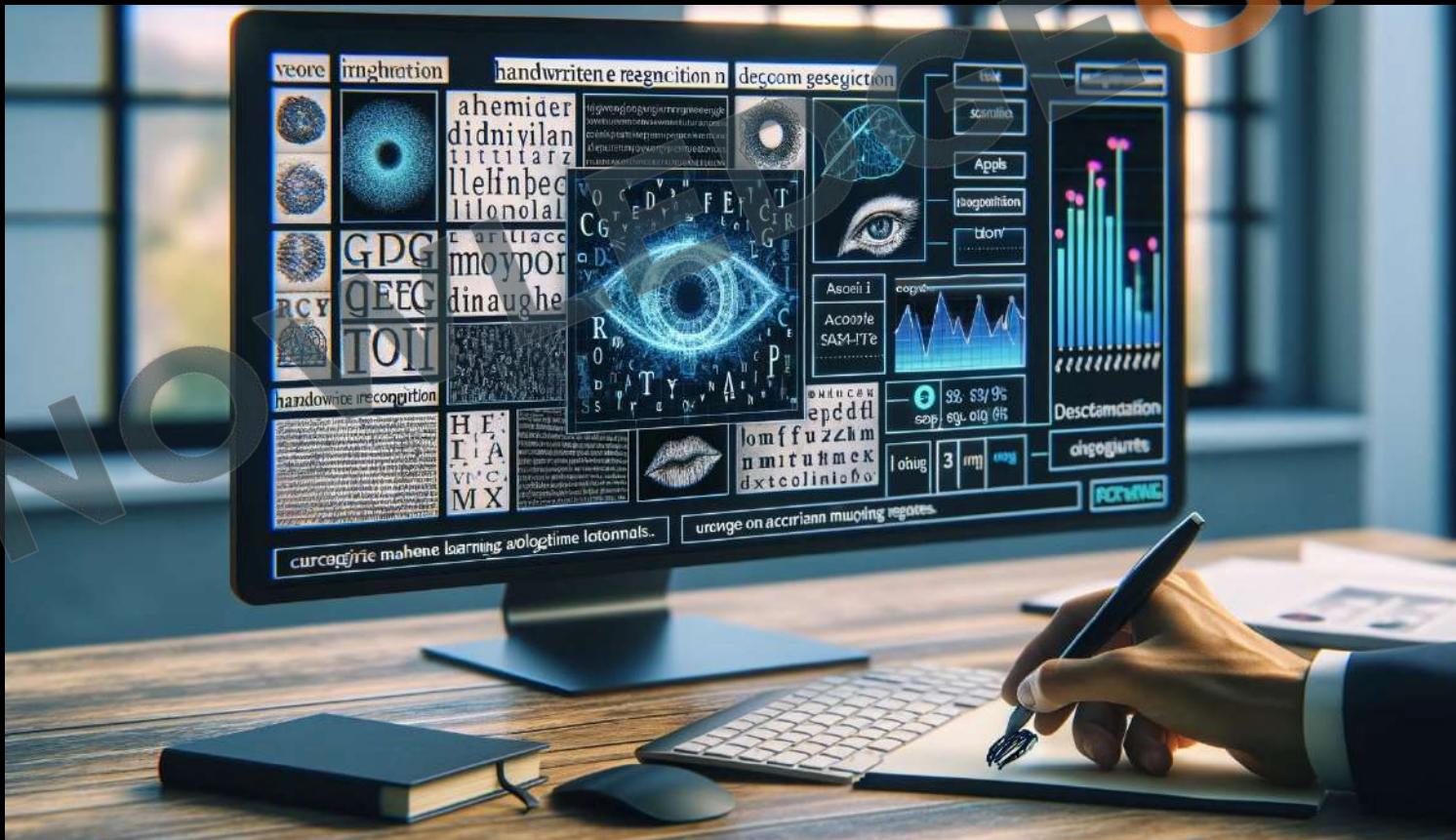
- Checkers Game:

- Task (T): Playing the game of checkers.
- Performance Measure (P): The percentage of games won against various opponents.
- Experience (E): Engaging in numerous practice games, possibly including self-play.



- **Handwriting Recognition:**

- **Task (T):** Identifying and categorizing handwritten words in images.
- **Performance Measure (P):** The accuracy rate, measured as the percentage of words correctly recognized.
- **Experience (E):** Analysis of a large dataset of labeled handwritten word images.



- **Autonomous Driving Robot:**

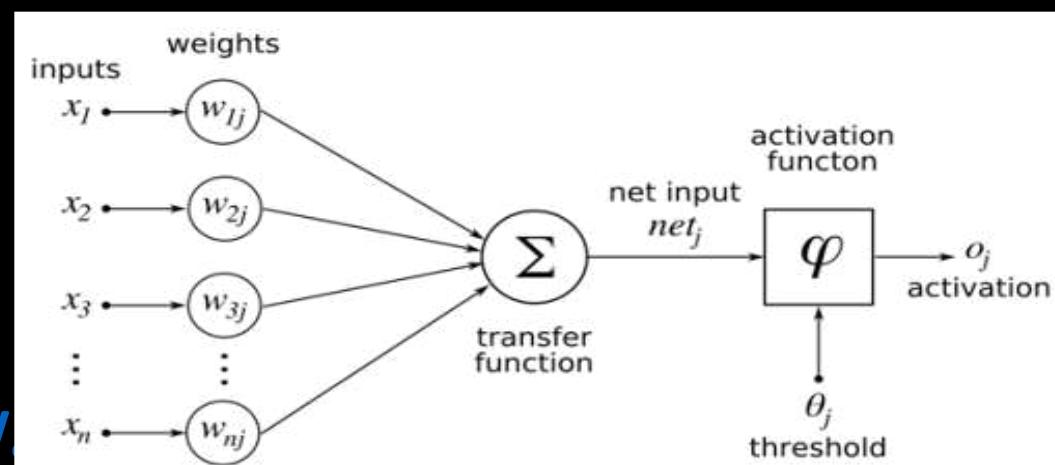
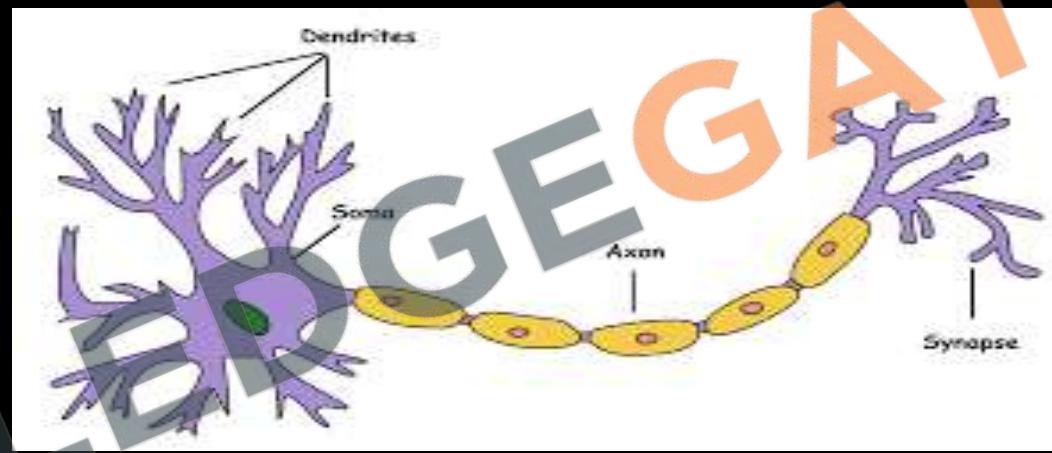
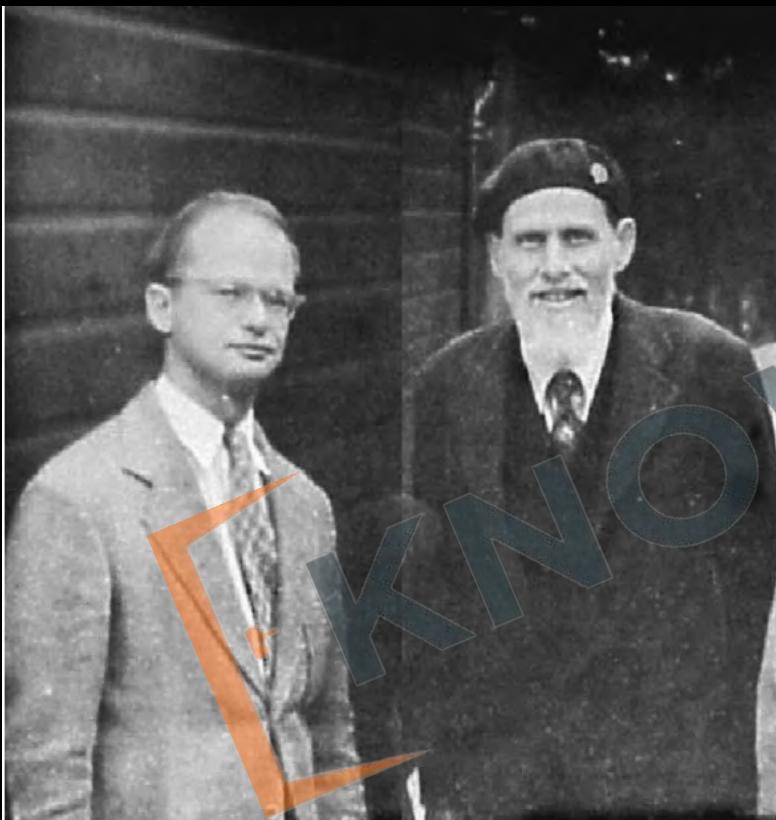
- **Task (T):** Navigating public four-lane highways using vision-based sensors.
- **Performance Measure (P):** The average distance the robot travels without making a mistake, as determined by a human supervisor.
- **Experience (E):** Processing sequences of images and corresponding steering commands previously collected from human drivers.



Overview of the history of Machine Learning

Early Developments:

- 1943: Neurophysiologist Warren McCulloch and mathematician Walter Pitts introduced the concept of a neural network by modeling neurons with electrical circuits.



<http://www>

Overview of the history of Machine Learning

Early Developments:

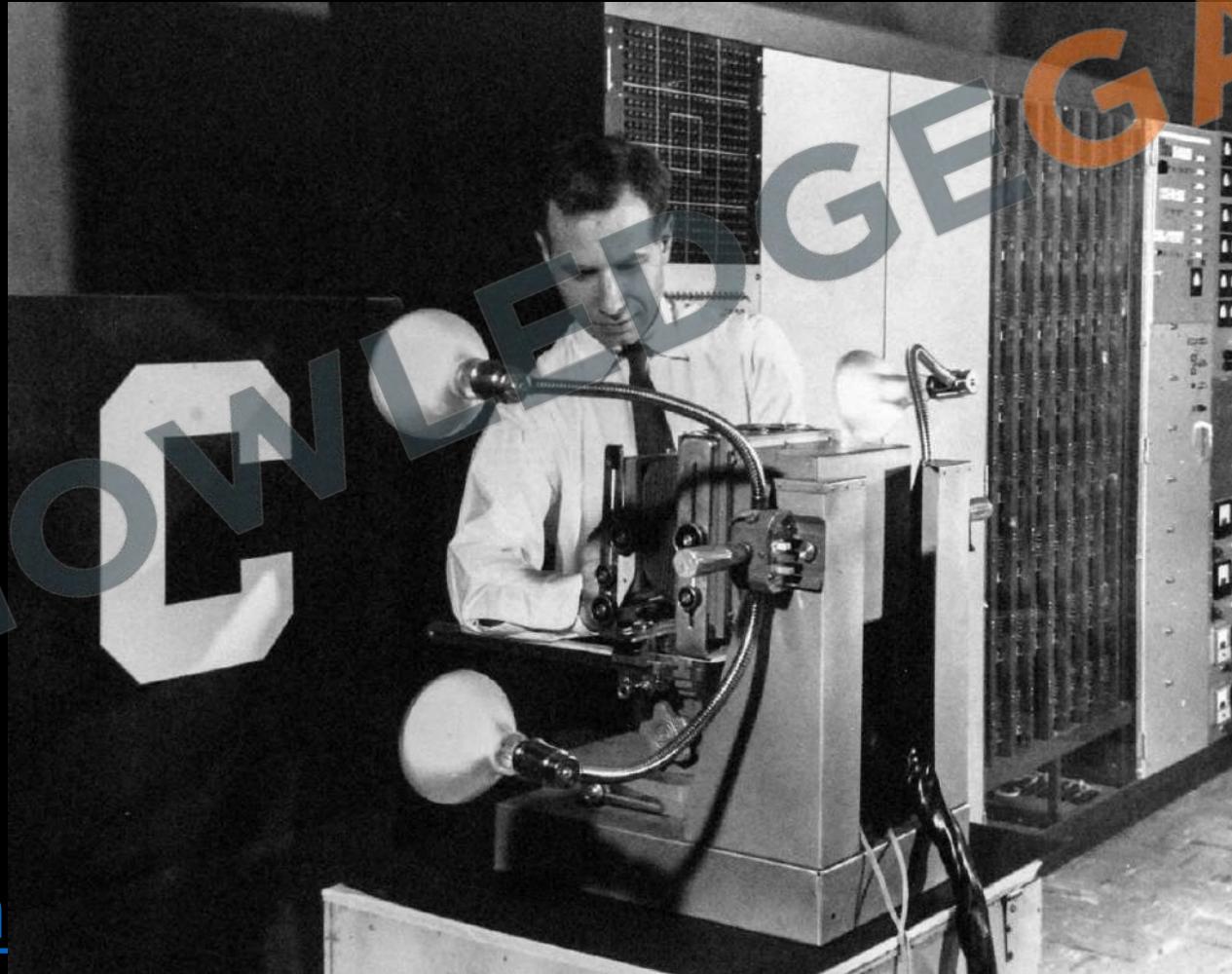
- 1952: Arthur Samuel developed the first computer program capable of learning from its activities.



Overview of the history of Machine Learning

Early Developments:

- 1958: Frank Rosenblatt created the Perceptron, the first artificial neural network, which was designed for pattern and shape recognition.



Overview of the history of Machine Learning

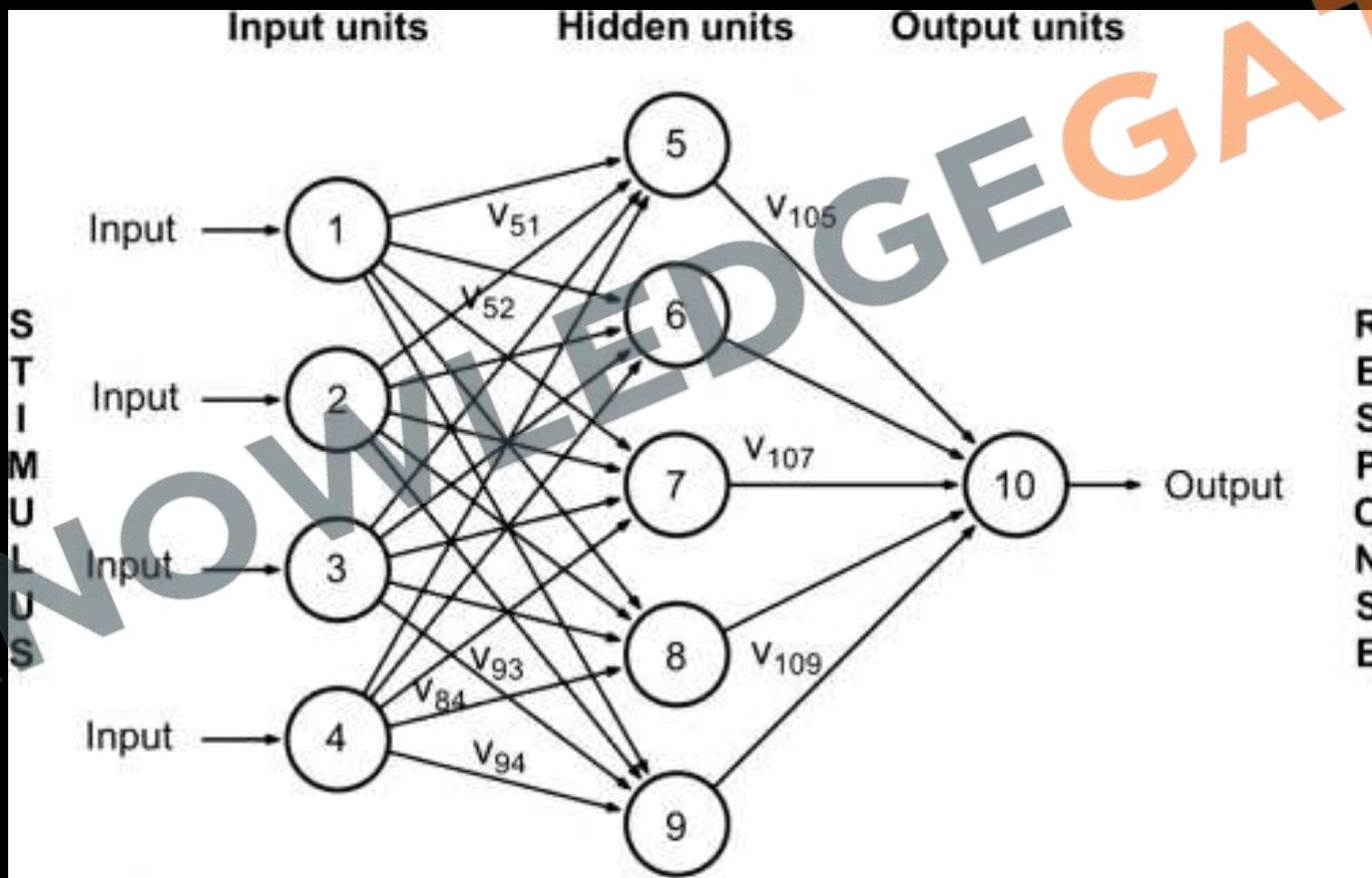
Early Developments:

- 1959: Bernard Widrow and Marcian Hoff developed two neural network models: ADELINe, which could detect binary patterns, and MADELINE, which was used to reduce echo on phone lines.



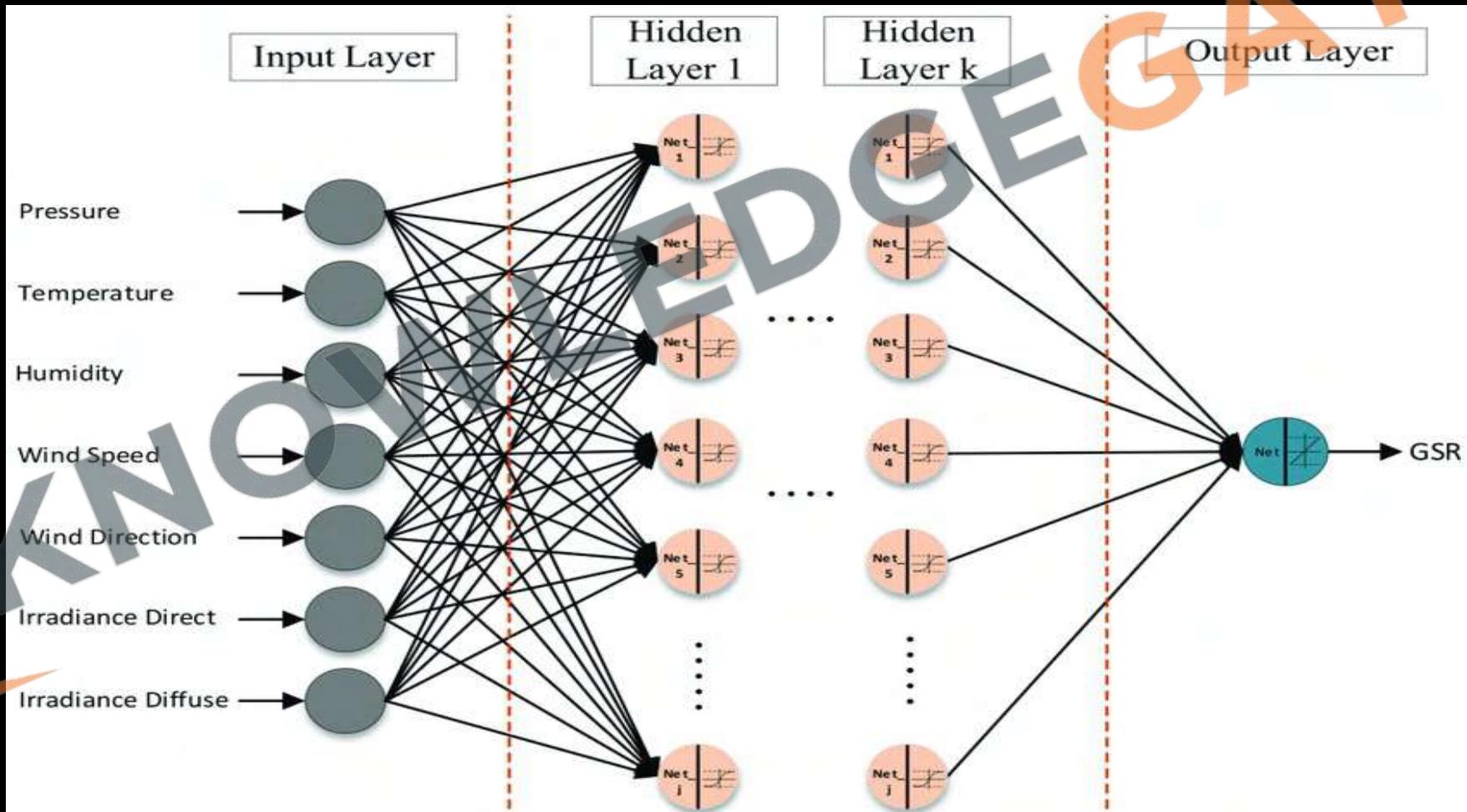
Advancements in the 1980s and 1990s:

- **1982**: John Hopfield proposed a network with bidirectional lines that mimicked actual neuronal structures.



Advancements in the 1980s and 1990s:

- 1986: The backpropagation algorithm was popularized, allowing the use of multiple layers in neural networks, enhancing their learning capabilities.



Advancements in the 1980s and 1990s:

- 1997: IBM's Deep Blue, a chess-playing computer, famously beat the reigning world chess champion.



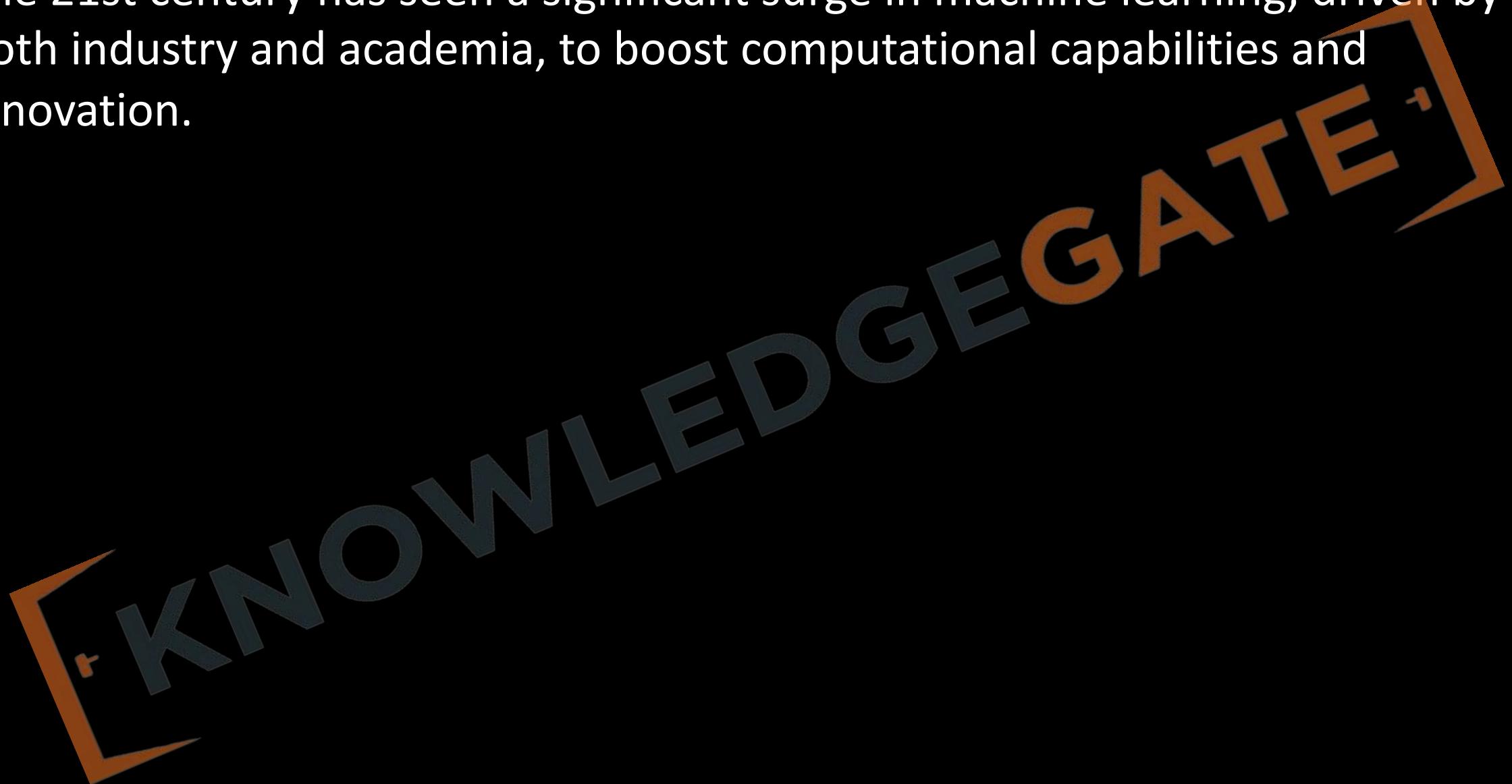
Advancements in the 1980s and 1990s:

- **1998**: AT&T Bell Laboratories achieved significant progress in digit recognition, notably enhancing the ability to recognize handwritten postcodes for the US Postal Service.



21st Century Innovations:

- The 21st century has seen a significant surge in machine learning, driven by both industry and academia, to boost computational capabilities and innovation.



<http://www.knowledgegate.in/gate>

- Notable projects include:
 - **GoogleBrain (2012)**: A deep learning project.
 - **AlexNet (2012)**: A deep convolutional neural network.
 - **DeepFace (2014)** and DeepMind (2014): Projects that advanced facial recognition and AI decision-making.
 - **OpenAI (2015), ResNet (2015), and U-net (2015)**: Each contributed to advancements in AI capabilities, from gameplay to medical imaging.

Machine learning

- Machine learning is a subset of artificial intelligence (AI) that enables computers to learn from and make decisions based on data, without being explicitly programmed.
- **Definition:** Machine learning involves developing algorithms that allow computers to process and learn from data automatically.
- **Purpose:** The aim is to enable computers to learn from their experiences and improve their performance over time without human intervention.
- **Functionality:** Machine learning algorithms analyze vast amounts of data, enabling them to perform tasks more efficiently and accurately. This could be anything from predicting consumer behavior to detecting fraudulent transactions.
- **Integration:** Combining machine learning with AI and cognitive technologies enhances its ability to process and interpret large volumes of complex data.

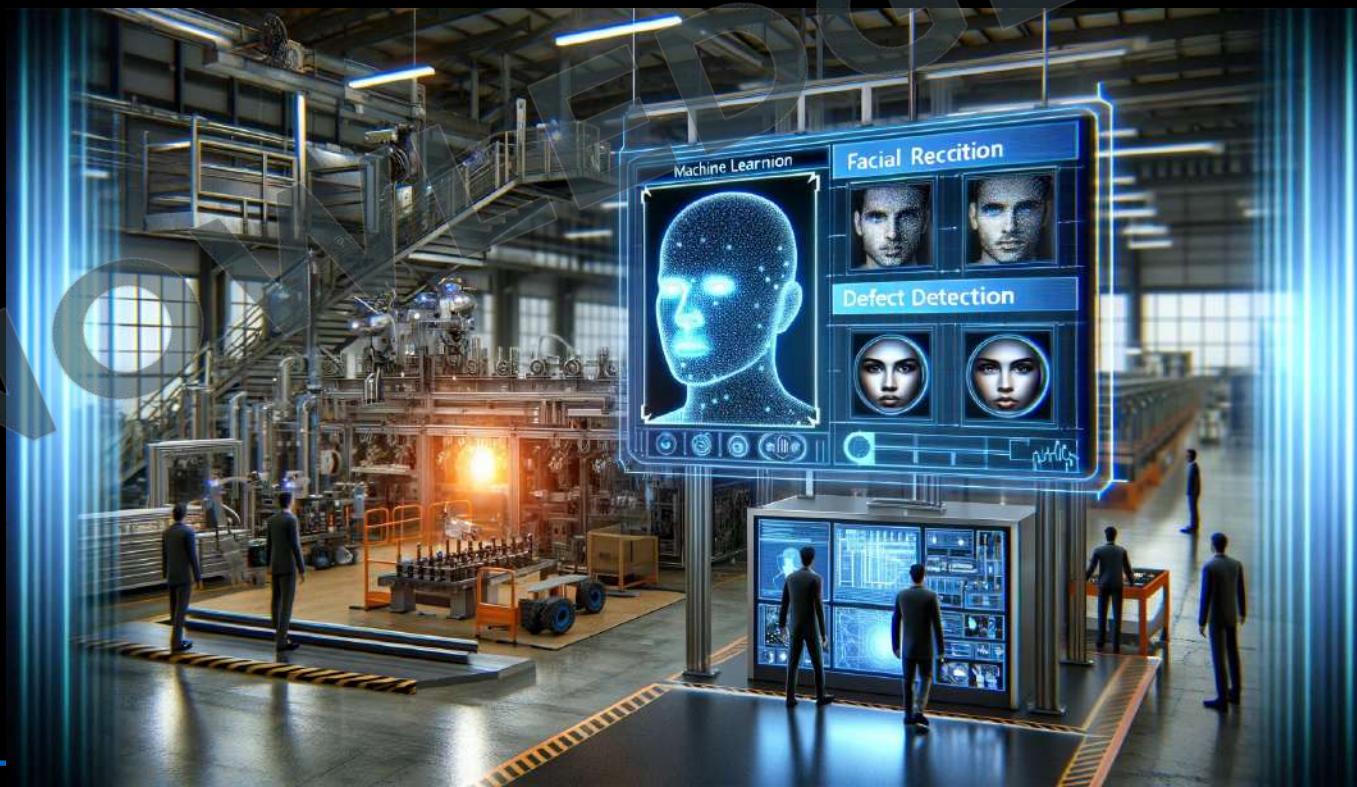
Example: Consider a streaming service like Netflix. Machine learning is used to analyze your viewing habits and the habits of others with similar tastes. Based on this data, the system recommends movies and shows that you might like. Here, the algorithm learns from the accumulated data to make increasingly accurate predictions over time, thereby enhancing user experience without manual intervention. This demonstrates machine learning's capability to adapt and improve autonomously, making it a powerful tool in many tech-driven applications.



Machine learning has a wide range of applications across different fields. Here are some key applications along with examples:

- **Image Recognition:**

- **Application:** Image recognition involves identifying objects, features, or patterns within digital images or videos.
- **Example:** Used in facial recognition systems for security purposes or to detect defective products on assembly lines in manufacturing.

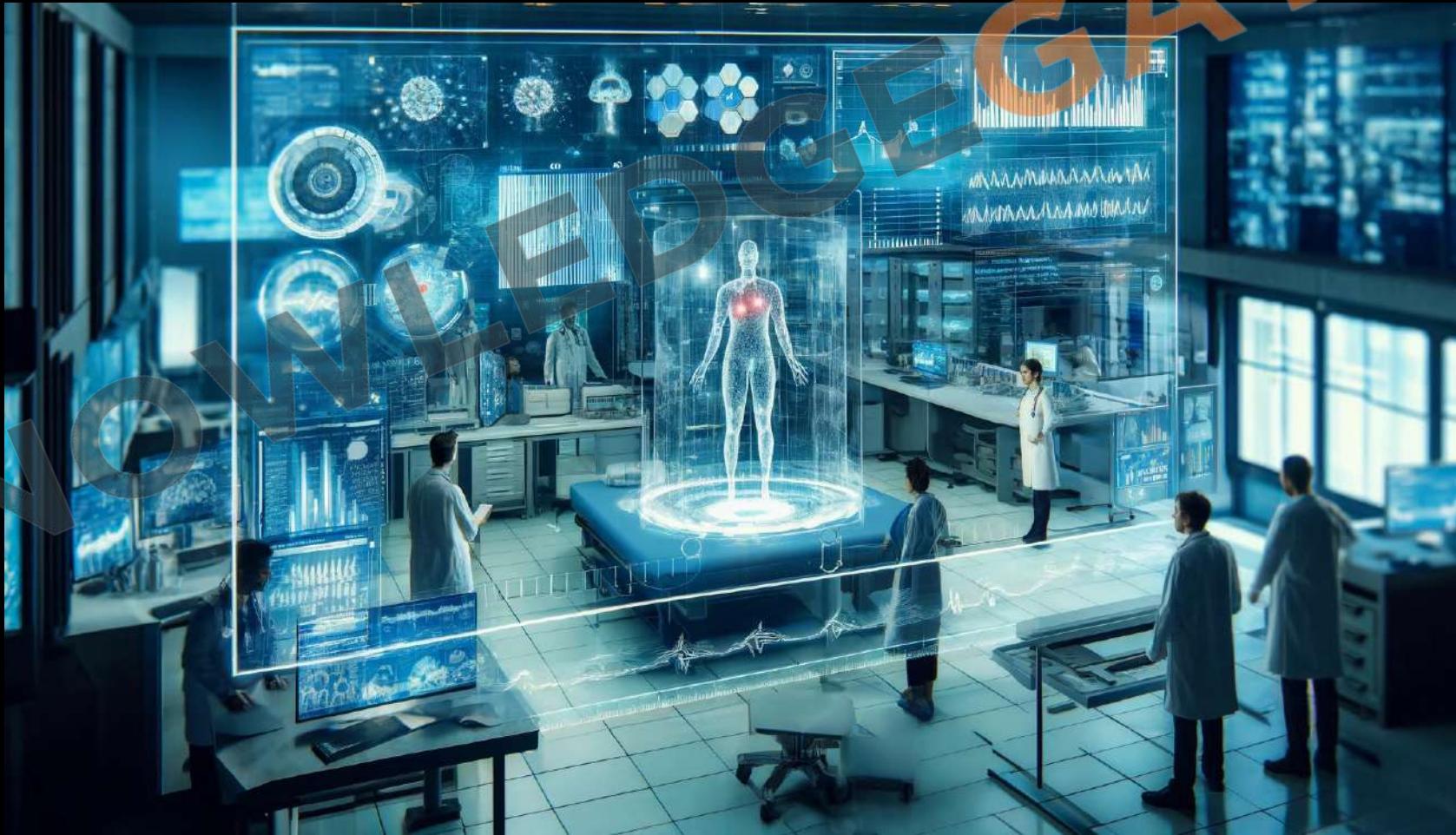


- **Speech Recognition:**
 - **Application:** Speech recognition technology converts spoken words into text, facilitating user interaction with devices and applications.
 - **Example:** Virtual assistants like Siri and Alexa use speech recognition to understand user commands and provide appropriate responses.



- **Medical Diagnosis:**

- **Application:** Machine learning assists in diagnosing diseases by analyzing clinical parameters and their combinations.
- **Example:** Predicting diseases such as diabetes or cancer by examining patient data and previous case histories to identify patterns that precede diagnoses.



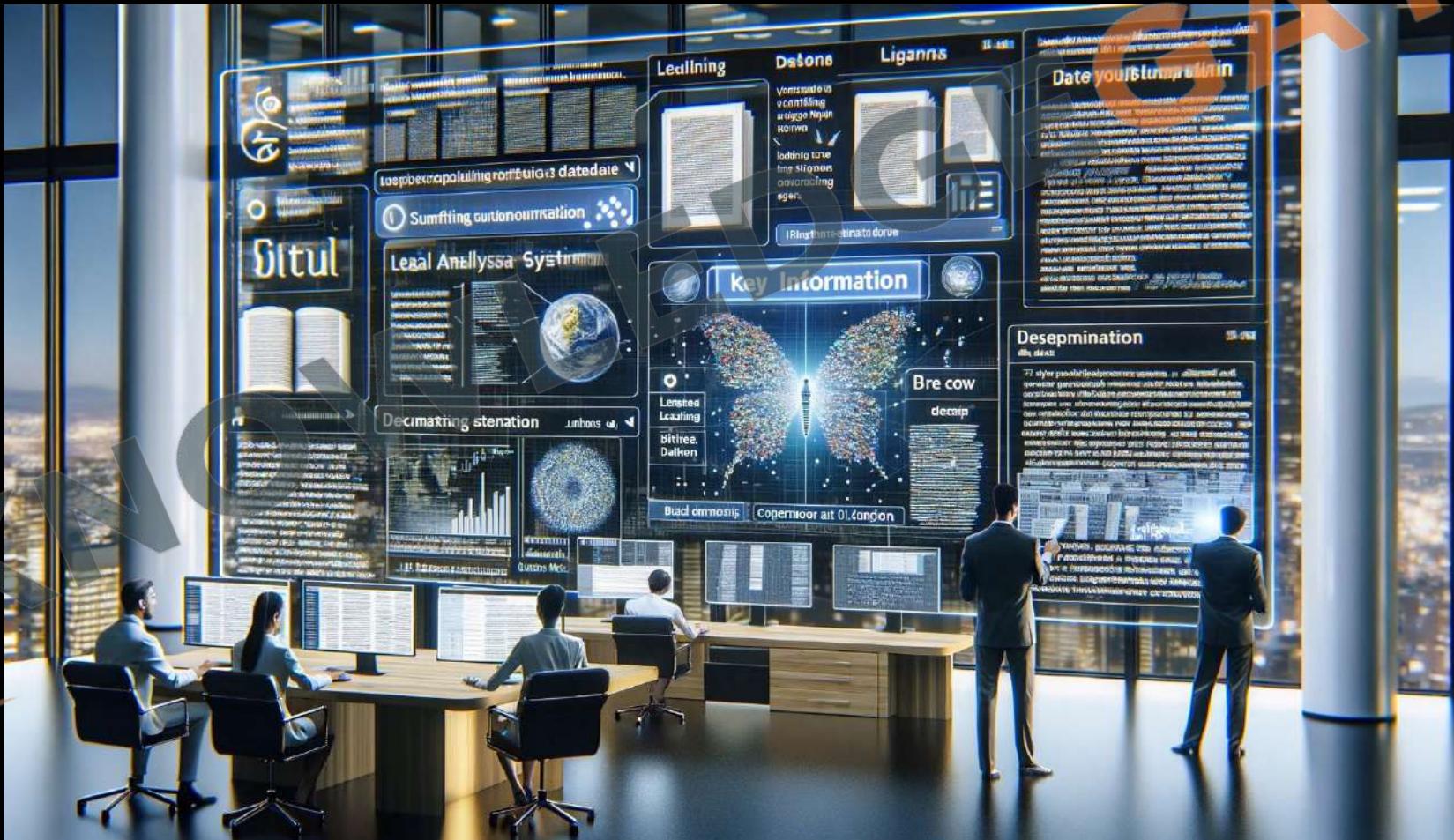
- **Statistical Arbitrage:**
 - **Application:** In finance, statistical arbitrage involves automated trading strategies that capitalize on patterns identified in trading data.
 - **Example:** Algorithmic trading platforms that analyze historical stock data to make buy or sell decisions in milliseconds to capitalize on market inefficiencies.



- **Learning Associations:**
 - **Application:** This process uncovers relationships between variables in large databases, often revealing hidden patterns.
 - **Example:** Market basket analysis in retail, which analyzes purchasing patterns to understand product associations and optimize store layouts.



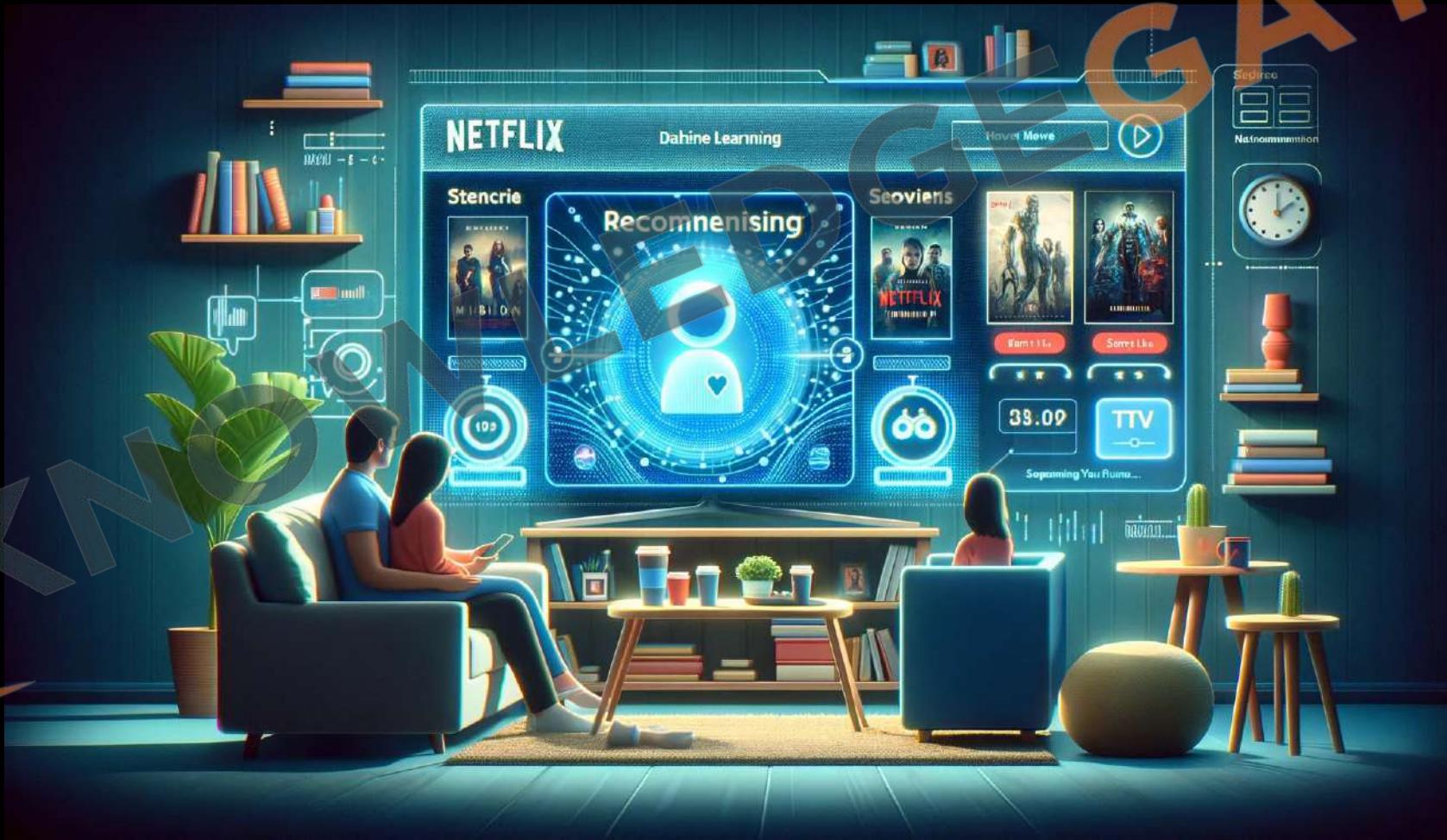
- **Information Extraction:**
- **Application:** Information extraction involves pulling structured information from unstructured data, like text.
- **Example:** Extracting key pieces of information from legal documents or news articles to summarize content or populate databases automatically.



Advantages of Machine Learning:

- Identifies Trends and Patterns:

- Example: Streaming services like Netflix analyze viewer data to identify viewing patterns and recommend shows and movies that individual users are likely to enjoy.



Advantages of Machine Learning:

- **Automation:**
 - **Example:** Autonomous vehicles use machine learning to interpret sensory data and make driving decisions without human input, improving transportation efficiency and safety.



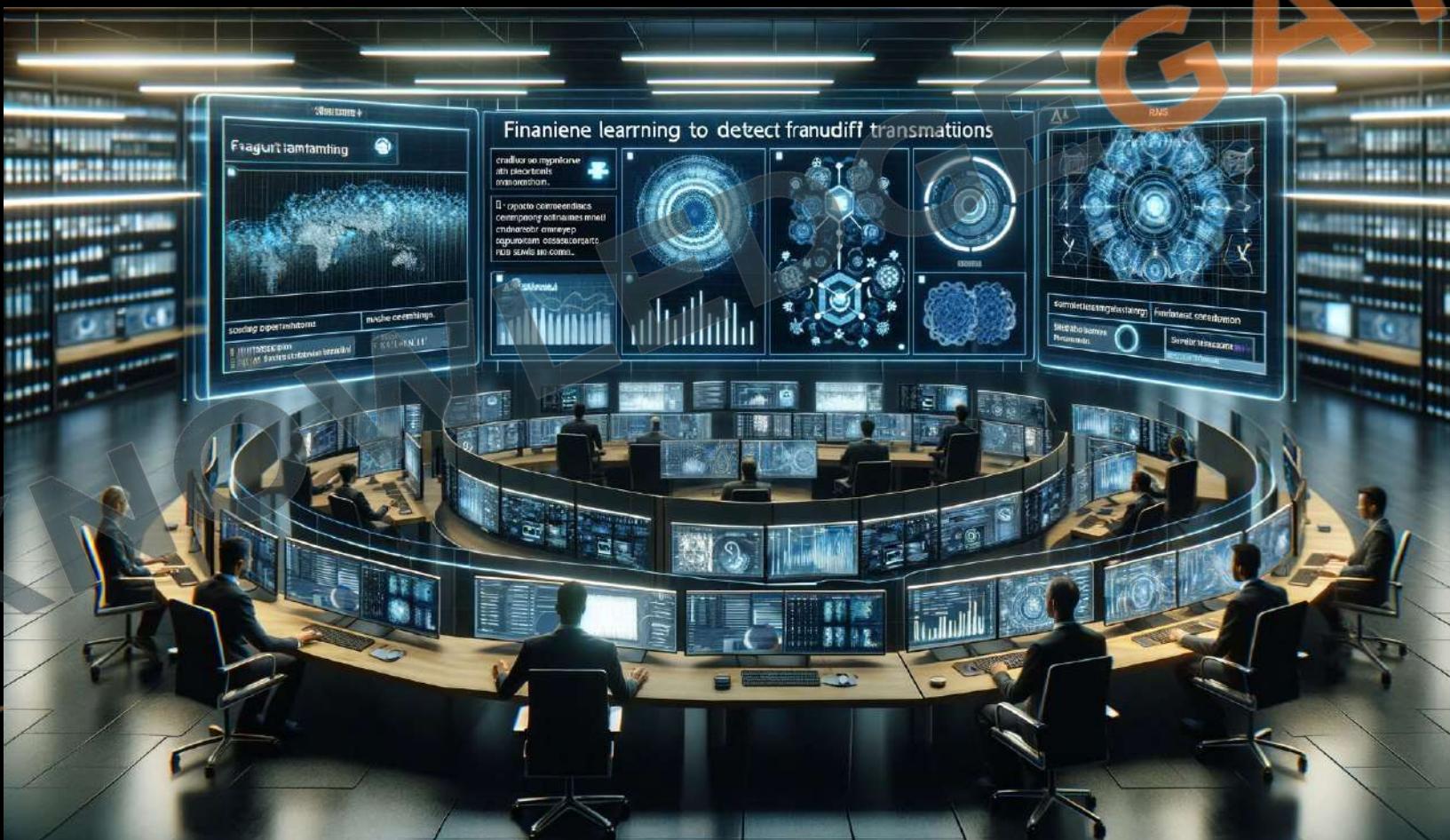
Advantages of Machine Learning:

- Continuous Improvement:
 - Example: Credit scoring systems evolve by learning from new customer data, becoming more accurate in predicting creditworthiness over time.



Advantages of Machine Learning:

- Handling Complex Data:
 - Example: Financial institutions use machine learning algorithms to detect fraudulent transactions by analyzing complex patterns of customer behavior that would be difficult for humans to process.



Disadvantages of Machine Learning:

- **Data Acquisition:**
 - **Example:** In healthcare, acquiring large datasets of patient medical records that are comprehensive and privacy-compliant is challenging and expensive.

Disadvantages of Machine Learning:

- **Time and Resources:**
 - **Example:** Developing a machine learning model for predicting stock market trends requires extensive computational resources and time to analyze years of market data before it can be deployed.

Disadvantages of Machine Learning:

- **Interpretation of Results:**
 - **Example:** In genomics research, interpreting the vast amounts of data produced by machine learning algorithms requires highly specialized knowledge to ensure findings are accurate and meaningful.

Disadvantages of Machine Learning:

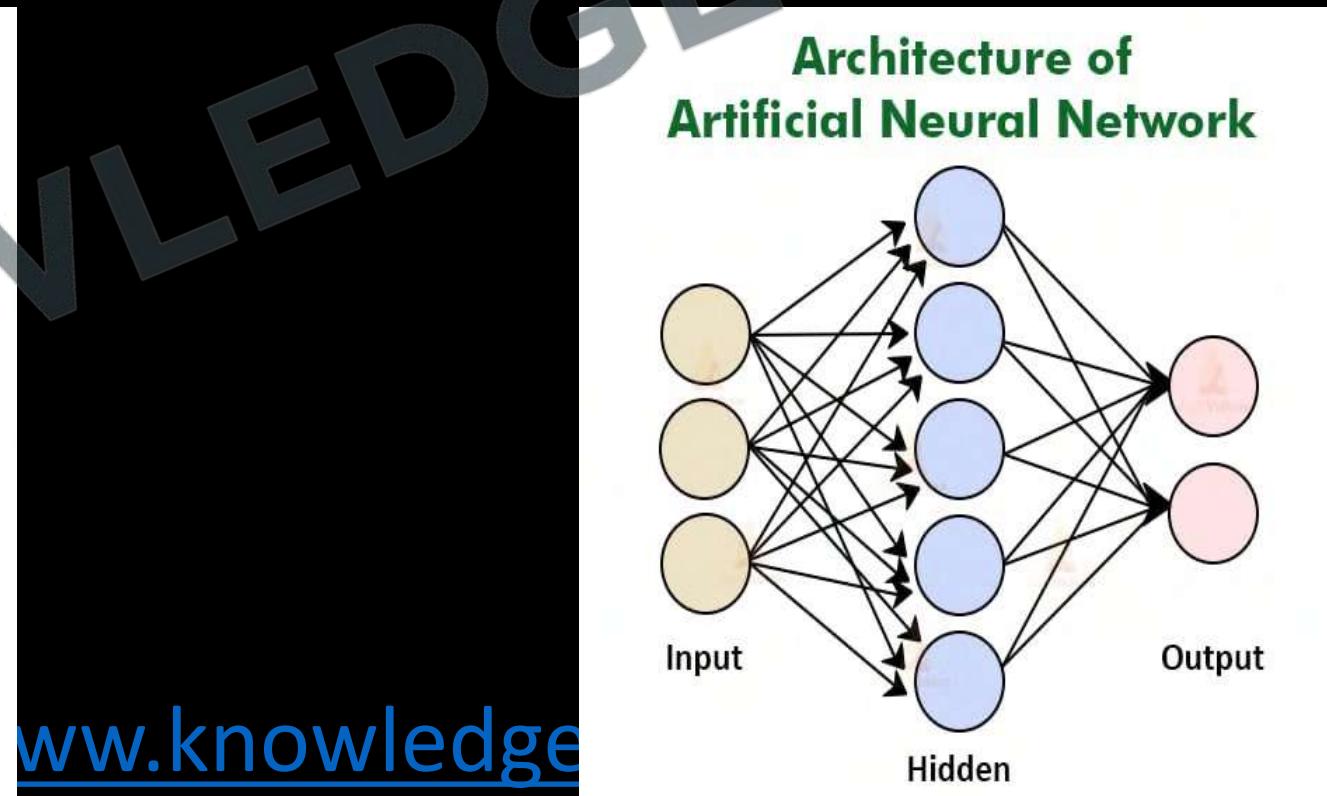
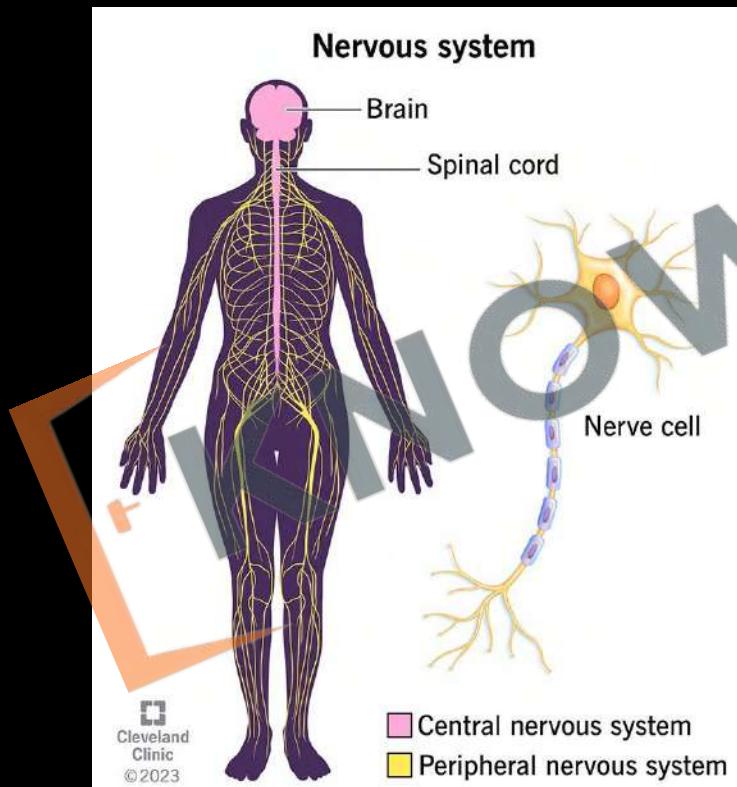
- **High Error-Susceptibility**:
 - **Example**: Early stages of facial recognition technology showed high error rates, particularly in accurately identifying individuals from minority groups, leading to potential biases and inaccuracies.

Machine Learning Approaches

Artificial Neural Network

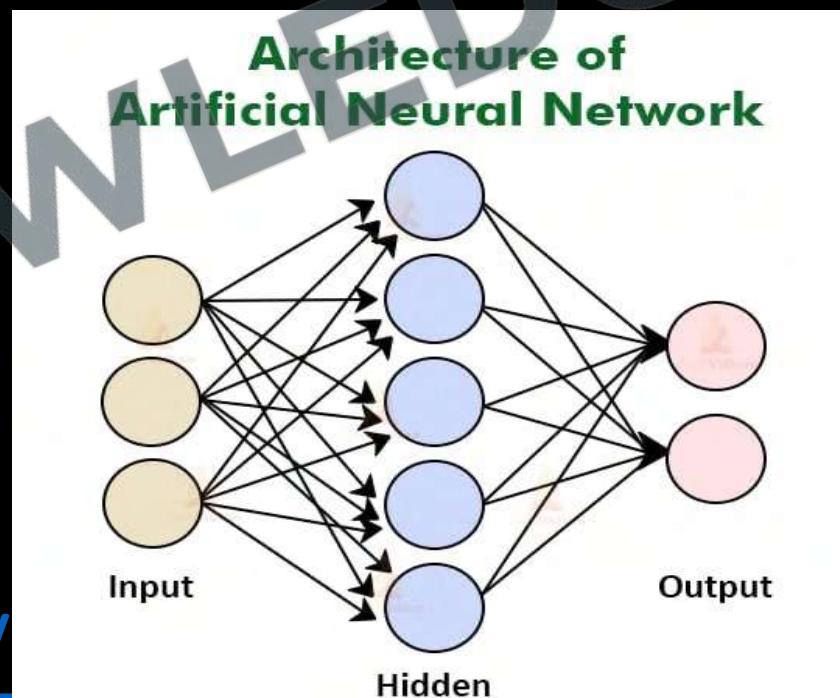
- Overview of ANNs:

- Inspiration: ANNs mimic the structure and function of the nervous systems in animals, particularly how neurons transmit signals.
- Functionality: These networks are used for machine learning and pattern recognition, handling complex data inputs effectively.



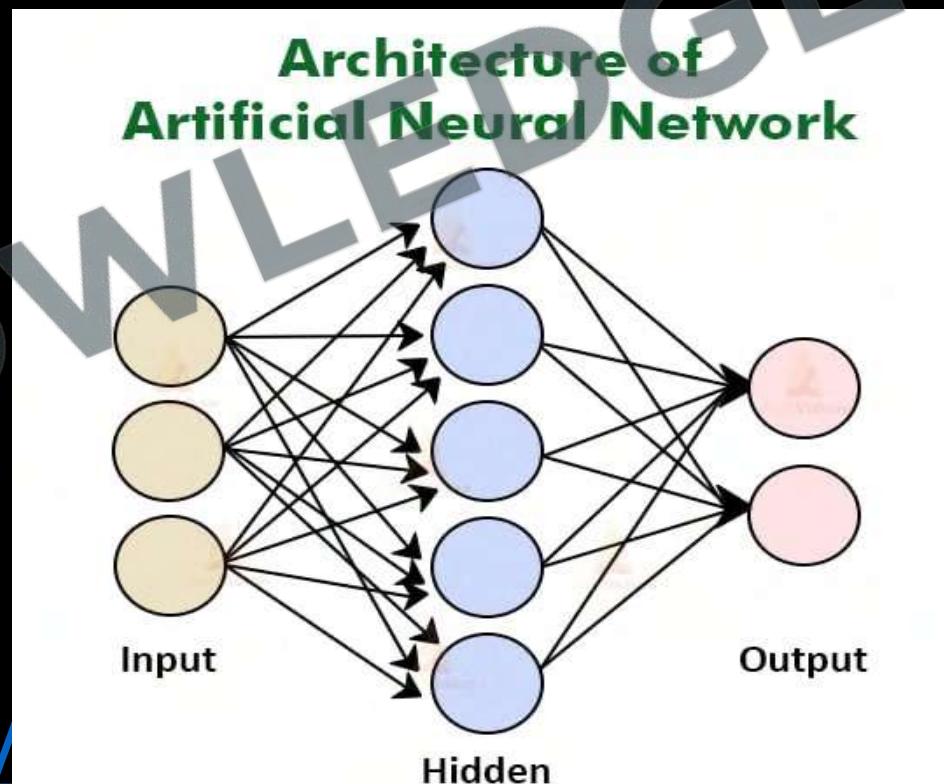
Artificial Neural Network

- Components of ANNs:
 - **Neurons**: Modeled as nodes within a network.
 - **Connections**: Nodes are linked by arcs that represent synapses, with weights that signify the strength of each connection.
 - **Processing**: The network processes signals in a way analogous to neural activity in biological brains.



Artificial Neural Network

- Operation:
 - Signal Transmission: Connections in the network facilitate the propagation of data, similar to synaptic transmission in biology.
 - Information Processing: ANNs adjust the weights of connections to learn from data and make informed decisions.

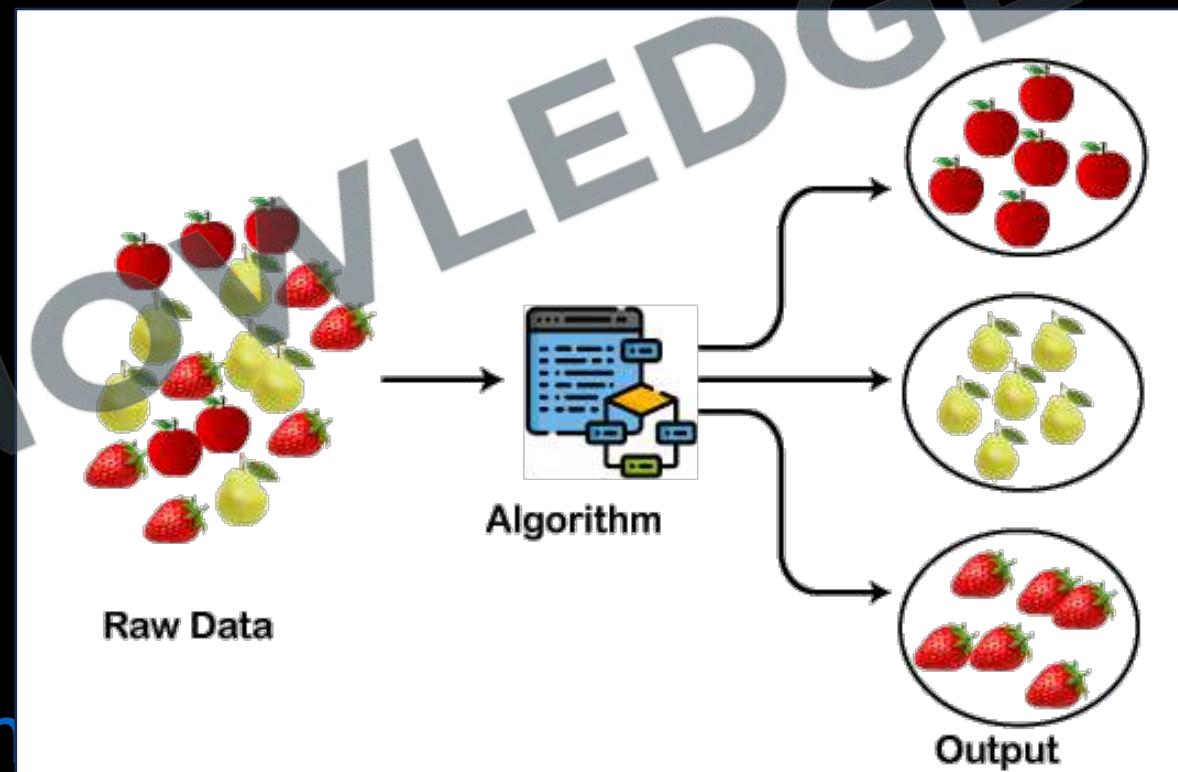


<http://>

/gate

Clustering

- **Definition:** Clustering is the process of sorting items into groups based on their similarities, forming distinct clusters where items within each cluster are more alike to each other than to those in other clusters.
- **Visual Representation:** Imagine organizing fruits into groups by type, such as grouping apples together, oranges in another group, and bananas in a separate one, visually representing how clusters segregate similar items.



- **Characteristics**: Clusters act like exclusive clubs, where members share common traits but differ significantly from members of other clusters, illustrating the distinctiveness of each group.
- **Multidimensional Space**: Clusters are akin to islands in an expansive ocean, with dense population points representing similar items within each cluster, and low-density water symbolizing dissimilar items separating clusters.

- **Machine Learning Perspective**: Clustering entails discovering patterns without explicit guidance, akin to exploring a forest without a map, where similarities guide the grouping process. It's a form of unsupervised learning, akin to solving a puzzle without knowledge of the final solution.
- **Unsupervised Learning**: Clustering is learning through observation, not instruction. It's like solving a puzzle without knowing what the final picture looks like.

- **Data Reduction:**
 - **Example:** Imagine sorting a massive collection of books into genres (fiction, non-fiction, sci-fi, etc.). Clustering reduces the data into manageable chunks for easier processing.
- **Hypothesis Generation:**
 - **Example:** Grouping customer purchase data to generate hypotheses about shopping preferences, which can then be tested with additional research.
- **Hypothesis Testing:**
 - **Example:** Using clustering to verify if certain customer segments show different purchasing behaviors, confirming or disproving existing hypotheses.
- **Prediction Based on Groups:**
 - **Example:** Suppose we have a dataset of customer demographics and spending habits. By clustering similar customers, we can predict the behavior of new customers based on their group's characteristics. For instance, if a new customer shares similarities with the "budget-conscious" cluster, we can predict their spending patterns accordingly.

Clustering

Classification

1.

Clustering analyzes data objects without known class label.

In classification, data are grouped by analyzing the data objects whose class label is known.

2.

There is no prior knowledge of the attributes of the data to form clusters.

There is some prior knowledge of the attributes of each classification.

3.

It is done by grouping only the input data because output is not predefined.

It is done by classifying output based on the values of the input data.

4.

The number of clusters is not known before clustering. These are identified after the completion of clustering.

The number of classes is known before classification as there is predefined output based on input data.

5.

Unknown class label

Known class label

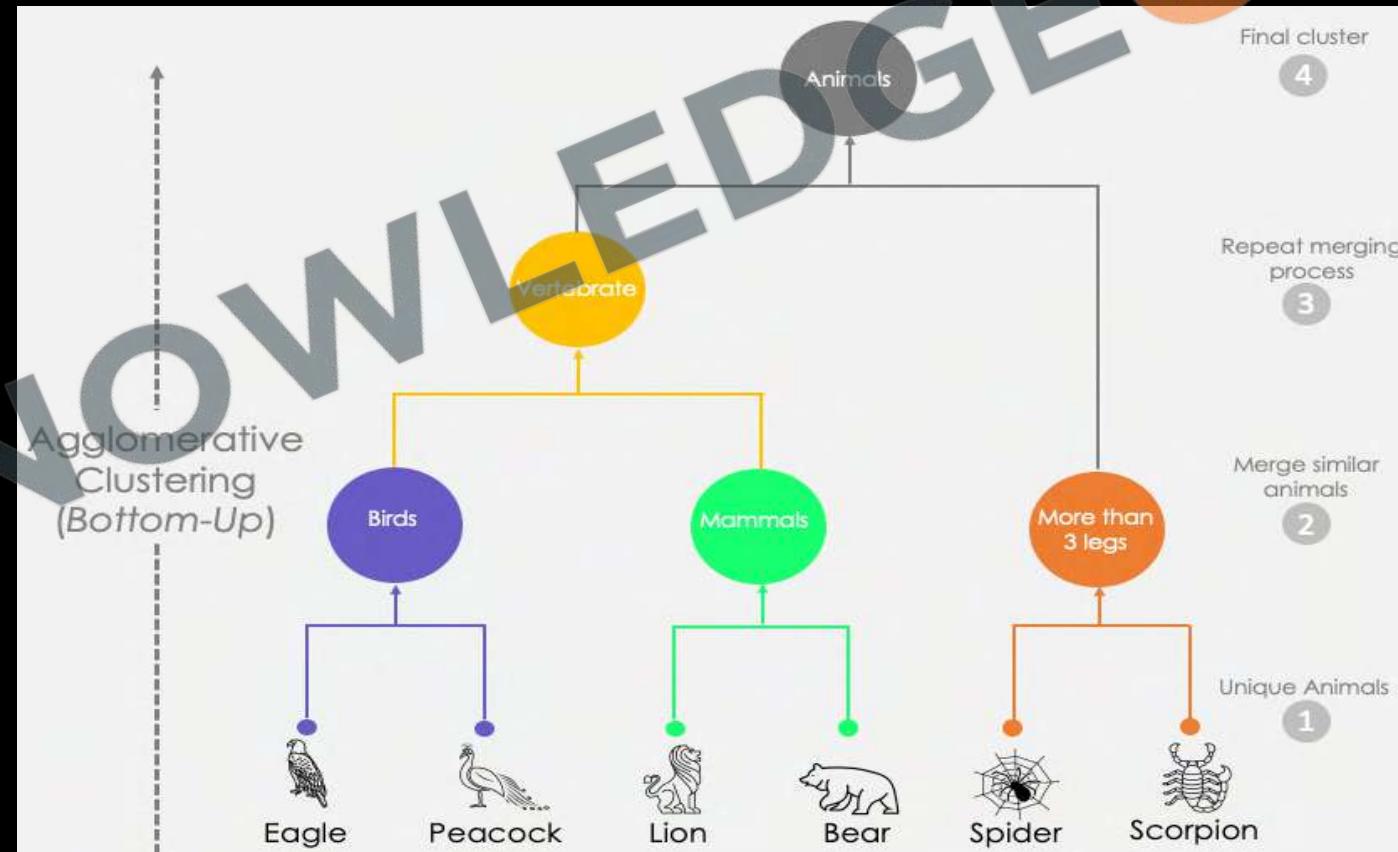
6.

It is considered as unsupervised learning because there is no prior knowledge of the class labels.

It is considered as the supervised learning because class labels are known before.

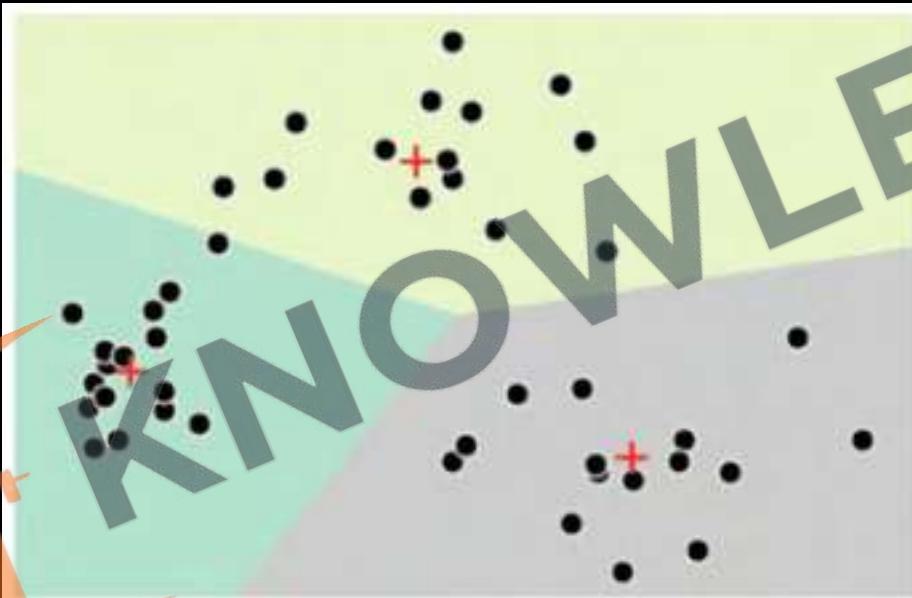
- Hierarchical Clustering:

- Agglomerative Hierarchical Clustering: Treats each data point as its own cluster, then merges clusters into larger ones. For example, a dataset of academic papers starts with each paper as its own cluster, then papers on similar topics merge into bigger clusters.
- Divisive Hierarchical Clustering: Starts with all data points in one cluster and splits them into smaller clusters. For instance, starting with one cluster of all store customers, the cluster is split based on purchasing behavior until each customer forms their own cluster.



- **Partitional Clustering:**

- **Centroid-based Clustering (e.g., K-means):** Partitions data into clusters, each represented by a centroid. Clusters minimize distance between data points and centroid, optimizing intra-cluster similarity and inter-cluster dissimilarity. For example, retail customers can be clustered by buying patterns, with each cluster's centroid reflecting average behavior.
- **Model-based Clustering:** Uses a statistical model for each cluster, finding the best data fit. For instance, Gaussian mixture models assume data points in each cluster are Gaussian distributed. This method is used in image processing to model different textures as coming from different Gaussian distributions.

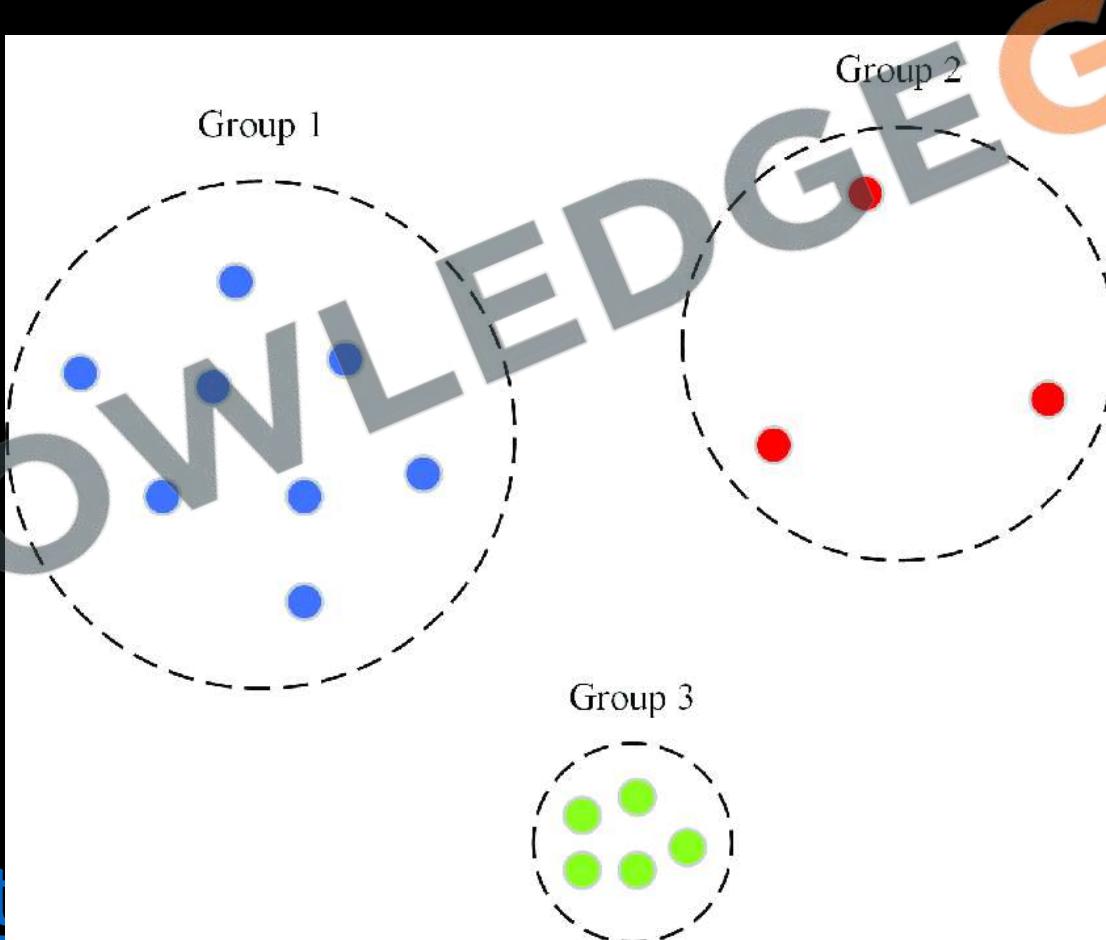


K-means clustering

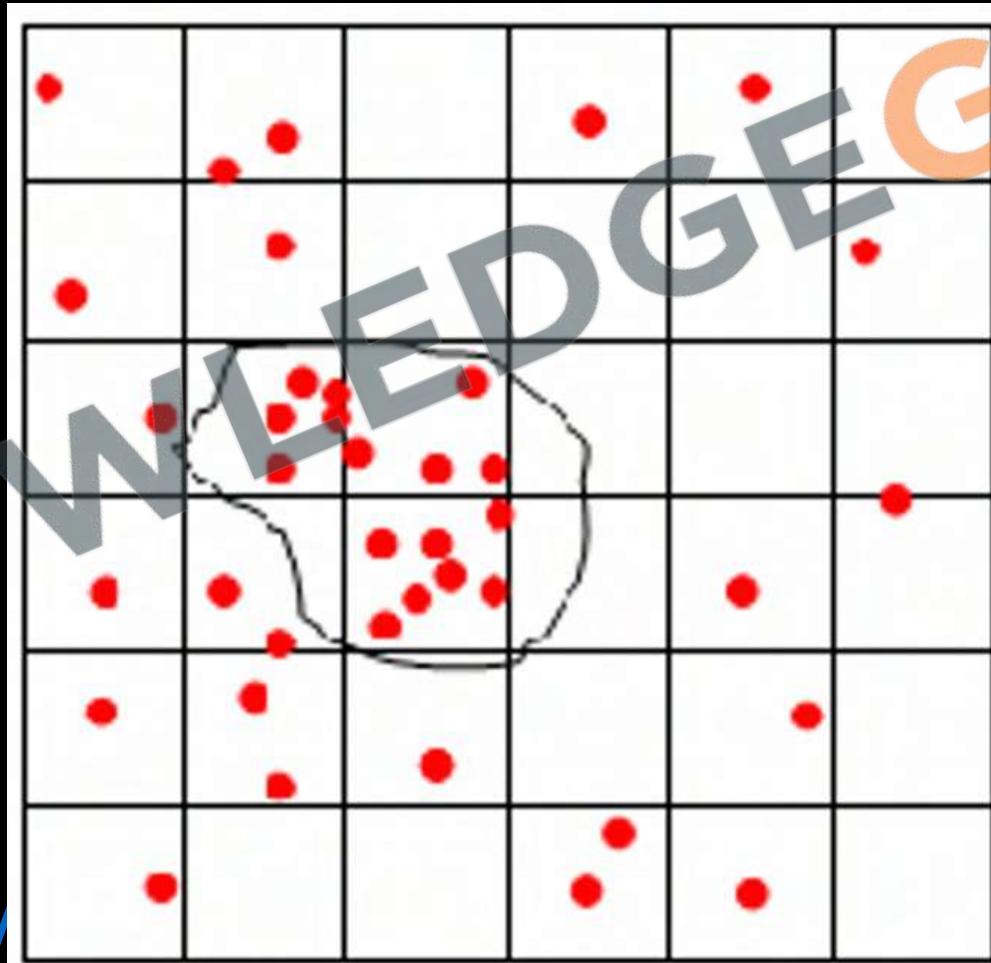


Mixture model (Gaussian)

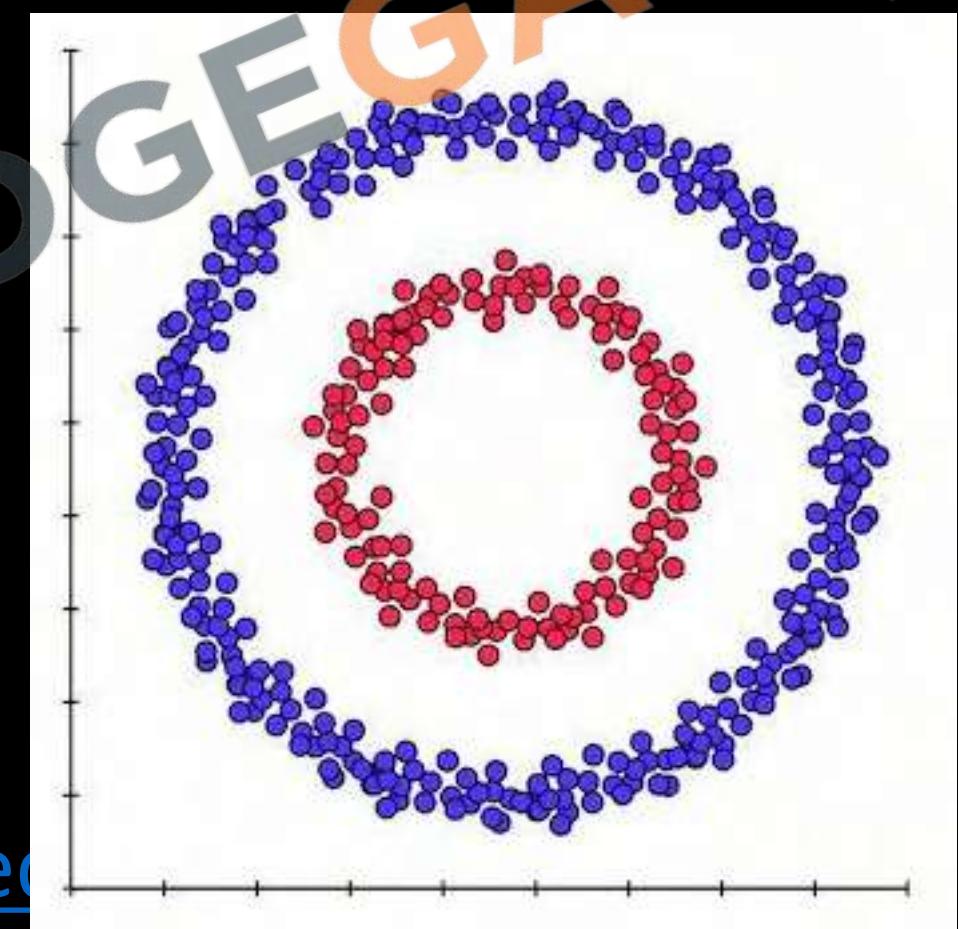
- Density-based Clustering (e.g., DBSCAN):
 - This method clusters points that are closely packed together, marking as outliers points that lie alone in low-density regions. This is useful in geographical data analysis where, for example, identifying regions of high economic activity based on point density of businesses can be achieved.



- **Grid-based Clustering:**
 - This method quantizes the space into a finite number of cells that form a grid structure and then performs clustering on the grid structure. This is effective for large spatial data sets, as it speeds up the clustering process. For example, in meteorological data, clustering can be applied to grid squares to categorize regional weather patterns.

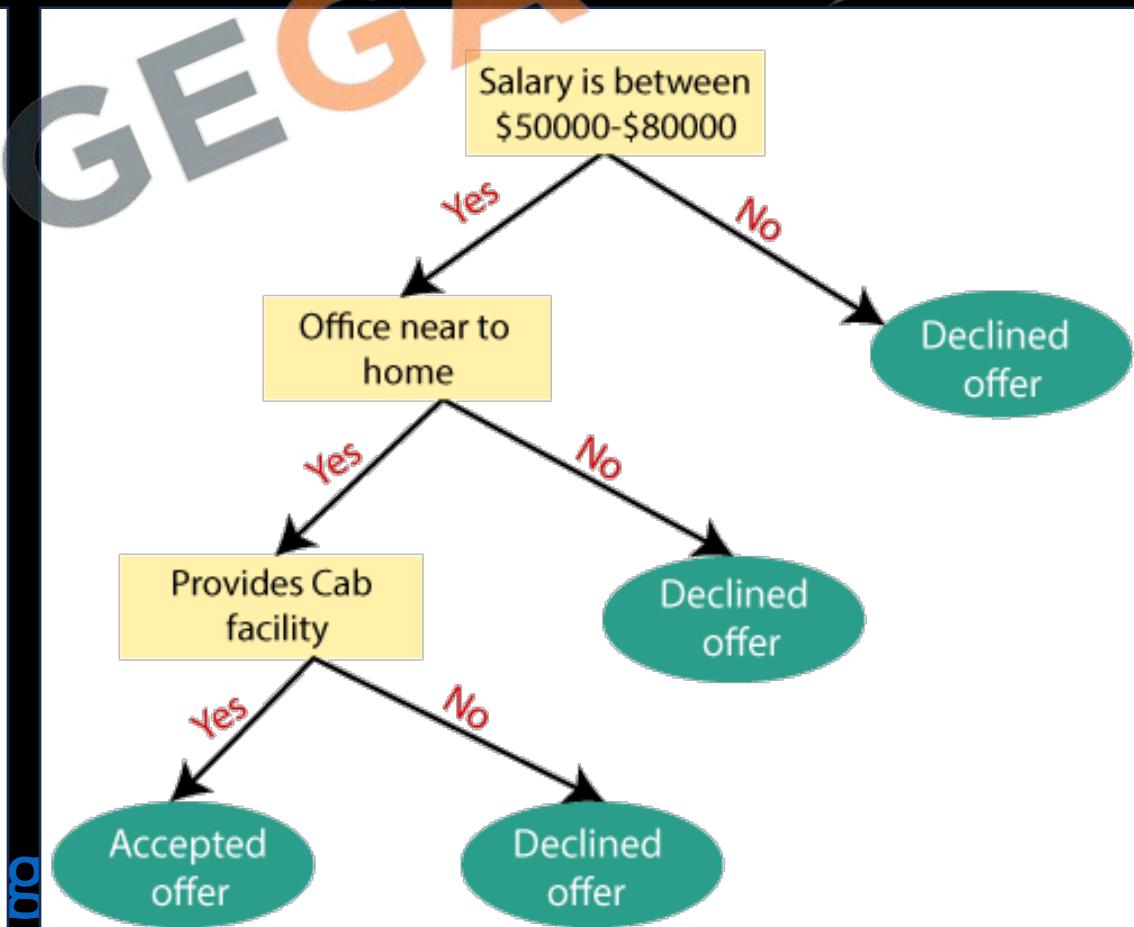
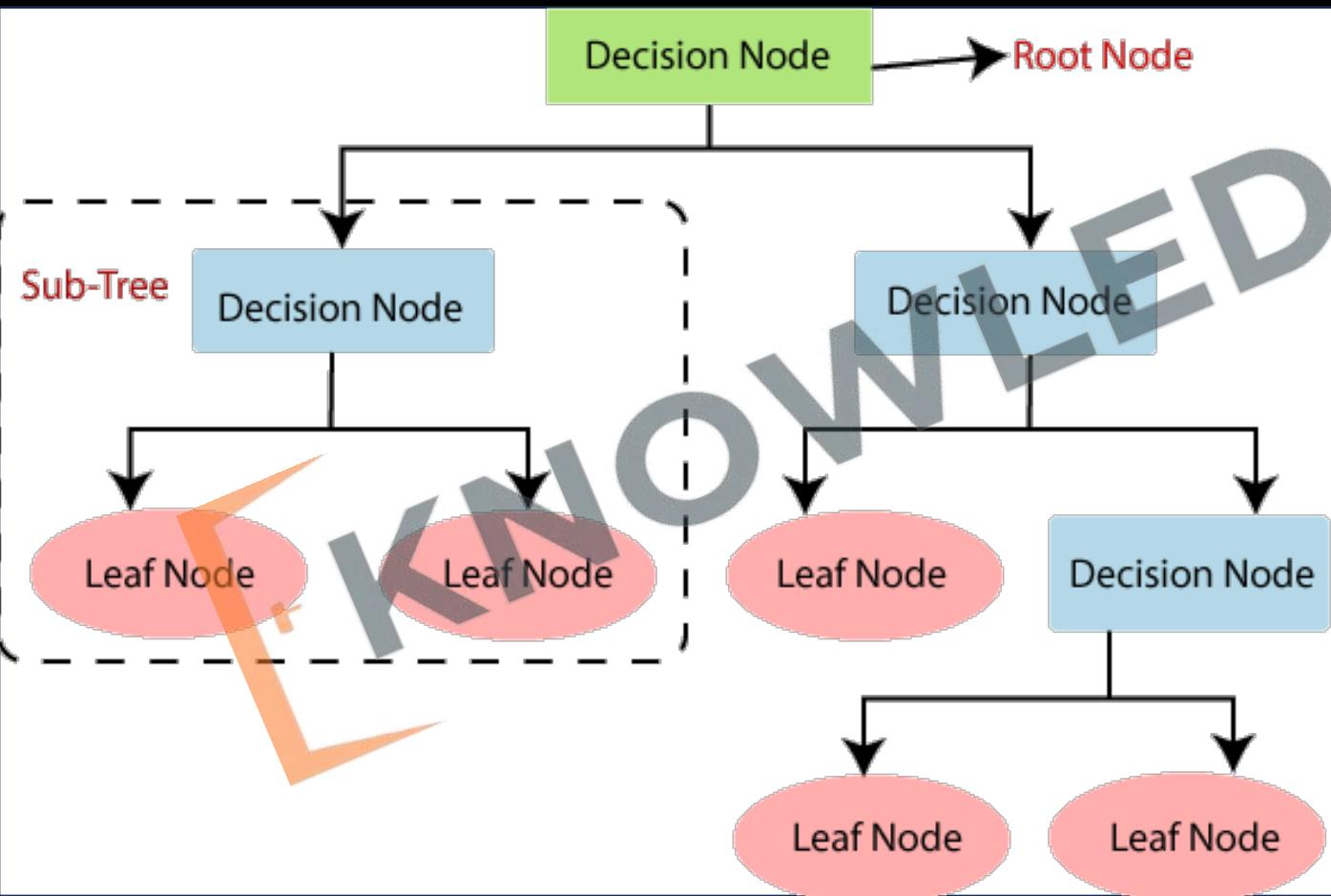


- **Spectral Clustering:**
 - Uses the eigenvalues of a similarity matrix to reduce dimensionality before clustering in fewer dimensions. This technique is particularly useful when the clusters have a complex shape, unlike centroid-based clustering which assumes spherical clusters. For example, in social network analysis, spectral clustering can help identify communities based on the patterns of relationships between members.

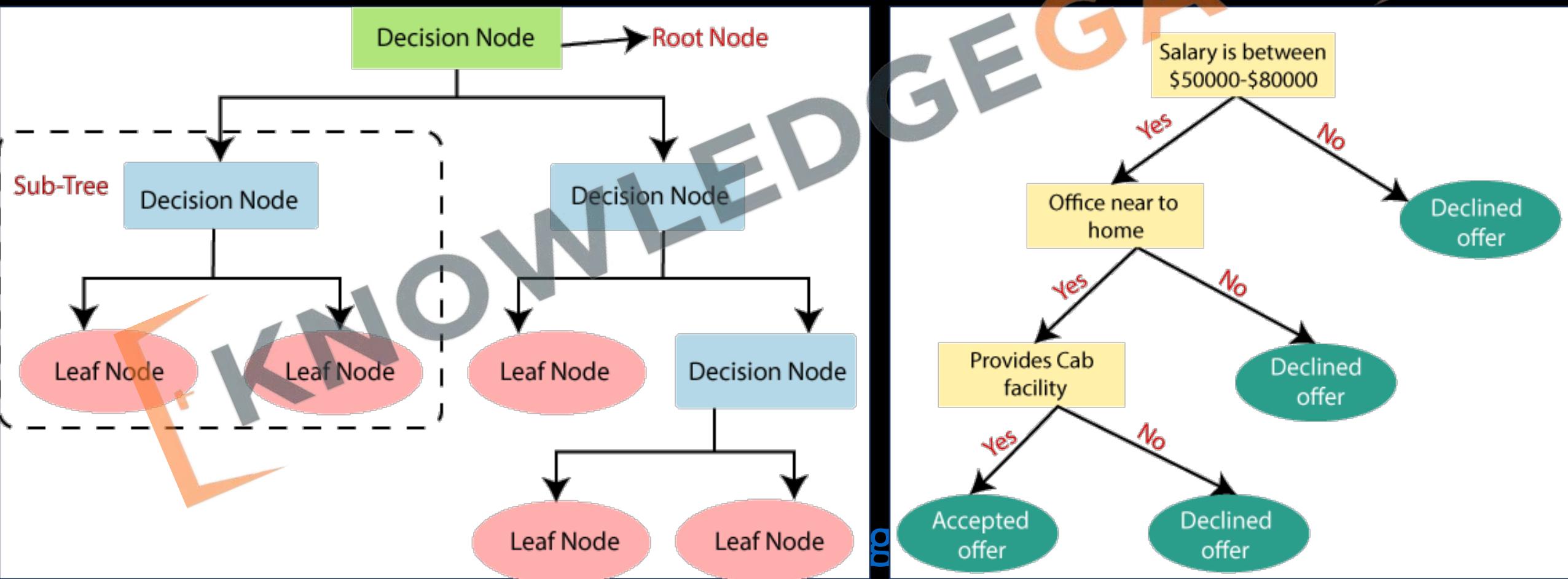


Decision Tree

- A decision tree is a model used in data mining, statistics, and machine learning to predict an outcome based on input variables. It resembles a tree structure with branches and leaves, where each internal node represents a "decision" based on a feature, each branch represents the outcome of that decision, and each leaf node represents the final outcome or class label.

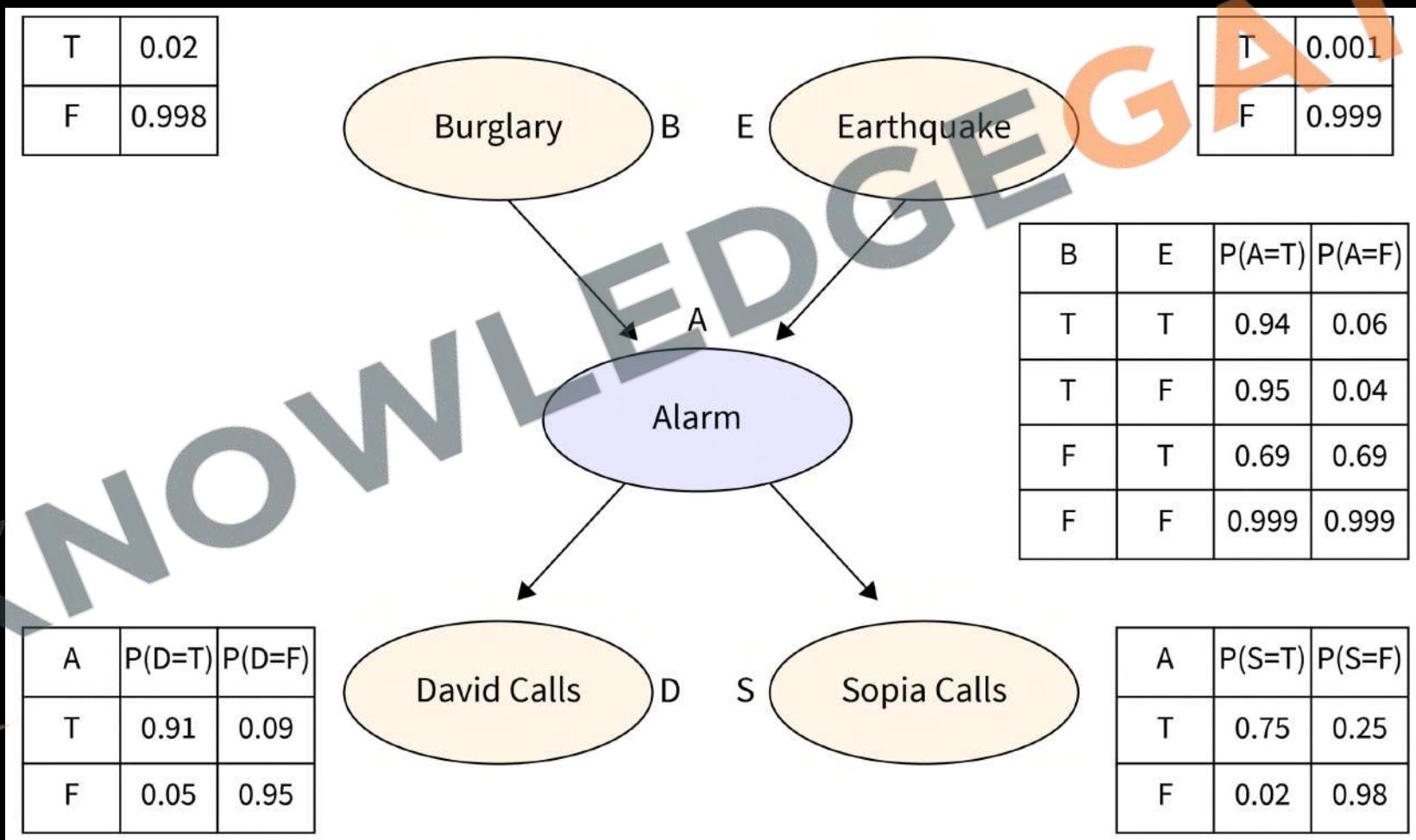


- Advantages and Limitations:
 - Advantages:
 - Easy to interpret and visualize.
 - Requires little data preparation compared to other algorithms.
 - Can handle both numerical and categorical data.
 - Limitations:
 - Prone to overfitting, especially with many branches.
 - Can be biased towards features with more levels.
 - Decisions are based on heuristics, hence might not provide the best split in some cases.

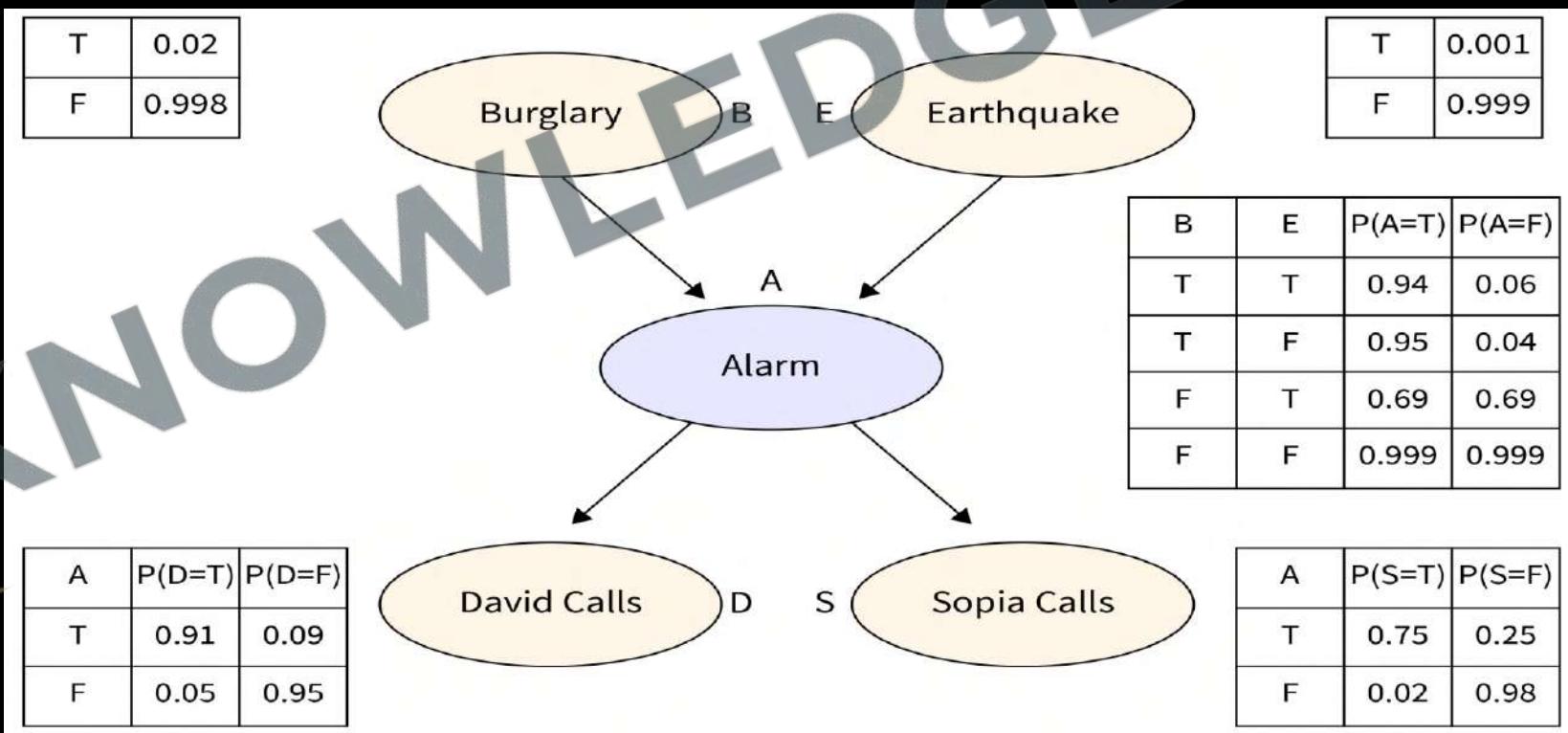


Bayesian belief networks

- Are tools for representing and reasoning under conditions of uncertainty. They capture the probabilistic relationships among a set of variables and allow for the inference of probabilities even with partial information.

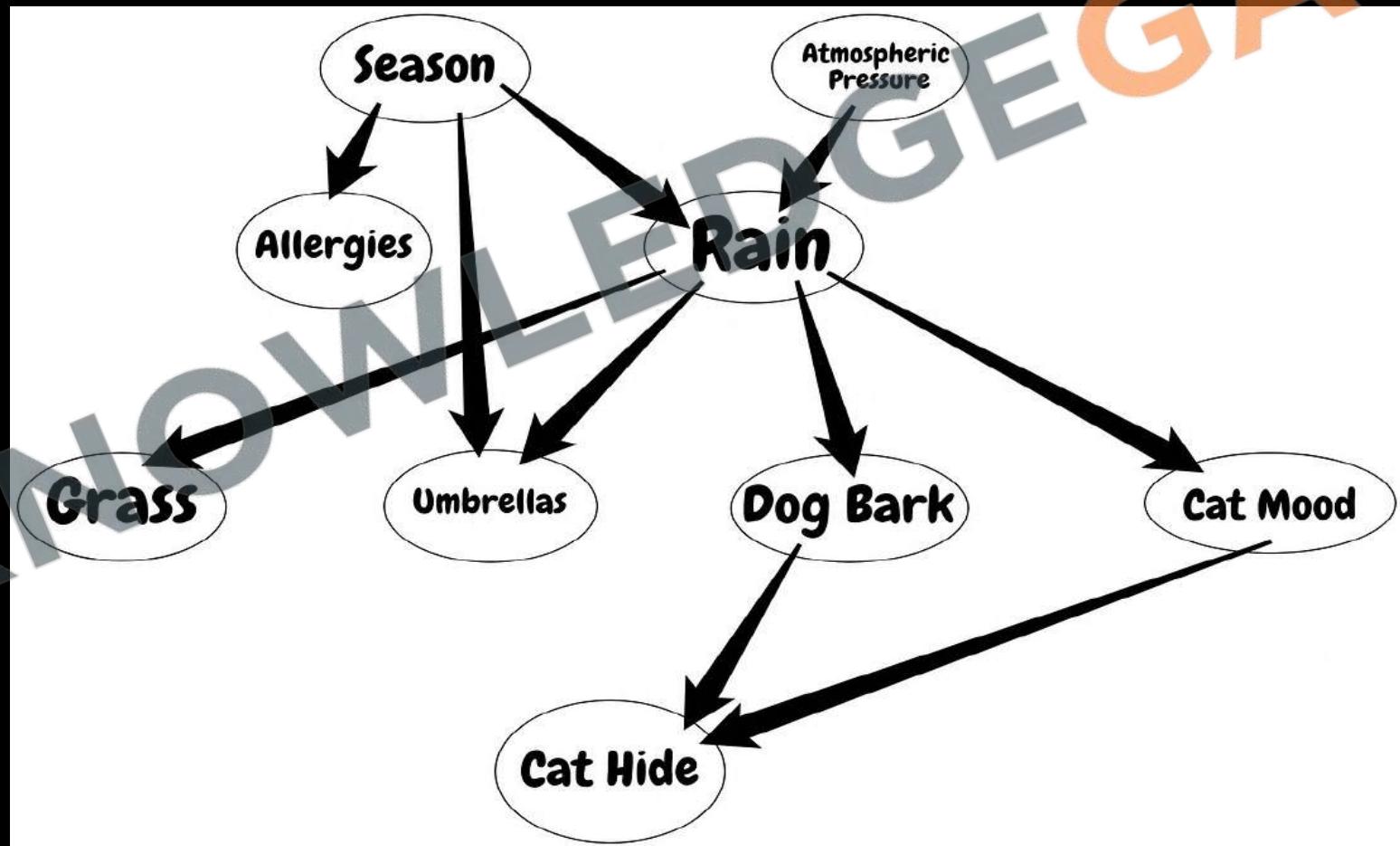


- **Structure:** The core components of a Bayesian belief network include:
 - **Directed Acyclic Graph (DAG):** Each node in the graph represents a random variable, which can be either discrete or continuous. These variables often correspond to attributes in data. Arrows or arcs between nodes represent causal influences.
 - **Conditional Probability Tables (CPTs):** Each node has an associated table that quantifies the effect of the parents on the node.



- Usage:

- Learning: Bayesian networks can be trained using data to learn the conditional dependencies.
- Inference: Once trained, the network can be used for inference, such as predicting the likelihood of lung cancer given that a patient is a smoker with no family history.
- Classification: Bayesian networks can classify new cases based on learned probabilities.



Reinforcement learning

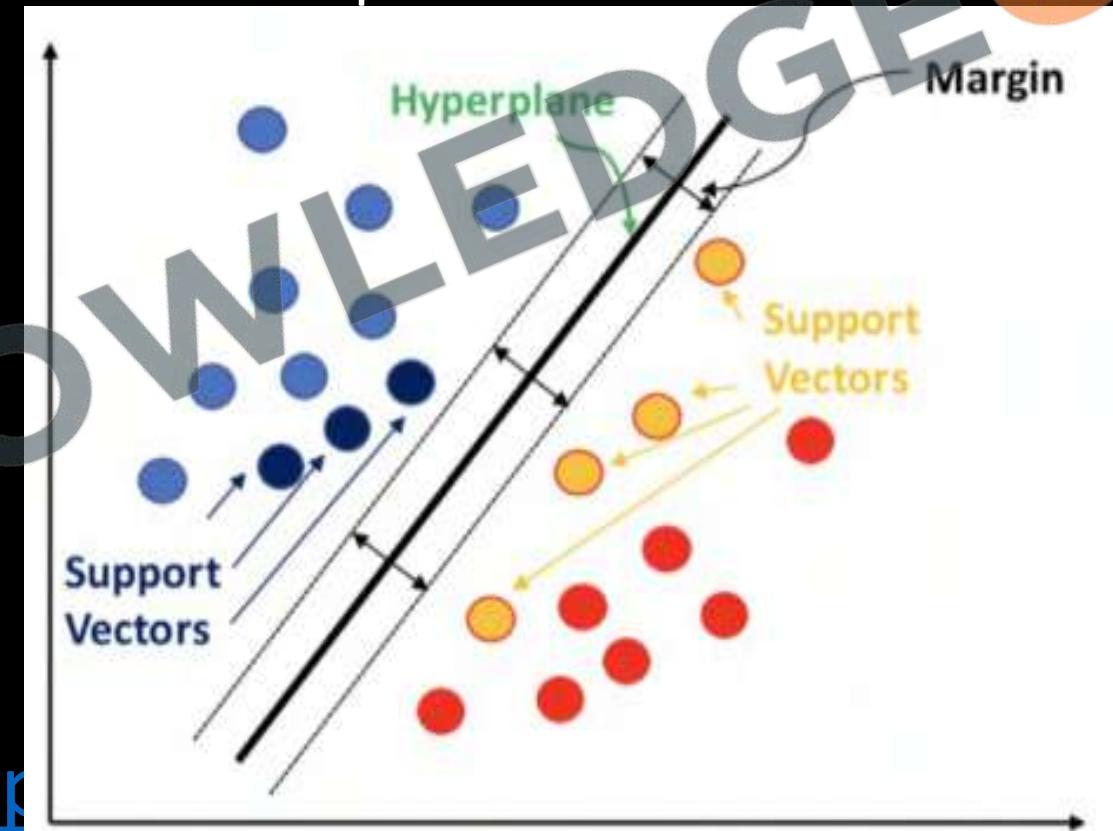
- Reinforcement learning is a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback in the form of rewards or penalties. This method is similar to how individuals learn from the consequences of their actions in real life.
- Key Concepts in Reinforcement Learning:
 - Environment: The world in which the agent operates.
 - State: The current situation of the agent.
 - Actions: What the agent can do.
 - Rewards: Feedback from the environment which can be positive (reinforcements) or negative (punishments).

- Imagine a robot navigating a maze. The robot has to find the shortest path to a destination without prior knowledge of the layout. Each step it takes provides new information:
- If it moves closer to the destination, it receives a positive reward.
- If it hits a wall or moves away from the goal, it receives a negative reward. Through trial and error, the robot learns the optimal path by maximizing its cumulative rewards.



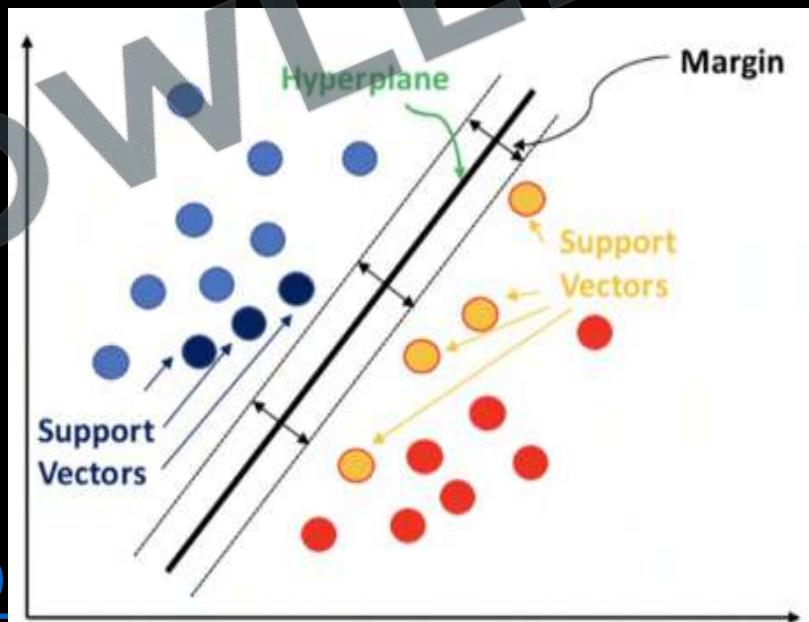
Support Vector Machine

- A Support Vector Machine (SVM) is a powerful machine most commonly used in classification problems.
- SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification. The goal is to find the best hyperplane that has the largest distance to the nearest training data points of any class (functional margin), in order to improve the classification performance on unseen data.



- Applications of SVM:

- Text and Hypertext Classification: For filtering spam and categorizing text based content for news articles.
- Image Classification: Useful in categorizing images into different groups (e.g., animals, cars, fruits).
- Handwritten Character Recognition: Used to recognize letters and digits from handwritten documents.
- Biological Sciences: Applied in protein classification and cancer classification based on gene expression data.



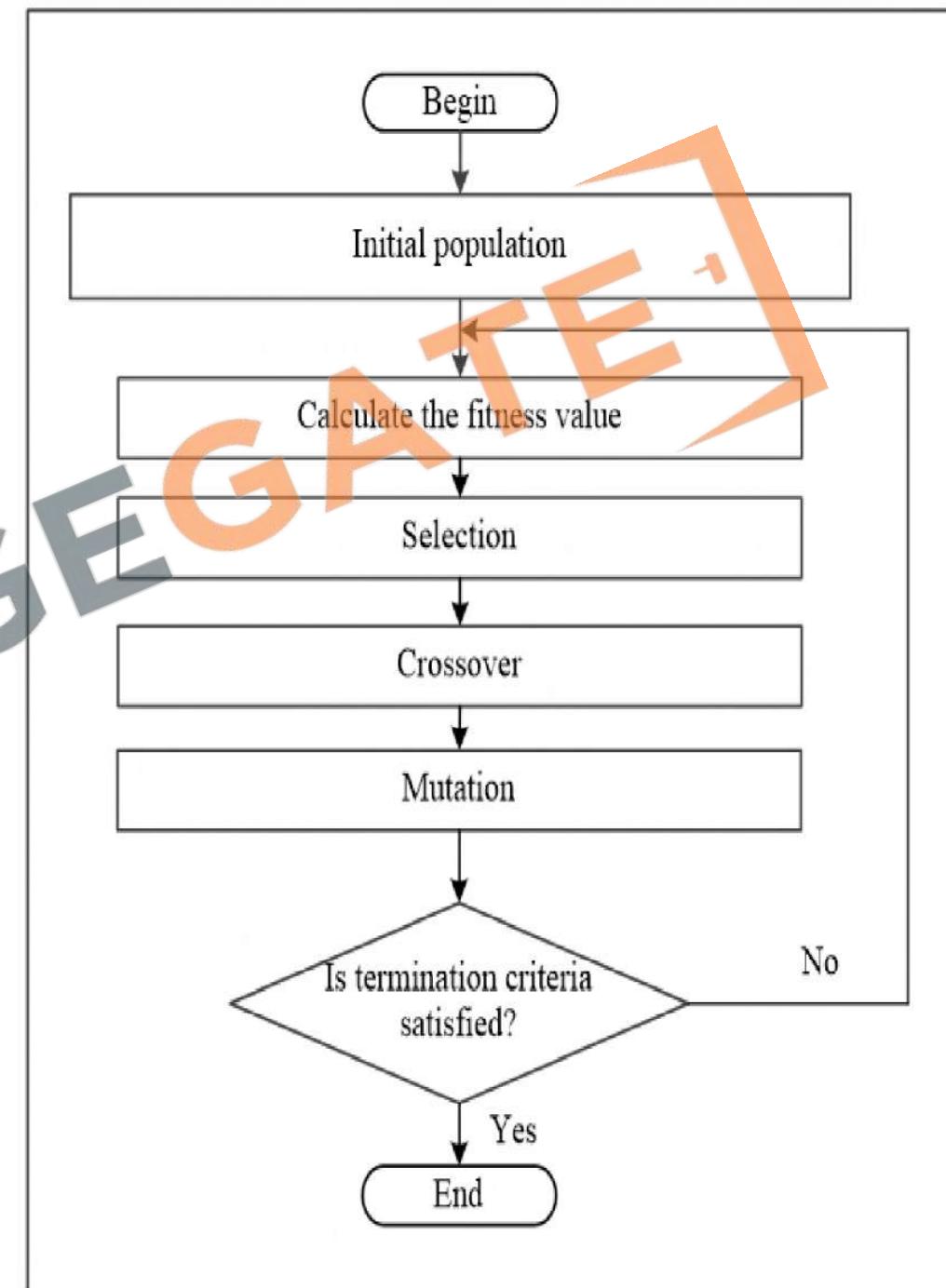
Genetic Algorithm

- A genetic algorithm (GA) is a search heuristic inspired by Charles Darwin's theory of natural selection. It is used to find optimal or near-optimal solutions to complex problems which might otherwise take a long time to solve.
- Overview of Genetic Algorithm:
 - Purpose: Genetic algorithms are used to solve optimization and search problems by mimicking the process of natural selection.
 - Process: This involves a population of individuals which evolve towards a better solution by combining the characteristics of high-quality individuals.

Flowchart of Genetic Algorithm Process:

1. **Initialize Population:** Start with a randomly generated population of n individuals.
2. **Fitness Evaluation:** Evaluate the fitness of each individual in the population. The fitness score determines how good an individual solution is at solving the problem.
3. **Selection:** Select pairs of individuals (parents) based on their fitness scores. Higher fitness scores generally mean a higher chance of selection.
4. **Crossover (Recombination):** Combine the features of selected parents to create offspring. This simulates sexual reproduction.
5. **Mutation:** Introduce random changes to individual offspring to maintain genetic diversity within the population.
6. **Replacement:** Replace the older generation with the new generation of offspring.
7. **Termination:** Repeat the process until a maximum number of generations is reached or a satisfactory fitness level is achieved.

<http://www.knowledge>



Example of Genetic Algorithm: Imagine we want to optimize the design of an aerodynamic car. The objective is to minimize air resistance, which directly impacts fuel efficiency.

- **Encoding:** Each car design is encoded as a string of numbers (genes), representing different design parameters like shape, size, and materials.
- **Initial Population:** Generate a random set of car designs.
- **Fitness Evaluation:** Use a simulation to calculate the air resistance of each design.
- **Selection:** Choose designs with the lowest air resistance.
- **Crossover:** Create new designs by mixing the features of selected designs.
- **Mutation:** Slightly alter the designs to explore a variety of design possibilities.
- **Repeat:** Continue the process to evolve increasingly efficient designs over multiple generations.

Issues in Machine Learning

- Data Quality:
 - **Importance of Quality:** High-quality data is crucial for developing effective ML models. Poor data can lead to inaccurate predictions and unreliable outcomes.
 - Challenges:
 - **Data Evaluation and Integration:** Ensuring data is clean, well-integrated, and representative. For example, a model trained to recognize faces needs a diverse dataset that reflects various ethnicities, ages, and lighting conditions.
 - **Data Exploration and Governance:** Implementing robust data governance to maintain the integrity and usability of data over time.

- **Transparency:**
 - **Model Explainability:** ML models, especially complex ones like deep neural networks, can act as "black boxes," where it's unclear how decisions are made.
 - **Example:** In a credit scoring model, it's crucial for regulatory and fairness reasons to explain why a loan application was denied, which can be challenging with highly complex ML models.

- **Manpower:**
 - **Skill Requirement:** Effective use of ML requires a combination of skills in data science, software development, and domain expertise.
 - **Bias Avoidance:** Having diverse teams is important to prevent biases in model development.
 - **Example:** An organization implementing an ML solution for *customer service* might need experts in natural language processing, software engineering, and customer interaction to develop a comprehensive tool.

- **Other Issues:**

- **Misapplication of Technology**: ML is not suitable for every problem, and its misuse can lead to wasted resources or poor decisions.
 - **Example**: Employing deep learning for a simple data analysis task, where traditional statistical methods would be more appropriate and less costly.
- **Innovation Misuse**: The hype around new ML techniques can lead to premature adoption without proper understanding or necessity.
 - **Example**: The early overuse of deep learning in situations where simpler models could suffice, like predicting straightforward outcomes from small datasets.
- **Traceability and Reproducibility**: Ensuring that ML experiments are reproducible and that results can be traced back to specific data and configuration settings.
 - **Example**: A research team must be able to replicate an ML experiment's results using the same datasets and parameters to verify findings and ensure reliability.

S. No.	Data Science	Machine Learning
1	Involves data cleansing, preparation, and analysis.	Practice of using algorithms to learn from and make predictions based on data.
2	Deals with a variety of data operations.	A subset of Artificial Intelligence focused on statistical models and algorithms.
3	Focuses on sourcing, cleaning, and processing data to extract meaningful insights.	Programs learn from data and improve autonomously without explicit instructions.
4	Tools include SAS, Tableau, Apache Spark, MATLAB.	Tools include Amazon Lex, IBM Watson Studio, Microsoft Azure ML Studio.
5	Applied in fraud detection, healthcare analysis, and business optimization.	Used in recommendation systems like Spotify, facial recognition technologies.

(UNIT-2: REGRESSION & BAYESIAN LEARNING)

- **REGRESSION**: Linear Regression and Logistic Regression.
- **BAYESIAN LEARNING** - Bayes theorem, Concept learning, Bayes Optimal Classifier, Naïve Bayes classifier, Bayesian belief networks, EM algorithm.
- **SUPPORT VECTOR MACHINE**: Introduction, Types of support vector kernel - (Linear kernel, polynomial kernel, and Gaussian kernel), Hyperplane - (Decision surface), Properties of SVM, and Issues in SVM.

Regression

- Regression is a statistical technique used to analyze the relationship between a dependent variable and one or more independent variables. It is widely used in areas like finance, economics, and more, to predict outcomes and understand variable interactions.



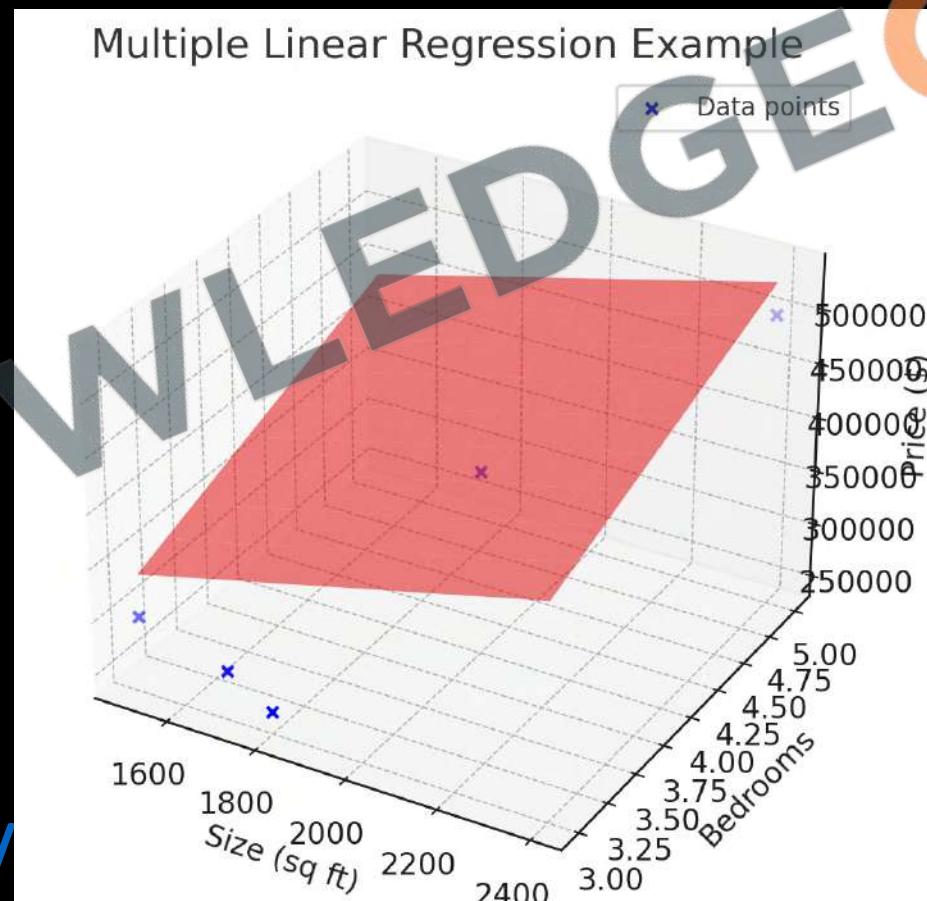
Types of Regression

- **Simple Linear Regression:**

- This method involves one independent variable used to predict the outcome of a dependent variable. The formula is $Y=a+bX+u$, where:
 - Y is the dependent variable we want to predict.
 - X is the independent variable used for prediction.
 - a is the intercept of the regression line (value of Y when X is 0).
 - b is the slope of the regression line, representing the change in Y for a one-unit change in X .
 - u is the regression residual, which is the error in the prediction.
- Example: Predicting house prices (Y) based on house size (X). A larger house size generally increases the house price.

- **Multiple Linear Regression:**

- Involves two or more independent variables to predict the outcome. The formula is $Y=a+b_1X_1+b_2X_2+\dots+b_nXn+u$, where:
 - Each X_i represents a different independent variable.
 - Each b_i is the coefficient for the corresponding independent variable, showing how much Y changes when that variable changes by one unit, holding other variables constant.



Logistic Regression

Definition of Logistic Regression:

- Logistic regression is a statistical method and a type of supervised machine learning algorithm. It is used to estimate the probability that a given input belongs to a certain category (typically a binary outcome).
- Characteristics of the Dependent Variable:
 - The target variable in logistic regression is binary, meaning it has two possible outcomes. These outcomes are usually coded as 1 (indicating success or the presence of a feature, like "yes") and 0 (indicating failure or the absence of a feature, like "no").



- Applications:
 - Logistic regression is widely applied in fields such as medicine, finance, and marketing. It helps in binary classification tasks such as detecting whether an email is spam or not, predicting whether a patient has a disease like diabetes, or determining if a transaction might be fraudulent.
- Example:
 - **Predicting Disease Occurrence:** Suppose a medical researcher wants to predict the likelihood that individuals have diabetes based on their age and BMI. Here, the outcome variable Y is whether the person has diabetes (1) or not (0), and the predictors X_1 and X_2 are age and BMI, respectively. The logistic regression model would help estimate the probability of diabetes for different age groups and BMI levels, using historical data to determine the coefficients b_1 and b_2 for age and BMI.

Aspect	Linear Regression	Logistic Regression
Type of Model	Supervised regression model	Supervised classification model
Prediction Outcome	Predicts continuous values	Predicts binary outcomes (0 or 1)
Mathematical Model	Uses linear functions	Uses logistic functions with an activation function
Purpose	Estimates values of a dependent variable	Estimates probability of an event
Example	Predicting house prices based on size	Predicting whether a patient has a disease or not

Bayes' Theorem describes the probability of an event based on prior knowledge of conditions related to the event. It is expressed mathematically as:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

1 2 3 4 5 6 7 8 9 10

Where:

- $P(A|B)$ is the probability of event A occurring given that B is true.
- $P(B|A)$ is the probability of event B occurring given that A is true.
- $P(A)$ is the probability of event A occurring.
- $P(B)$ is the probability of event B occurring.

2,3,5,7 357

$$P(o) = P(p) \cdot P(o|p) + P(\sim p) \cdot P(o|\sim p)$$

$$= \frac{4}{10} \times \frac{3}{4} + \frac{6}{10} \times \frac{1}{4}$$

$$= \frac{5}{10} = .5$$

$$P(p|o) = \frac{P(p) \cdot P(o|p)}{P(o)}$$

<http://www.knowledgegate.in/gate>

Concept learning

- **Concept learning** is the process of inferring a function from labeled training data in supervised learning. It involves identifying patterns or rules that correctly classify instances into predefined categories, using methods like decision trees or neural networks to search through possible hypotheses and select the best one.

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- The given table represents a dataset where Tom's enjoyment of his favorite water sports is recorded based on various weather conditions. The goal is to determine under what conditions Tom enjoys water sport

Attributes and Their Values

- Sky:** Sunny, Rainy
- AirTemp:** Warm, Cold
- Humidity:** Normal, High
- Wind:** Strong
- Water:** Warm, Cool
- Forecast:** Same, Change
- EnjoySport:** Indicates whether Tom enjoys water sports under these conditions. Possible values: Yes, No

1. Initialization:

- Start with the most specific hypothesis, which initially does not match any instances.
- Example: $h = (\text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same})$

2. Generalization:

- Update the hypothesis to generalize it as more examples are considered.
- If the example is positive ($\text{EnjoySport} = \text{Yes}$), generalize the hypothesis to match the example.

3. Updating Hypothesis:

- For each positive example, update the hypothesis to the least general generalization that still matches the example.
- For each negative example, ensure the hypothesis does not match the negative example.

1. Initial Hypothesis:

- $h = (\text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same})$

2. First Example (Sunny, Warm, Normal, Strong, Warm, Same, Yes):

- Hypothesis remains the same because it matches the example.
- $h = (\text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same})$

3. Second Example (Sunny, Warm, High, Strong, Warm, Same, Yes):

- Generalize the hypothesis to match both examples.
- $h = (\text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same})$
- '?' means any value is acceptable for Humidity.

4. Third Example (Rainy, Cold, High, Strong, Warm, Change, No):

- Negative example, so no change to the hypothesis.

5. Fourth Example (Sunny, Warm, High, Strong, Cool, Change, Yes):

- Generalize the hypothesis further to include this example.
- $h = (\text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ?)$

Final Hypothesis

The final hypothesis represents the conditions under which Tom enjoys water sports:

- (Sunny, Warm, ?, Strong, ?, ?)

This means Tom enjoys water sports when:

- The sky is Sunny,
- The air temperature is Warm,
- The wind is Strong,
- Humidity, water temperature, and forecast can be any value.

- **Advantages:**
 - Generalization: Learns broad rules from specific examples.
 - Interpretability: Produces human-readable rules or models.
- **Disadvantages:**
 - Overfitting: Risk of overly complex models.
 - Requires Labeled Data: Needs a lot of labeled examples.
- **Applications:**
 - Email Spam Detection: Classifies emails as spam or not.
 - Medical Diagnosis: Predicts diseases from patient data.

Bayes Optimal Classifier

- The Bayes Optimal Classifier is a theoretical model in machine learning that makes predictions based on the highest posterior probability. It uses Bayes' theorem to combine prior knowledge with observed data to make the most accurate possible predictions.

Example: Spam Email Classification

Suppose we want to classify an email as "spam" or "not spam" based on certain features. Let's consider the features "contains the word 'offer'" and "contains the word 'win'".

Given Data:

- Prior Probabilities:
 - $P(\text{spam}) = 0.4$
 - $P(\text{not spam}) = 0.6$
- Likelihoods:
 - $P(\text{contains 'offer'}|\text{spam}) = 0.8$
 - $P(\text{contains 'win'}|\text{spam}) = 0.7$
 - $P(\text{contains 'offer'}|\text{not spam}) = 0.2$
 - $P(\text{contains 'win'}|\text{not spam}) = 0.1$
- Feature Observation:
 - The email contains both the words "offer" and "win".

Calculate Posterior Probabilities

1. Calculate the joint likelihoods:

- For spam: $P(x|\text{spam}) = P(\text{contains 'offer'}|\text{spam}) \cdot P(\text{contains 'win'}|\text{spam})$

$$P(x|\text{spam}) = 0.8 \cdot 0.7 = 0.56$$

- For not spam: $P(x|\text{not spam}) = P(\text{contains 'offer'}|\text{not spam}) \cdot P(\text{contains 'win'}|\text{not spam})$

$$P(x|\text{not spam}) = 0.2 \cdot 0.1 = 0.02$$

2. Calculate the evidence:

- $P(x) = P(x|\text{spam}) \cdot P(\text{spam}) + P(x|\text{not spam}) \cdot P(\text{not spam})$

$$P(x) = (0.56 \cdot 0.4) + (0.02 \cdot 0.6) = 0.224 + 0.012 = 0.236$$

3. Calculate the posterior probabilities using Bayes' theorem:

- For spam:

$$P(\text{spam}|x) = \frac{P(x|\text{spam}) \cdot P(\text{spam})}{P(x)} = \frac{0.56 \cdot 0.4}{0.236} \approx 0.949$$

- For not spam:

$$P(\text{not spam}|x) = \frac{P(x|\text{not spam}) \cdot P(\text{not spam})}{P(x)} = \frac{0.02 \cdot 0.6}{0.236} \approx 0.051$$

Since $P(\text{spam}|x) \approx 0.949$ and $P(\text{not spam}|x) \approx 0.051$, the Bayes Optimal Classifier will classify this email as "spam" because the posterior probability for "spam" is higher.

$x = \text{offer} \wedge \text{win}$

$$\begin{aligned} & P(\text{spam}) \quad P(x|\text{spam}) \\ & 4 \quad 56 \\ & \overline{4 \times 56 = 224} \\ & P(\text{not spam}) \quad P(x|\text{not spam}) \\ & 6 \quad 0.2 \\ & \overline{6 \times 0.2 = 0.12} \\ & 224 + 0.12 \\ & \overline{= 0.236} \\ & P(\text{spam}) \quad P(x|\text{not spam}) \\ & 4 \quad 0.2 \\ & \overline{4 \times 0.2 = 0.8} \\ & P(\text{not spam}) \quad P(x|\text{spam}) \\ & 6 \quad 56 \\ & \overline{6 \times 56 = 336} \end{aligned}$$

Posterior Probability Calculation:

- Uses Bayes' theorem to calculate the posterior probability of each class given the data.
- Formula: $P(C_i|x) = \frac{P(x|C_i) \cdot P(C_i)}{P(x)}$
 - $P(C_i|x)$: Posterior probability of class C_i given instance x
 - $P(x|C_i)$: Likelihood of instance x given class C_i
 - $P(C_i)$: Prior probability of class C_i
 - $P(x)$: Probability of instance x

Classification:

- Assigns the class label with the highest posterior probability to the instance.
- $C^* = \arg \max_{C_i} P(C_i|x)$

- **Advantages**
 - **Optimal Predictions**: Provides the most accurate predictions theoretically possible by minimizing the probability of misclassification.
 - **Incorporates All Information**: Uses all available data and prior knowledge.
- **Disadvantages**
 - **Computationally Intensive**: Often impractical to compute for real-world applications due to the need for exact probability distributions.
 - **Requires Accurate Probabilities**: Performance depends on the accuracy of the estimated probabilities.
- **Applications**
 - **Theoretical Benchmark**: Serves as an ideal performance benchmark for other classifiers.
 - **Ensemble Methods**: Used in ensemble learning techniques like Bayesian averaging to improve classification performance.

Naive Bayes Classifier

- The Naive Bayes classifier is a simple and effective probabilistic classifier based on Bayes' Theorem. It assumes that the features are conditionally independent given the class label, which is often not true in practice but simplifies the computation significantly.

<http://www.knowledgegate.in/gate>

Bayes' Theorem for Class "Yes" and "No" with Single and Multiple Features

1. Bayes' Theorem for Single Feature X and Class "Yes" (Y):

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

2. Bayes' Theorem for Multiple Features X_1, X_2, \dots, X_n and Class "Yes" (Y):

$$P(Y|X_1, X_2, \dots, X_n) = \frac{P(X_1|Y) \cdot P(X_2|Y) \cdot P(X_3|Y) \cdots P(X_n|Y) \cdot P(Y)}{P(X_1) \cdot P(X_2) \cdot P(X_3) \cdots P(X_n)}$$

3. Bayes' Theorem for Single Feature X and Class "No" (N):

$$P(N|X) = \frac{P(X|N) \cdot P(N)}{P(X)}$$

4. Bayes' Theorem for Multiple Features X_1, X_2, \dots, X_n and Class "No" (N):

$$P(N|X_1, X_2, \dots, X_n) = \frac{P(X_1|N) \cdot P(X_2|N) \cdot P(X_3|N) \cdots P(X_n|N) \cdot P(N)}{P(X_1) \cdot P(X_2) \cdot P(X_3) \cdots P(X_n)}$$

- Overweight Due to Eating Habits or Medical Issues

Person	Eating Habits (Good/Bad)	Medical Issues (Yes/No)	Overweight (Yes/No)
1	Bad	No	Yes
2	Bad	No	No
3	Bad	Yes	Yes
4	Good	No	No
5	Good	Yes	Yes
6	Good	No	No
7	Bad	No	Yes
8	Good	Yes	Yes
9	Bad	No	No
10	Good	Yes	No

- Overweight Due to Eating Habits or Medical Issues

Step 1: Prior Probability

- $P(\text{overweight} = \text{yes}) = \frac{5}{10} = 0.5$
- $P(\text{overweight} = \text{no}) = \frac{5}{10} = 0.5$

Step 2: Conditional Probability

	Overweight = Yes	Overweight = No
Eating Habits (Bad)	$\frac{3}{5}$	$\frac{2}{5}$
Eating Habits (Good)	$\frac{2}{5}$	$\frac{3}{5}$
Medical Issues (Yes)	$\frac{2}{5}$	$\frac{1}{5}$
Medical Issues (No)	$\frac{3}{5}$	$\frac{4}{5}$

Person	Eating Habits (Good/Bad)	Medical Issues (Yes/No)	Overweight (Yes/No)
1	Bad	No	Yes
2	Bad	No	No
3	Bad	Yes	Yes
4	Good	No	No
5	Good	Yes	Yes
6	Good	No	No
7	Bad	No	Yes
8	Good	Yes	Yes
9	Bad	No	No
10	Good	Yes	No

Bayes' Theorem for Class "Yes" and "No" with Single and Multiple Features

1. Bayes' Theorem for Single Feature X and Class "Yes" (Y):

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

$$\begin{aligned} & P(\text{OW}/\text{BE, ME}) \\ &= \underbrace{P(\text{BE}/\text{OW})}_{P(\text{BE})} \underbrace{P(\text{ME}/\text{OW})}_{P(\text{ME})} P(\text{OW}) \end{aligned}$$

2. Bayes' Theorem for Multiple Features X_1, X_2, \dots, X_n and Class "Yes" (Y):

$$P(Y|X_1, X_2, \dots, X_n) = \frac{P(X_1|Y) \cdot P(X_2|Y) \cdot P(X_3|Y) \cdots P(X_n|Y) \cdot P(Y)}{P(X_1) \cdot P(X_2) \cdot P(X_3) \cdots P(X_n)}$$

3. Bayes' Theorem for Single Feature X and Class "No" (N):

$$\begin{aligned} P(N|X) &= \frac{P(X|N) \cdot P(N)}{P(X)} \\ &= \underbrace{P(\text{BE}/\text{NOW})}_{P(\text{BE})} \underbrace{P(\text{ME}/\text{NOW})}_{P(\text{ME})} P(\text{NOW}) \end{aligned}$$

4. Bayes' Theorem for Multiple Features X_1, X_2, \dots, X_n and Class "No" (N):

$$P(N|X_1, X_2, \dots, X_n) = \frac{P(X_1|N) \cdot P(X_2|N) \cdot P(X_3|N) \cdots P(X_n|N) \cdot P(N)}{P(X_1) \cdot P(X_2) \cdot P(X_3) \cdots P(X_n)}$$

Step 3: Posterior Probability Calculation

Given a person with bad eating habits and medical issues, we want to calculate:

1. Probability of being overweight given bad eating habits and medical issues:

$$P(\text{overweight} = \text{yes} | \text{bad eating habits, medical issues}) = P(\text{bad eating habits|yes}) \cdot P(\text{medical issues|yes}) \cdot P(\text{yes})$$

$$P(\text{overweight} = \text{yes} | \text{bad eating habits, medical issues}) = \frac{3}{5} \cdot \frac{2}{5} \cdot 0.5 = 0.3 \cdot 0.4 = 0.12$$

2. Probability of not being overweight given bad eating habits and medical issues:

$$P(\text{overweight} = \text{no} | \text{bad eating habits, medical issues}) = P(\text{bad eating habits|no}) \cdot P(\text{medical issues|no}) \cdot P(\text{no})$$

$$P(\text{overweight} = \text{no} | \text{bad eating habits, medical issues}) = \frac{2}{5} \cdot \frac{1}{5} \cdot 0.5 = 0.2 \cdot 0.2 = 0.02$$

Conclusion

Given the person with bad eating habits and medical issues, we compare the posterior probabilities:

- $P(\text{overweight} = \text{yes} | \text{bad eating habits, medical issues}) = 0.12$
- $P(\text{overweight} = \text{no} | \text{bad eating habits, medical issues}) = 0.02$

Since $0.12 > 0.02$, the person is more likely to be overweight given bad eating habits and medical issues.

Bayes' Theorem

Bayes' Theorem for a given class C_i and feature vector x is:

$$P(C_i|x) = \frac{P(x|C_i) \cdot P(C_i)}{P(x)}$$

Naive Bayes Assumption

The naive assumption of feature independence means that the joint probability $P(x|C_i)$ can be decomposed into the product of individual probabilities:

$$P(x|C_i) = \prod_{j=1}^n P(x_j|C_i)$$

Thus, the classifier can be written as:

$$P(C_i|x) \propto P(C_i) \cdot \prod_{j=1}^n P(x_j|C_i)$$

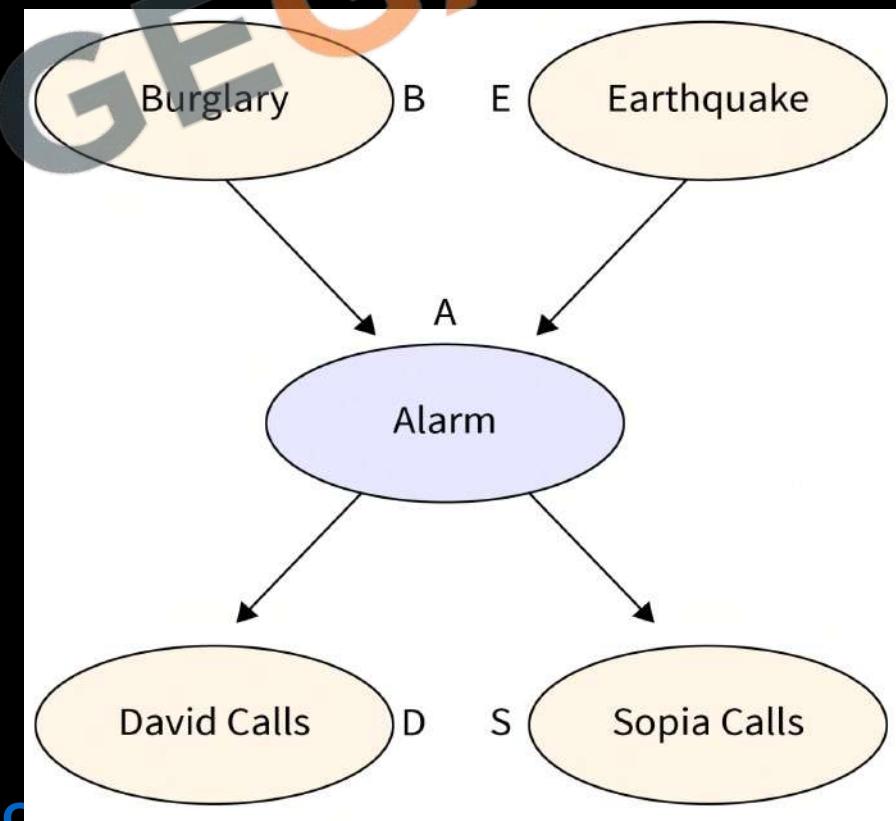
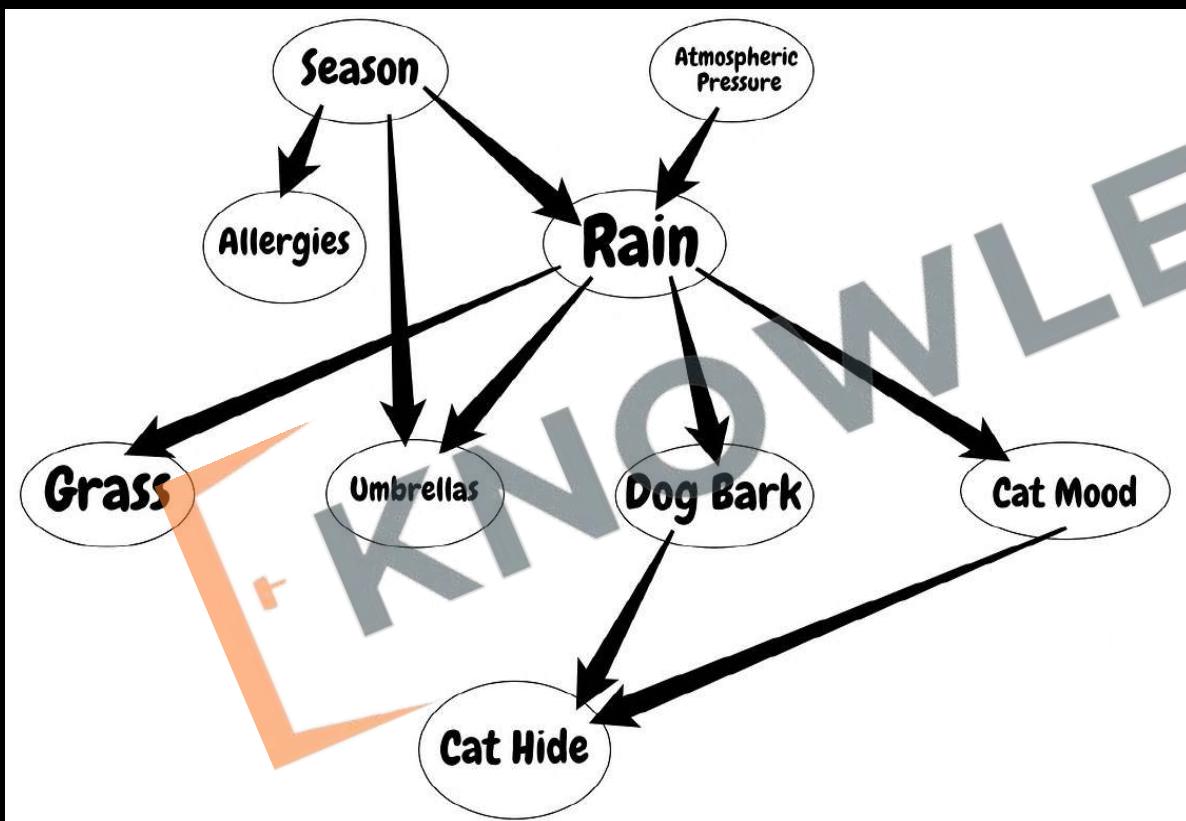
Classification Rule

The Naive Bayes classifier assigns a class label \hat{C} to a given instance x by choosing the class with the highest posterior probability:

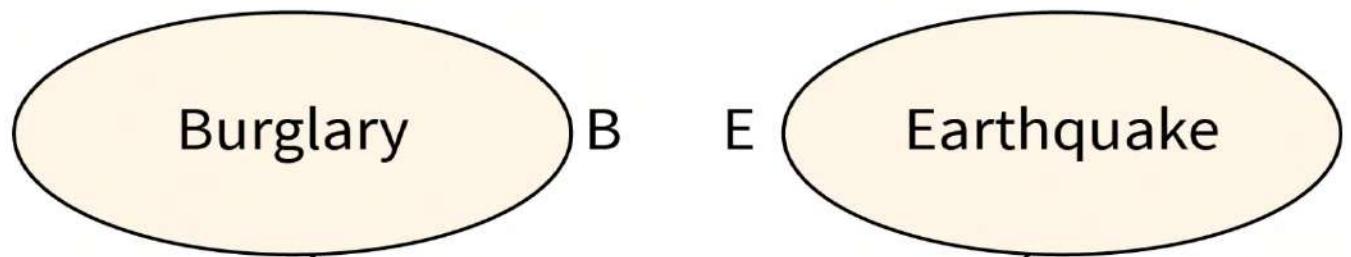
$$\hat{C} = \arg \max_{C_i} P(C_i) \cdot \prod_{j=1}^n P(x_j|C_i)$$

Bayesian Belief Network (BBN)

- A Bayesian Belief Network (BBN), also known as a Bayesian Network or Belief Network, is a graphical model that represents a set of variables and their conditional dependencies using a directed acyclic graph (DAG). Each node in the network represents a variable, and each edge represents a conditional dependency between variables.



T	0.02
F	0.998



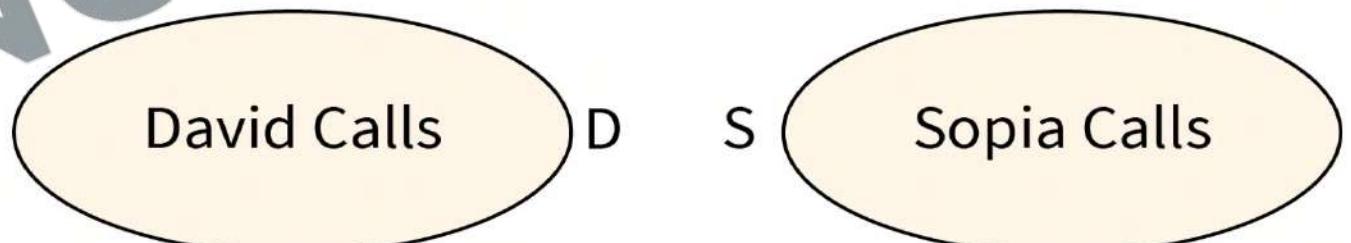
T	0.001
F	0.999

KNOWLEDGE GATE

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.79	0.21
F	F	0.001	0.999

KNOWLEDGE GATE

A	P(D=T)	P(D=F)
T	0.91	0.09
F	0.05	0.95



A	P(S=T)	P(S=F)
T	0.75	0.25
F	0.02	0.98

Key Components:

1. Nodes:

- Represent random variables.
- Example: Variables like "Smoking", "Cancer", "Coughing".

2. Edges:

- Directed edges represent conditional dependencies.
- Example: An edge from "Smoking" to "Cancer" indicates that smoking affects the probability of cancer.

3. Conditional Probability Tables (CPTs):

- Each node has a CPT that quantifies the effect of the parent nodes.
- Example: The CPT for "Cancer" might show the probability of having cancer given whether or not a person smokes.

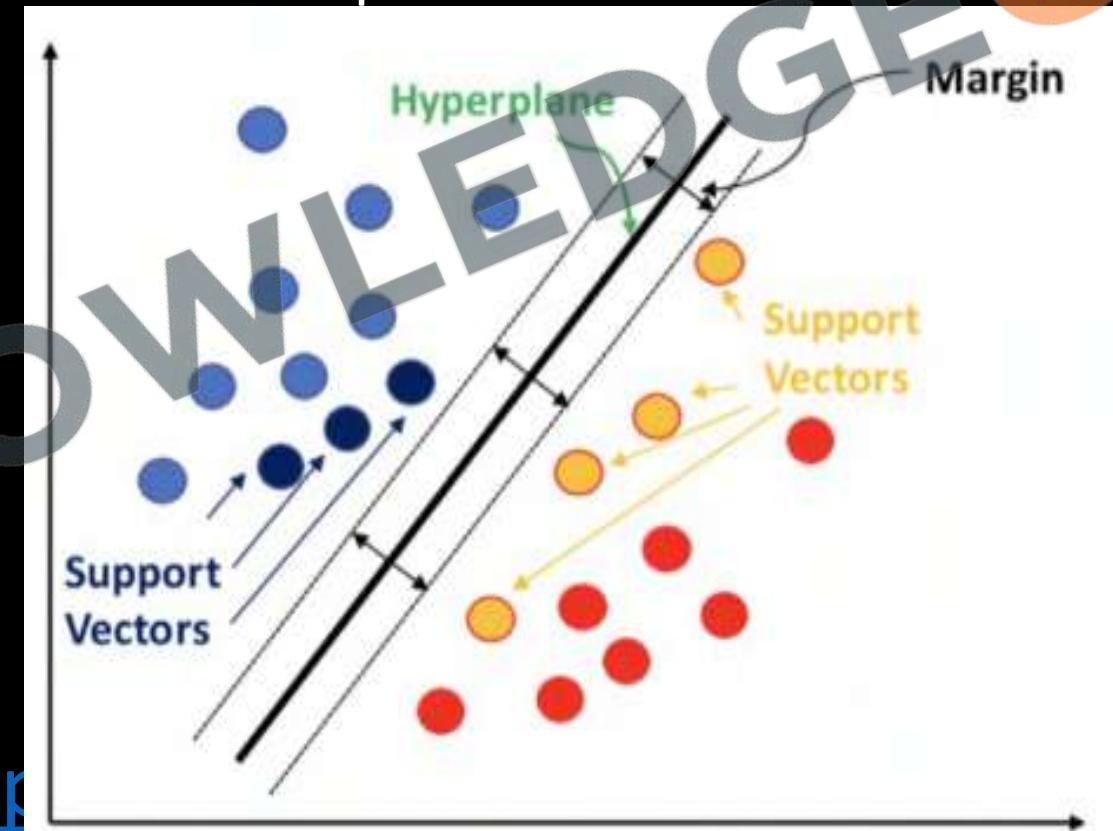
- **Advantages:**
 - **Modular Representation:** Each variable is conditionally independent of its non-descendants given its parents.
 - **Efficient Computation:** Supports efficient inference algorithms for calculating probabilities.
 - **Visualization:** Provides a clear visual representation of dependencies among variables.

- **Disadvantages:**
 - **Complexity:** Can become computationally intensive for large networks.
 - **Data Requirements:** Requires sufficient data to accurately estimate the CPTs.
 - **Design Effort:** Constructing a good network requires expert knowledge and effort.

- **Applications:**
 - **Medical Diagnosis**: Modeling diseases and symptoms to assist in diagnosis.
 - **Risk Assessment**: Evaluating risks in engineering and finance.
 - **Natural Language Processing**: Understanding dependencies between words and sentences.
 - **Decision Support Systems**: Providing recommendations based on probabilistic reasoning.

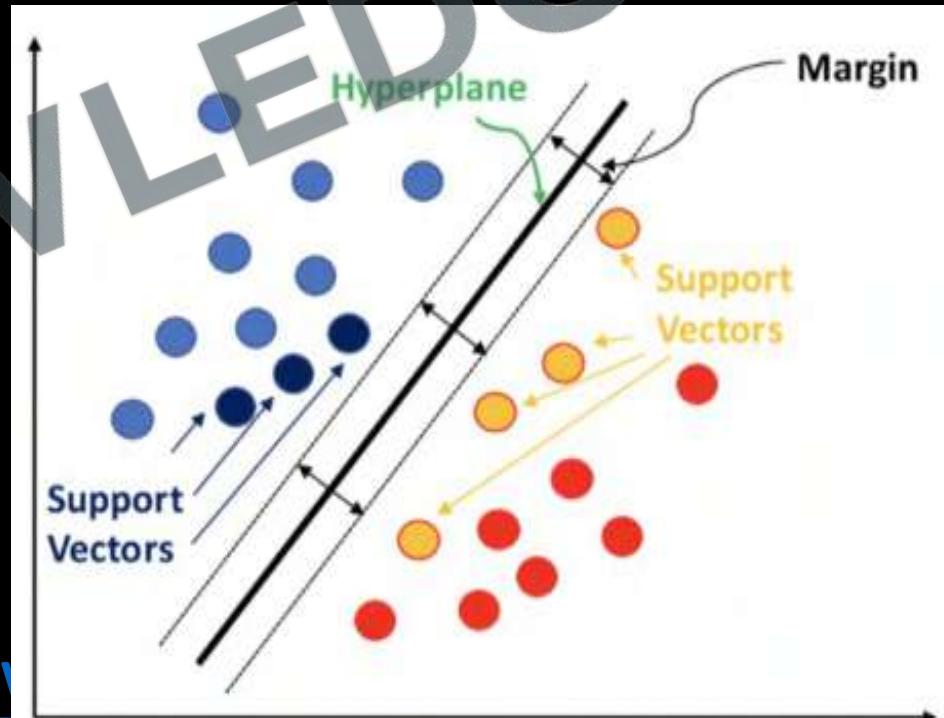
Support Vector Machine

- A Support Vector Machine (SVM) is a powerful machine most commonly used in classification problems.
- SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification. The goal is to find the best hyperplane that has the largest distance to the nearest training data points of any class (functional margin), in order to improve the classification performance on unseen data.



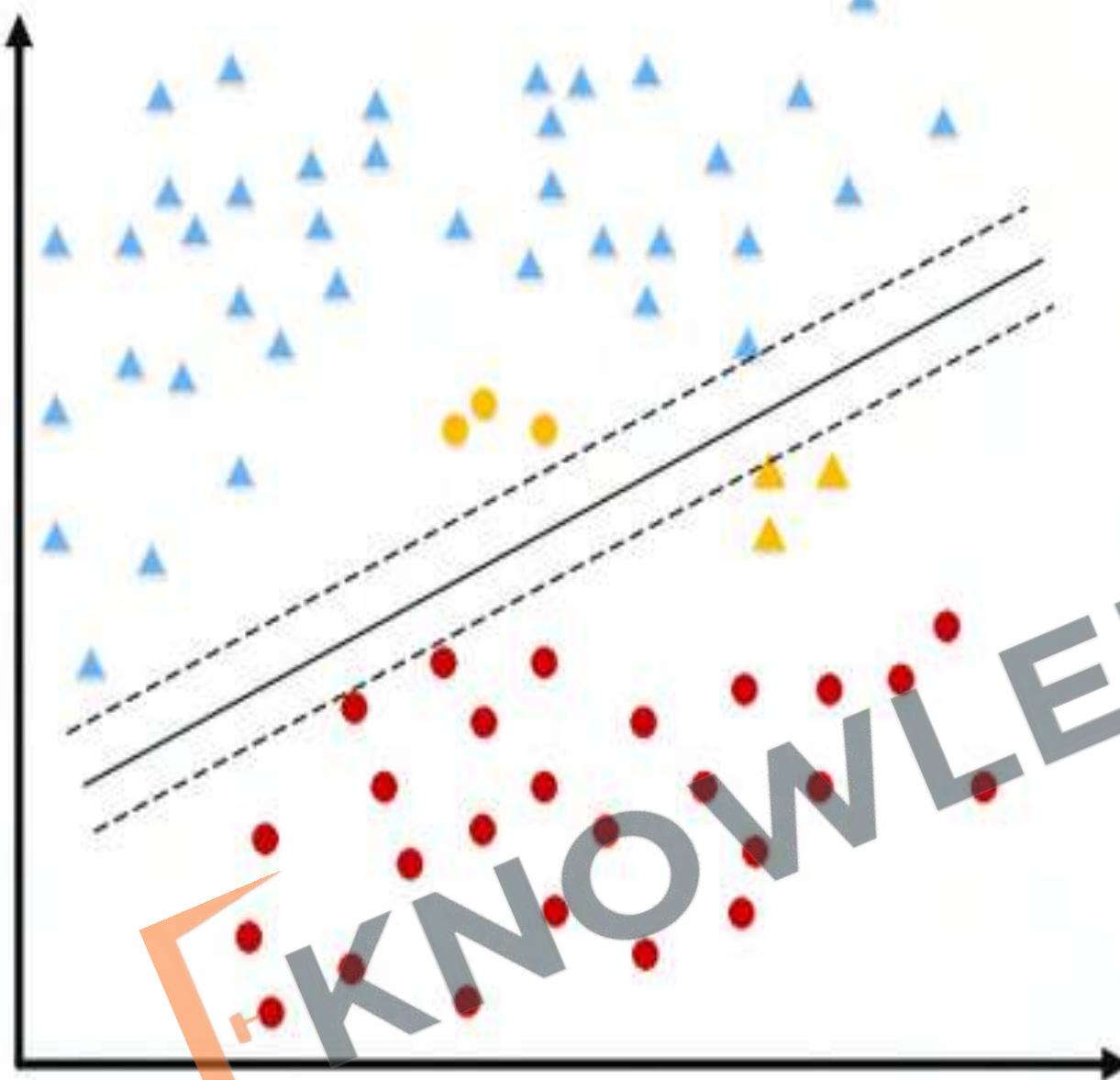
• Applications of SVM:

- **Text and Hypertext Classification**: For filtering spam and categorizing text based content for news articles.
- **Image Classification**: Useful in categorizing images into different groups (e.g., animals, cars, fruits).
- **Handwritten Character Recognition**: Used to recognize letters and digits from handwritten documents.
- **Biological Sciences**: Applied in protein classification and cancer classification based on gene expression data.

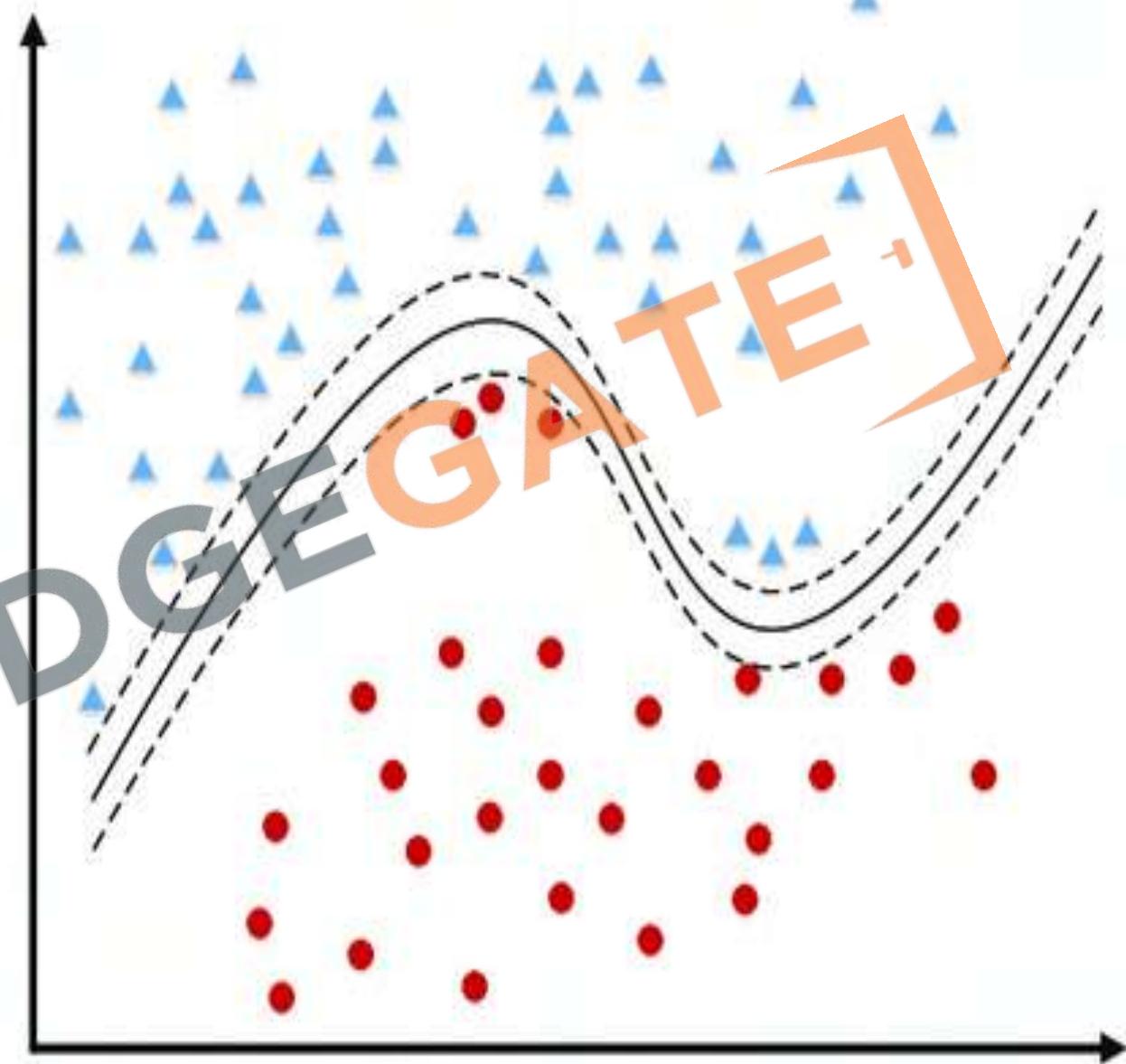


Linear SVM Vs Non-Liner SVM

Conclusion Point	Linear SVM	Non-Linear SVM ¹
Data Suitability	Best for linearly separable data	Suitable for complex, non-linearly separable data
Decision Boundary	Straight line or hyperplane	Non-linear boundary using kernel functions
Training Complexity	Simpler and faster	More computationally intensive
Kernel Usage	Not required	Requires kernel functions (e.g., RBF, polynomial)
Flexibility and Applications	Less flexible, used for simpler problems	Highly flexible, used for complex data patterns



A) Linear Separation



B) Non-linear Separation

Polynomial Kernel in SVM

- **Historical Context:** The polynomial kernel has been a fundamental part of kernel methods in machine learning since the 1990s. Introduced as a way to handle non-linearly separable data, it has helped extend the applicability of Support Vector Machines (SVMs) to more complex problems. Initially popularized by researchers like Vladimir Vapnik, the polynomial kernel has been used in various domains including image recognition, bioinformatics, and text classification.

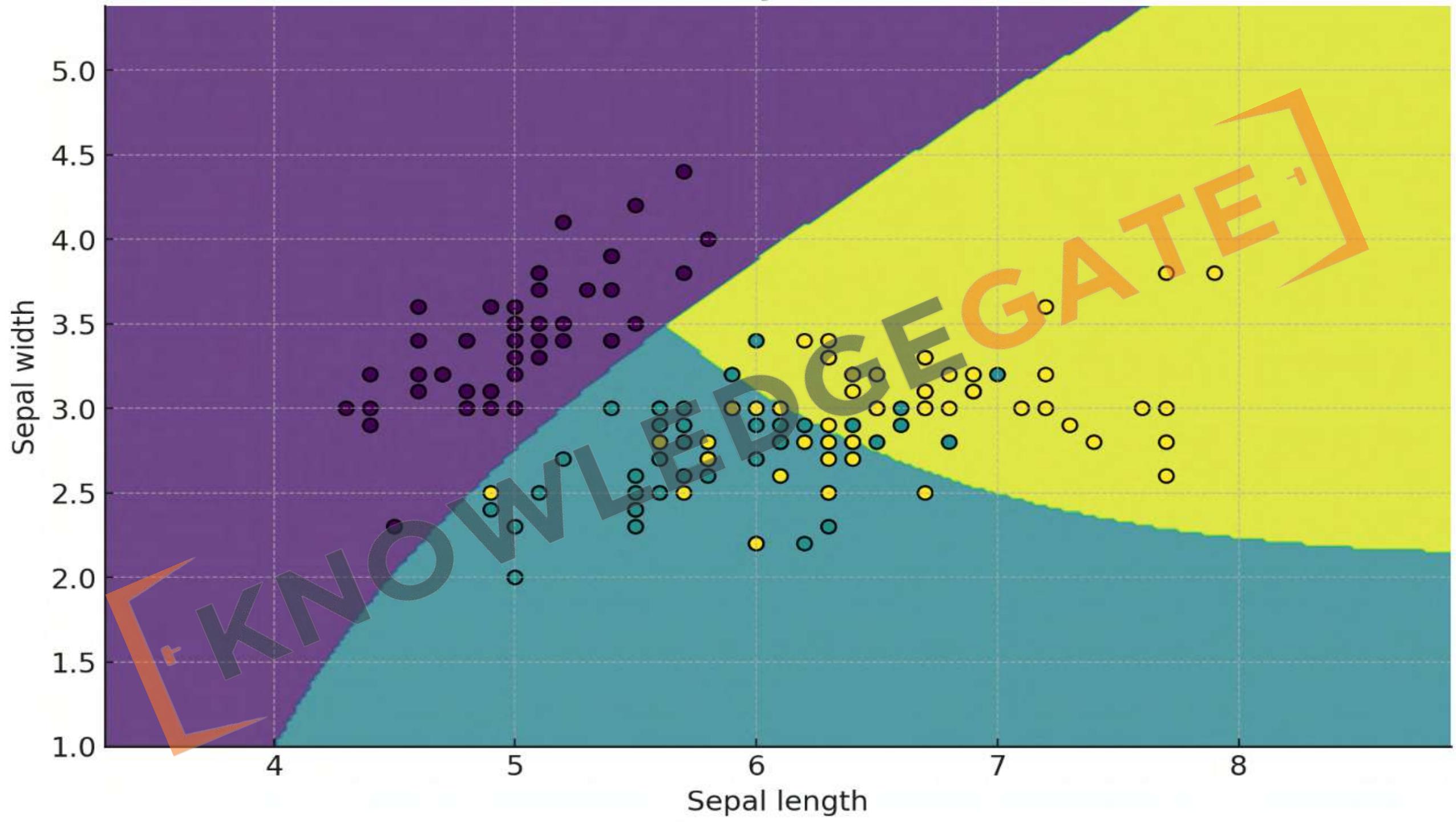
Mathematical Representation:

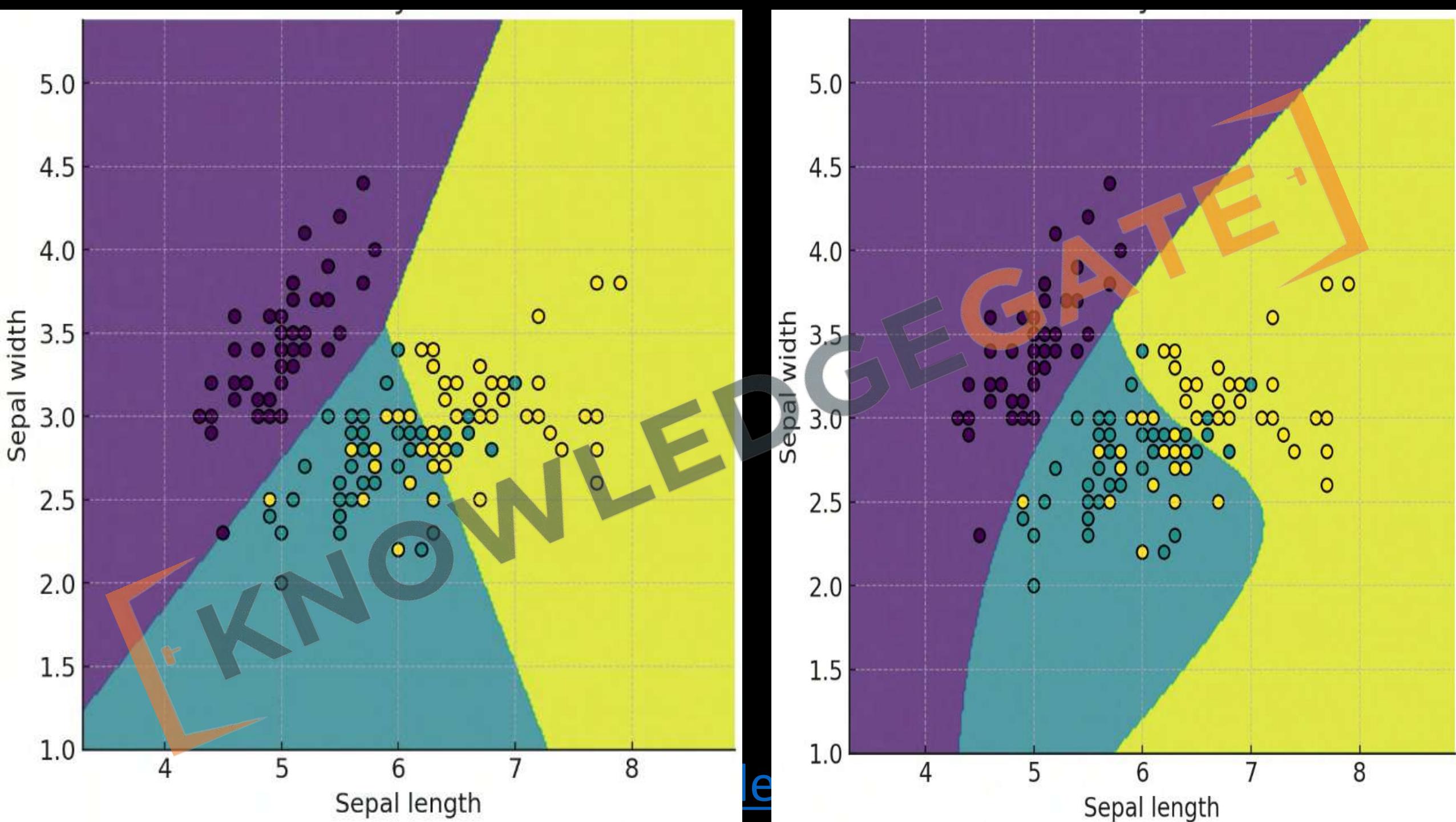
$$K(x, y) = (\gamma x^T y + r)^d$$

- x and y are input vectors.
- γ (gamma) is a scale factor, typically a positive value.
- r is a free parameter, generally a constant term.
- d is the degree of the polynomial.

Parameters:

- **Degree (d):** Determines the flexibility of the decision boundary. Higher degrees lead to more complex boundaries.
- **Gamma (γ):** Controls the influence of a single training example. Lower values mean 'far' and higher values mean 'close'.
- **Coefficient (r):** Shifts the kernel value by adding a constant.





Gaussian (RBF) Kernel in SVM

- The Gaussian kernel, also known as the Radial Basis Function (RBF) kernel, is widely used in SVM due to its ability to handle non-linear relationships between features. It maps input features into an infinite-dimensional space where a linear separator can be found.

The Gaussian kernel function is defined as:

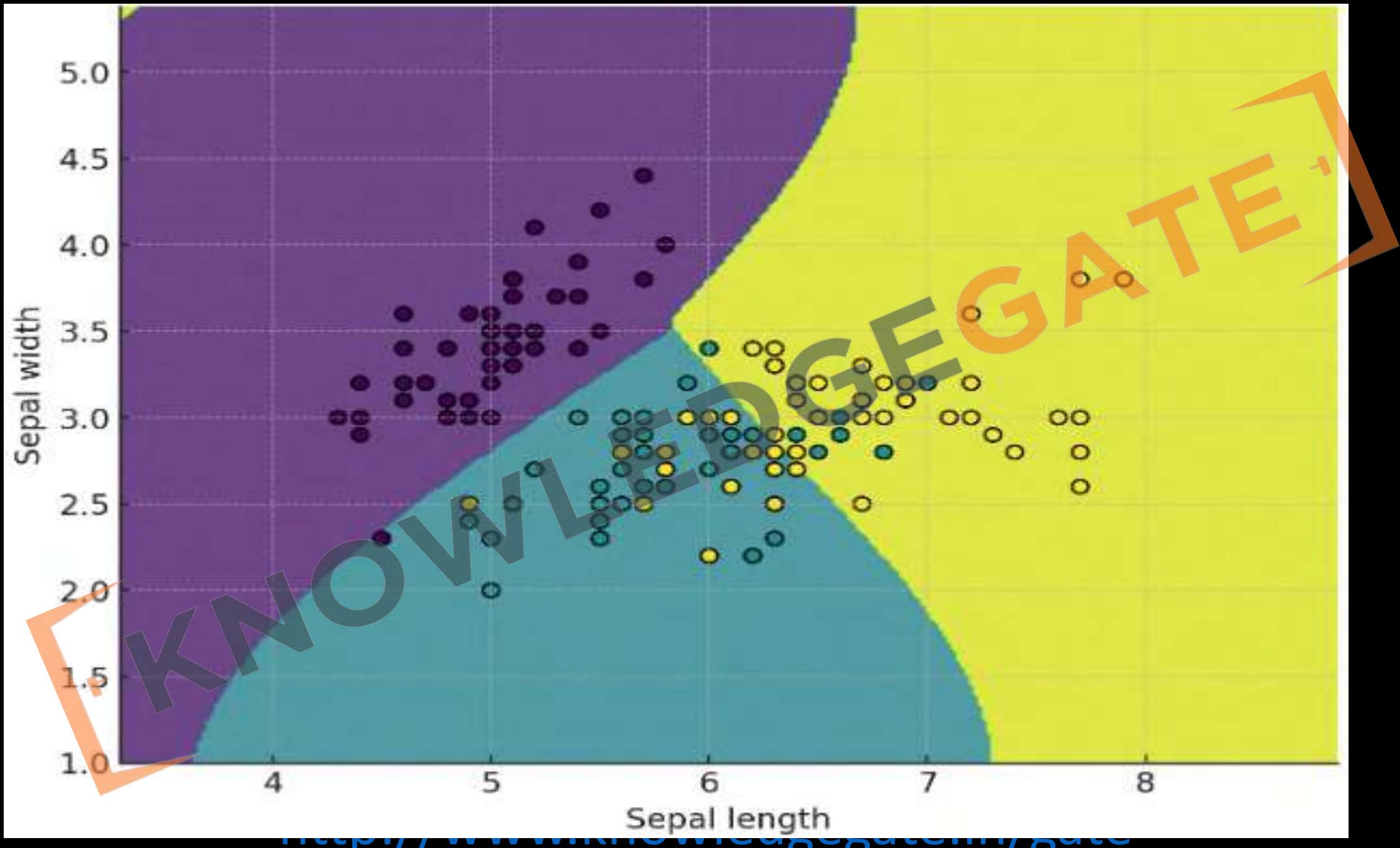
$$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

Where:

- x and y are input vectors.
- $\|x - y\|$ is the Euclidean distance between the two vectors.
- σ (sigma) is a parameter that defines the spread of the kernel (controls the width of the Gaussian).

In practice, the kernel is often written with a parameter γ (gamma) which is inversely proportional to σ^2 :

$$K(x, y) = \exp(-\gamma\|x - y\|^2)$$

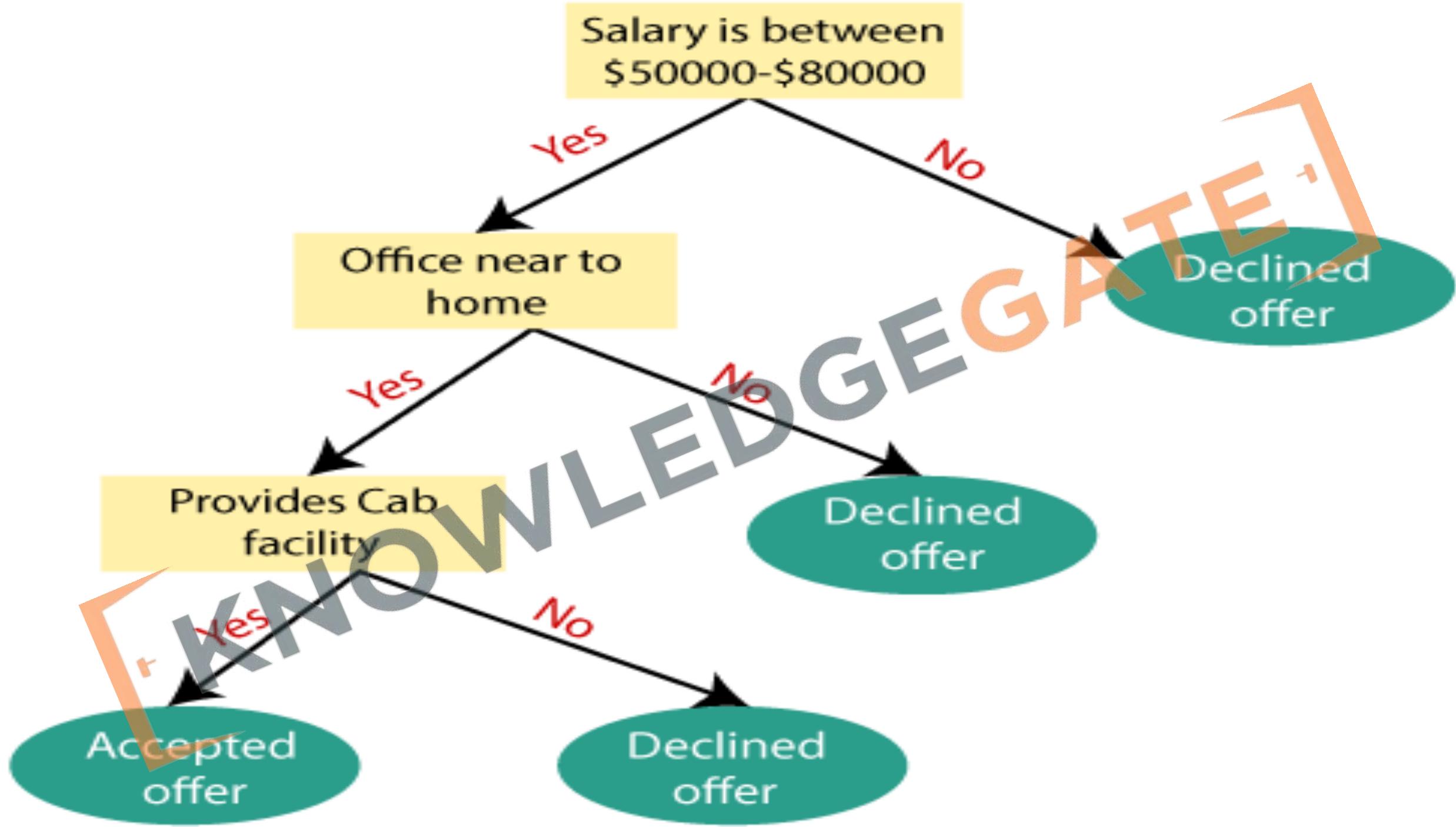


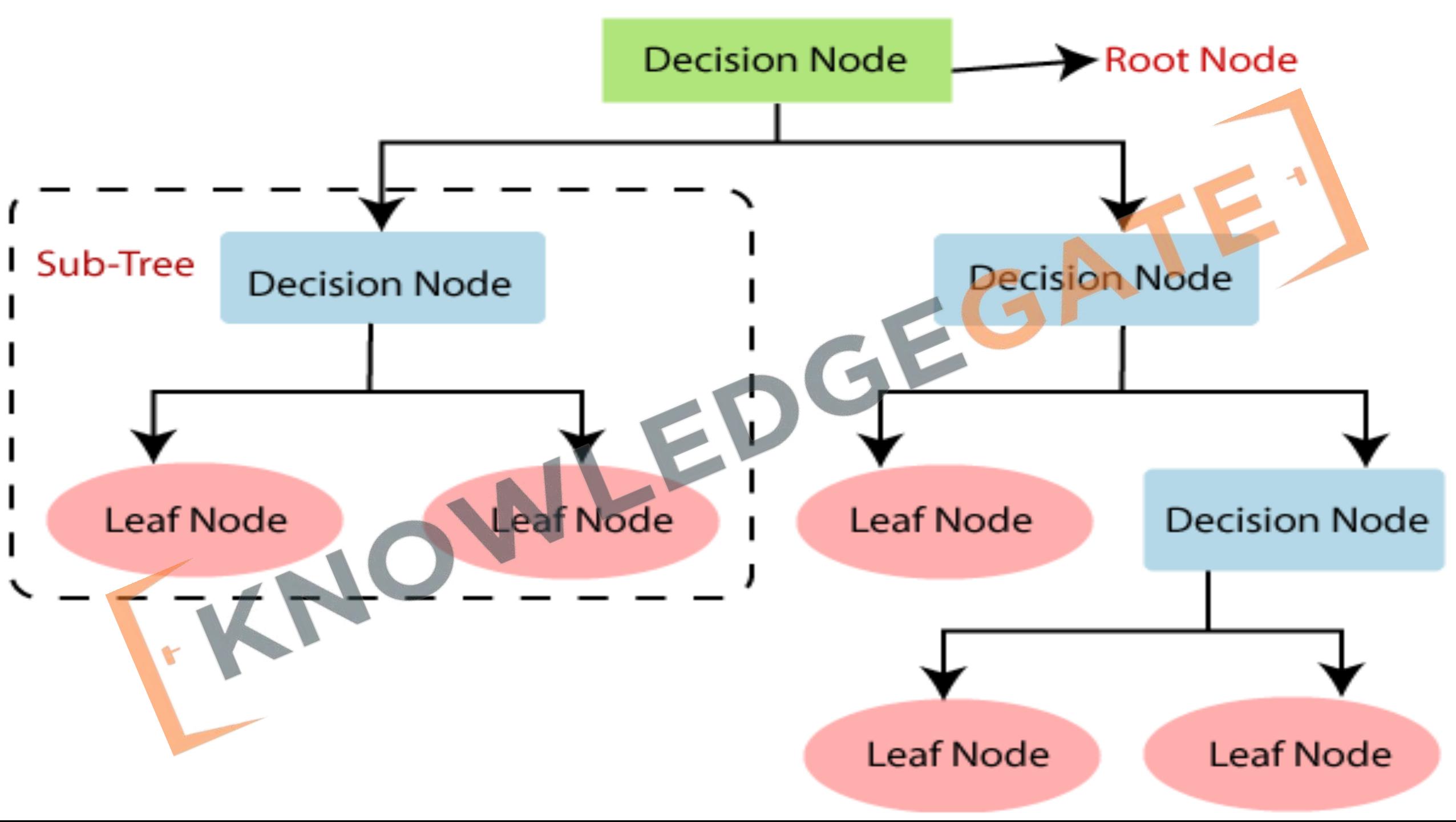
- **Applications**
 - **Image and text classification:** Effective for data with complex structures.
 - **Anomaly detection:** Widely used in one-class SVM for identifying outliers.
- **Advantages**
 - Capable of modelling complex decision boundaries.
 - Effective in high-dimensional spaces.
- **Disadvantages**
 - Requires careful tuning of the parameter γ .
 - Computationally intensive with large datasets.

- **(UNIT-3: DECISION TREE LEARNING) DECISION TREE**
LEARNING - Decision tree learning algorithm, Inductive bias, Inductive inference with decision trees, Entropy and information theory, Information gain, ID-3 Algorithm, Issues in Decision tree learning.
INSTANCE-BASED LEARNING - k-Nearest Neighbour Learning, Locally Weighted Regression, Radial basis function networks, Case-based learning.

Basic terminology used in Decision Trees

- **Root Node**: This is the starting point of the decision tree. It represents the entire dataset, which is then divided into smaller, more homogeneous groups.
- **Splitting**: This process involves dividing a node into two or more sub-nodes based on a certain condition that maximizes the separation of the classes.
- **Decision Node**: These nodes are where the splits happen. A decision node can lead to further decision nodes or to a leaf node.
- **Leaf (Terminal) Node**: These are the final nodes that do not split further. Each leaf node represents a classification or decision outcome.
- **Pruning**: This technique is used to reduce the size of a decision tree. It involves removing nodes that have little impact on the decision process, to prevent overfitting.
- **Branch/Sub-tree**: A section of the decision tree that represents a part of the whole decision process.
- **Parent and Child Nodes**: In any split, the original node is called the parent node, and the resulting sub-nodes are called child nodes.



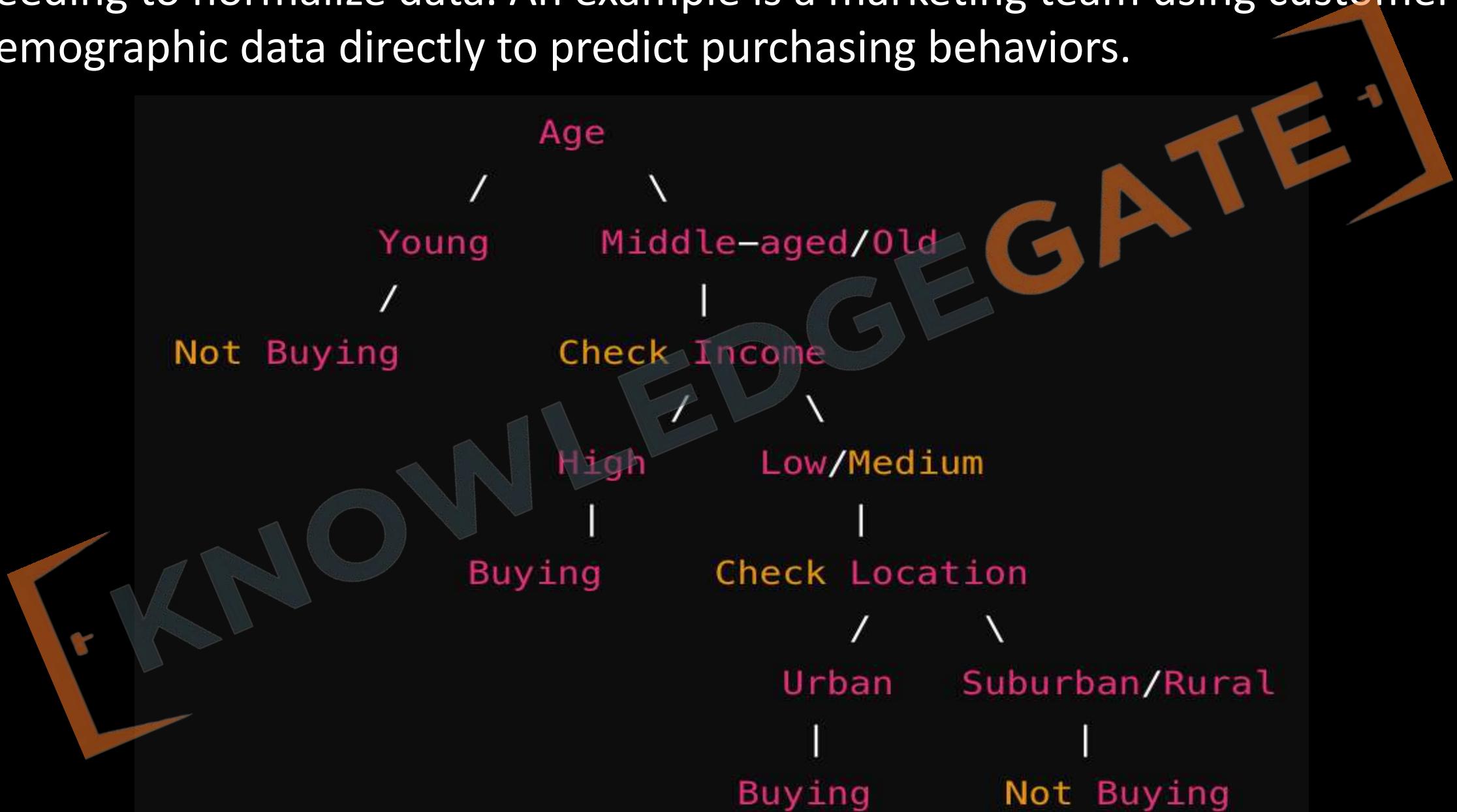


Decision trees are a popular tool for decision-making and prediction for several reasons

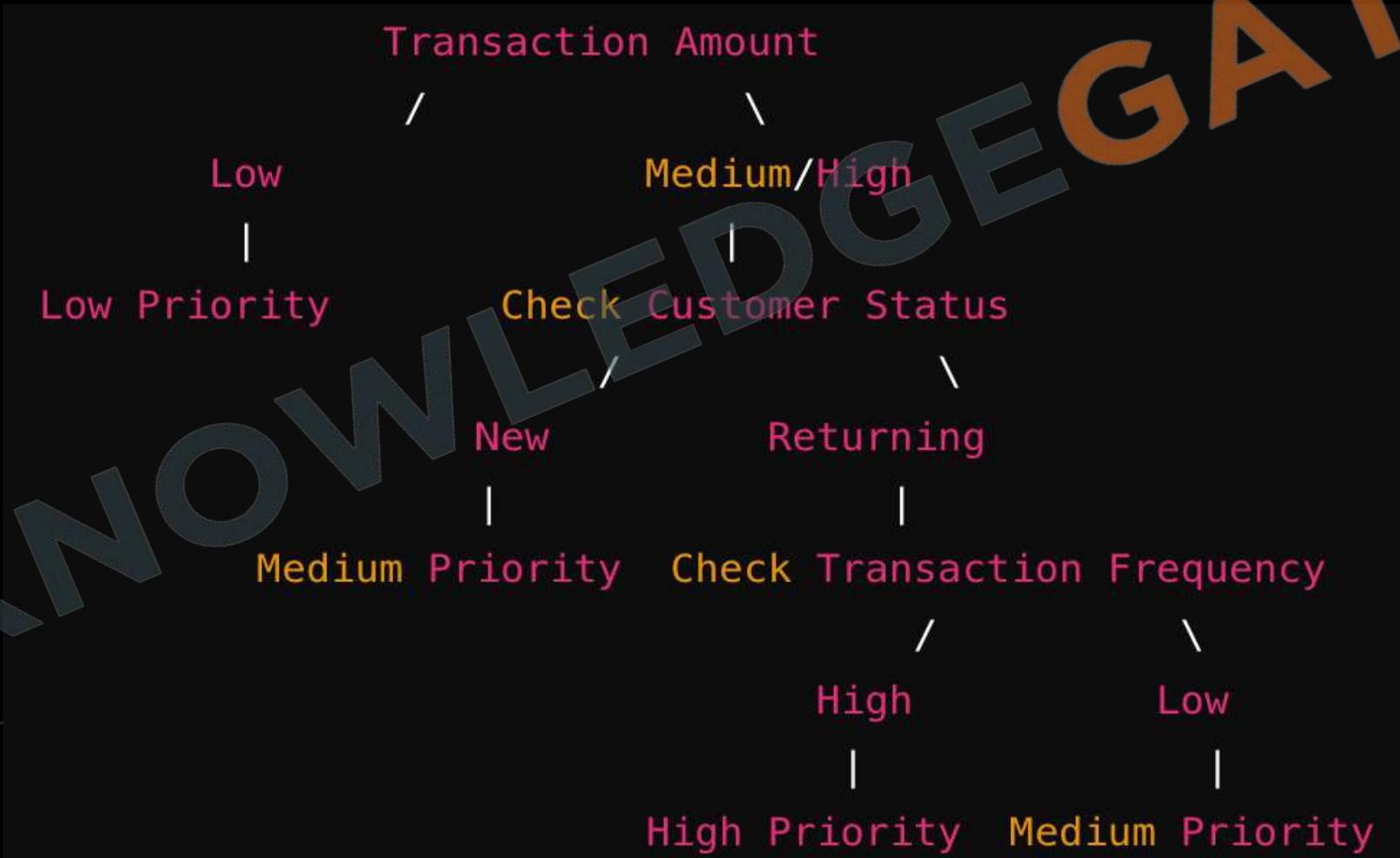
- **Clear Visualization**: Decision trees are easily visualized, allowing users to follow decision-making steps clearly. For example, a bank might use a decision tree to determine creditworthiness based on criteria like income, debt, and credit history.



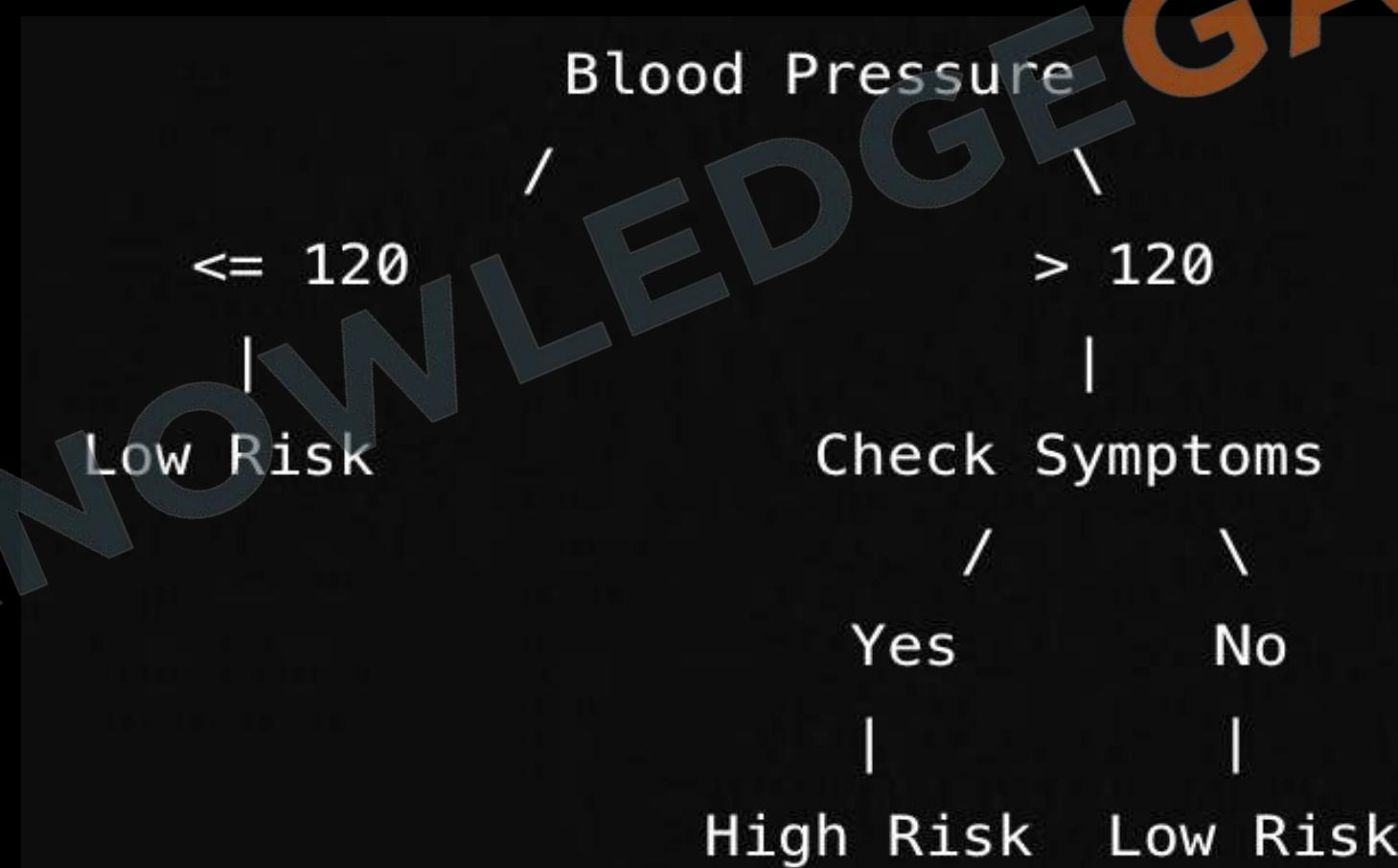
- **Low Data Preparation:** They require minimal data preprocessing, such as not needing to normalize data. An example is a marketing team using customer demographic data directly to predict purchasing behaviors.



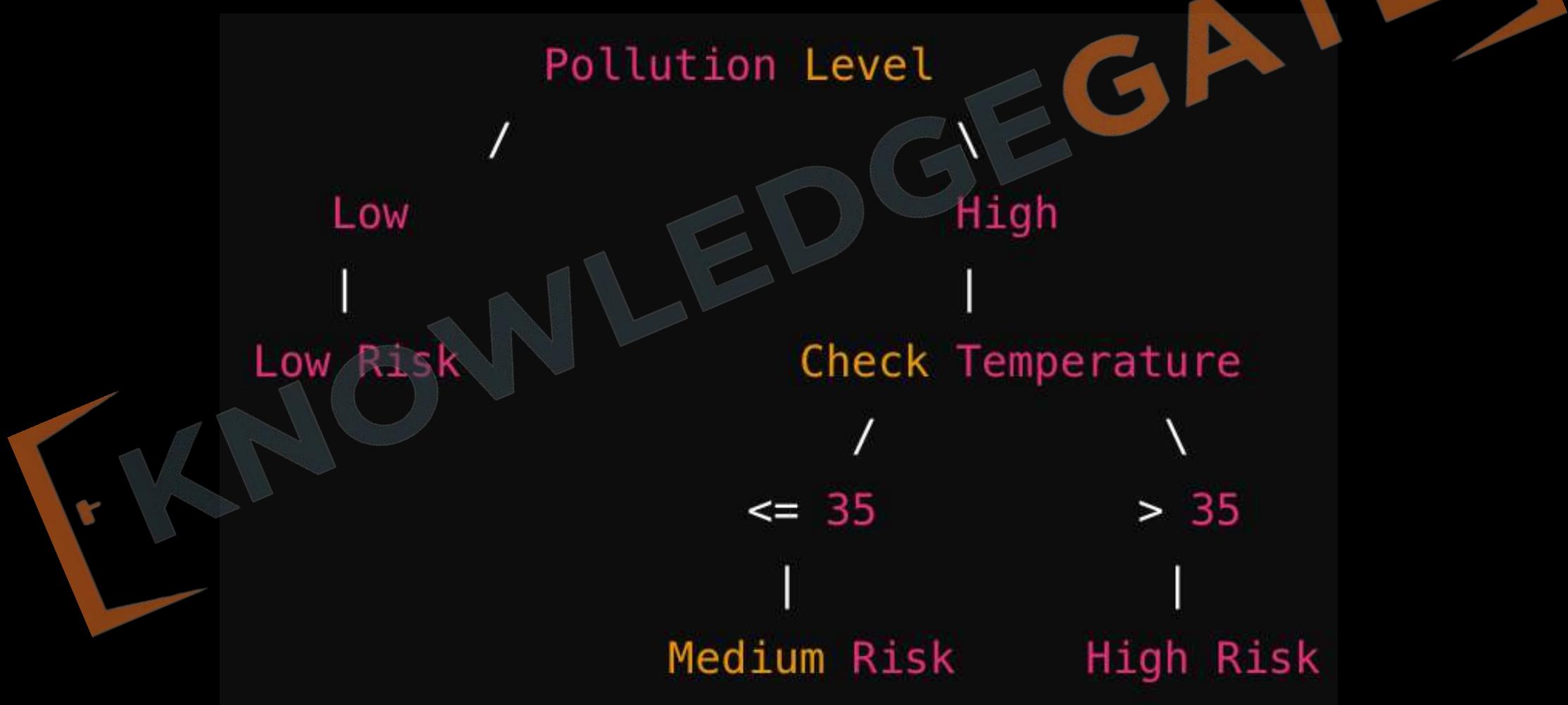
- **Efficient and Scalable:** The model is efficient, with prediction costs growing logarithmically with data size. For instance, an online retailer could use a decision tree to handle millions of customer transactions efficiently.



- **Flexible Data Handling:** Capable of processing both numerical and categorical data, making them suitable for diverse scenarios. For example, a healthcare provider could use decision trees to analyze patient data that includes both numerical (e.g., blood pressure readings) and categorical (e.g., symptom presence) data.



- **Robust to Assumptions:** Decision trees work well even when data assumptions are not met, such as non-linearity or non-normal distributions. An example is their use in environmental studies, where data often defies typical statistical assumptions.



Entropy

- Entropy is a measure of the uncertainty or impurity in a dataset. In the context of machine learning, particularly in decision trees, entropy quantifies the amount of randomness or disorder in the target variable. It helps in determining the best way to split the data to reduce this uncertainty and create pure subsets.

Interpretation:

- **Entropy = 0:** All instances in the dataset belong to a single class (pure set).
- **Entropy = 1:** The instances are equally distributed among all classes (maximum uncertainty).
- **$0 < \text{Entropy} < 1$:** The dataset contains a mix of classes, with varying degrees of uncertainty.

Formula:

For a binary classification problem, entropy $H(S)$ is calculated as:

$$H(S) = - \sum_{i=1}^c P_i \log_2 P_i$$

where:

- S is the set of examples (dataset).
- c is the number of classes.
- P_i is the probability of class i in the set S .

For a dataset with classes "Yes" and "No":

- $P(Yes)$ is the proportion of "Yes" instances in the dataset.
- $P(No)$ is the proportion of "No" instances in the dataset.

Calculate the probabilities:

- $P(Yes) = \frac{9}{14} \approx 0.643$
- $P(No) = \frac{5}{14} \approx 0.357$

Calculate the entropy using the formula:

$$\text{Entropy} = -(P(Yes) \log_2 P(Yes) + P(No) \log_2 P(No))$$

Substitute the probabilities into the entropy formula:

$$\text{Entropy} = -(0.643 \log_2 0.643 + 0.357 \log_2 0.357)$$

Calculate each term using a calculator:

$$0.643 \log_2 0.643 \approx -0.409$$

$$0.357 \log_2 0.357 \approx -0.530$$

Sum the values and take the negative:

$$\text{Entropy} = -(-0.409 + -0.530) \approx 0.939$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

1. Calculate Entropy for Each Subset:

We need to calculate the entropy for each subset of the "Outlook" attribute (Sunny, Overcast, Rain).

Subset: Sunny

- Count: 5
- Yes: 2
- No: 3

Entropy for Sunny:

$$H(\text{Sunny}) = -\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right)$$

$$H(\text{Sunny}) = -(0.4 \log_2 0.4 + 0.6 \log_2 0.6)$$

$$H(\text{Sunny}) = -(0.4 \times -1.322 + 0.6 \times -0.737)$$

$$H(\text{Sunny}) \approx 0.971$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Subset: Overcast

- Count: 4
- Yes: 4
- No: 0

Entropy for Overcast:

$$H(\text{Overcast}) = - \left(\frac{4}{4} \log_2 \frac{4}{4} + \frac{0}{4} \log_2 \frac{0}{4} \right)$$

$$H(\text{Overcast}) = -(1 \times 0 + 0 \times \log_2 0)$$

$$H(\text{Overcast}) = 0$$

Subset: Rain

- Count: 5
- Yes: 3
- No: 2

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy for Rain:

$$H(\text{Rain}) = - \left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5} \right)$$

$$H(\text{Rain}) = -(0.6 \log_2 0.6 + 0.4 \log_2 0.4)$$

$$H(\text{Rain}) = -(0.6 \times -0.737 + 0.4 \times -1.322)$$

$$H(\text{Rain}) \approx 0.971$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

1. Outlook

Outlook	Count	Yes	No
Sunny	5	2	3
Overcast	4	4	0
Rain	5	3	2

$$H(\text{Sunny}) = -\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) \approx 0.971$$

$$H(\text{Overcast}) = -\left(\frac{4}{4} \log_2 \frac{4}{4} + \frac{0}{4} \log_2 \frac{0}{4}\right) = 0$$

$$H(\text{Rain}) = -\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) \approx 0.971$$

$$H(\text{Outlook}) = \frac{5}{14}H(\text{Sunny}) + \frac{4}{14}H(\text{Overcast}) + \frac{5}{14}H(\text{Rain})$$

$$H(\text{Outlook}) = \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971$$

$$H(\text{Outlook}) \approx 0.693$$

$$\text{Gain}(S, \text{Outlook}) = H(S) - H(\text{Outlook})$$

$$\text{Gain}(S, \text{Outlook}) \approx 0.939 - 0.693 = 0.246$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Information Gain:

Entropy is used to calculate information gain, which measures the reduction in uncertainty about the target variable after splitting the data based on an attribute. The attribute with the highest information gain is chosen to split the dataset at each node in the tree.

$$\text{Information Gain}(S, \text{Attribute}) = H(S) - \sum_{v \in \text{Values}} \frac{|S_v|}{|S|} H(S_v)$$

where:

- S is the original dataset.
- S_v is the subset of S for which the attribute has value v .
- $H(S_v)$ is the entropy of the subset S_v .
- $\frac{|S_v|}{|S|}$ is the proportion of S represented by S_v .

By repeatedly choosing the attribute with the highest information gain, decision trees iteratively partition the data, reducing entropy and increasing purity in the resulting subsets, leading to the final decision tree.

2. Temperature

Temp	Count	Yes	No
Hot	4	2	2
Mild	6	4	2
Cool	4	3	1

$$H(Hot) = -\left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4}\right) = 1$$

$$H(Mild) = -\left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6}\right) \approx 0.918$$

$$H(Cool) = -\left(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}\right) \approx 0.811$$

$$H(Temp) = \frac{4}{14} H(Hot) + \frac{6}{14} H(Mild) + \frac{4}{14} H(Cool)$$

$$H(Temp) = \frac{4}{14} \times 1 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.811$$

$$H(Temp) \approx 0.911$$

$$\text{Gain}(S, \text{Temp}) = H(S) - H(Temp)$$

$$\text{Gain}(S, \text{Temp}) \approx 0.939 - 0.911 = 0.028$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

3. Humidity

Humidity	Count	Yes	No
High	7	3	4
Normal	7	6	1

$$H(High) = - \left(\frac{3}{7} \log_2 \frac{3}{7} + \frac{4}{7} \log_2 \frac{4}{7} \right) \approx 0.985$$

$$H(Normal) = - \left(\frac{6}{7} \log_2 \frac{6}{7} + \frac{1}{7} \log_2 \frac{1}{7} \right) \approx 0.592$$

$$H(Humidity) = \frac{7}{14} H(High) + \frac{7}{14} H(Normal)$$

$$H(Humidity) = \frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592$$

$$H(Humidity) \approx 0.789$$

$$\text{Gain}(S, \text{Humidity}) = H(S) - H(\text{Humidity})$$

$$\text{Gain}(S, \text{Humidity}) \approx 0.939 - 0.789 = 0.150$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

4. Wind

Wind	Count	Yes	No
Weak	8	6	2
Strong	6	3	3

$$H(Weak) = - \left(\frac{6}{8} \log_2 \frac{6}{8} + \frac{2}{8} \log_2 \frac{2}{8} \right) \approx 0.811$$

$$H(Strong) = - \left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6} \right) = 1$$

$$H(Wind) = \frac{8}{14} H(Weak) + \frac{6}{14} H(Strong)$$

$$H(Wind) = \frac{8}{14} \times 0.811 + \frac{6}{14} \times 1$$

$$H(Wind) \approx 0.892$$

$$\text{Gain}(S, \text{Wind}) = H(S) - H(Wind)$$

$$\text{Gain}(S, \text{Wind}) \approx 0.939 - 0.892 = 0.047$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Select the Attribute with the Highest Information Gain as the Root Node:

- In this case, "Outlook" has the highest information gain (0.246), so it becomes the root node.

Split the Dataset Based on the Root Node (Outlook):

- Create branches for each value of "Outlook": Sunny, Overcast, and Rain.

Outlook
Sunny / Overcast / Rain

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Step 2: Subset Splits

- Sunny:
 - Instances: D1, D2, D8, D9, D11
 - Calculate entropy and information gain for attributes: Temp, Humidity, Wind
 - Highest Gain: Humidity (e.g., $\text{Gain}(\text{Sunny}, \text{Humidity}) = 0.97)$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy of "PlayTennis" in Sunny Subset:

- Yes: 2
- No: 3

$$P(Yes) = \frac{2}{5} = 0.4$$

$$P(No) = \frac{3}{5} = 0.6$$

$$H(Sunny) = -(0.4 \log_2 0.4 + 0.6 \log_2 0.6)$$

$$H(Sunny) \approx -(0.4 \times -1.322 + 0.6 \times -0.737)$$

$$H(Sunny) \approx 0.971$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Temperature

- Hot: 2 instances (No, No)
- Mild: 2 instances (No, Yes)
- Cool: 1 instance (Yes)

Entropy for each subset:

$$H(\text{Hot}) = -(1 \log_2 1 + 0 \log_2 0) = 0$$

$$H(\text{Mild}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

$$H(\text{Cool}) = -(1 \log_2 1 + 0 \log_2 0) = 0$$

Weighted Entropy:

$$H(\text{Temp}) = \frac{2}{5} \times 0 + \frac{2}{5} \times 1 + \frac{1}{5} \times 0$$

$$H(\text{Temp}) = 0 + 0.4 + 0$$

$$H(\text{Temp}) = 0.4$$

Information Gain:

$$\text{Gain}(\text{Sunny}, \text{Temp}) = H(\text{Sunny}) - H(\text{Temp})$$

$$\text{Gain}(\text{Sunny}, \text{Temp}) = 0.971 - 0.4$$

$$\text{Gain}(\text{Sunny}, \text{Temp}) = 0.571$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Humidity

- High: 3 instances (No, No, No)
- Normal: 2 instances (Yes, Yes)

Entropy for each subset:

$$H(High) = -(0 \log_2 0 + 1 \log_2 1) = 0$$

$$H(Normal) = -(1 \log_2 1 + 0 \log_2 0) = 0$$

Weighted Entropy:

$$H(Humidity) = \frac{3}{5} \times 0 + \frac{2}{5} \times 0$$

$$H(Humidity) = 0$$

Information Gain:

$$\text{Gain}(Sunny, \text{Humidity}) = H(Sunny) - H(\text{Humidity})$$

$$\text{Gain}(Sunny, \text{Humidity}) = 0.971 - 0$$

$$\text{Gain}(Sunny, \text{Humidity}) = 0.971$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Wind

- Weak: 3 instances (No, No, Yes)
- Strong: 2 instances (No, Yes)

Entropy for each subset:

$$H(Weak) = -(0.333 \log_2 0.333 + 0.666 \log_2 0.666) \approx 0.918$$

$$H(Strong) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Weighted Entropy:

$$H(Wind) = \frac{3}{5} \times 0.918 + \frac{2}{5} \times 1$$

$$H(Wind) \approx 0.950$$

Information Gain:

$$\text{Gain}(Sunny, Wind) = H(Sunny) - H(Wind)$$

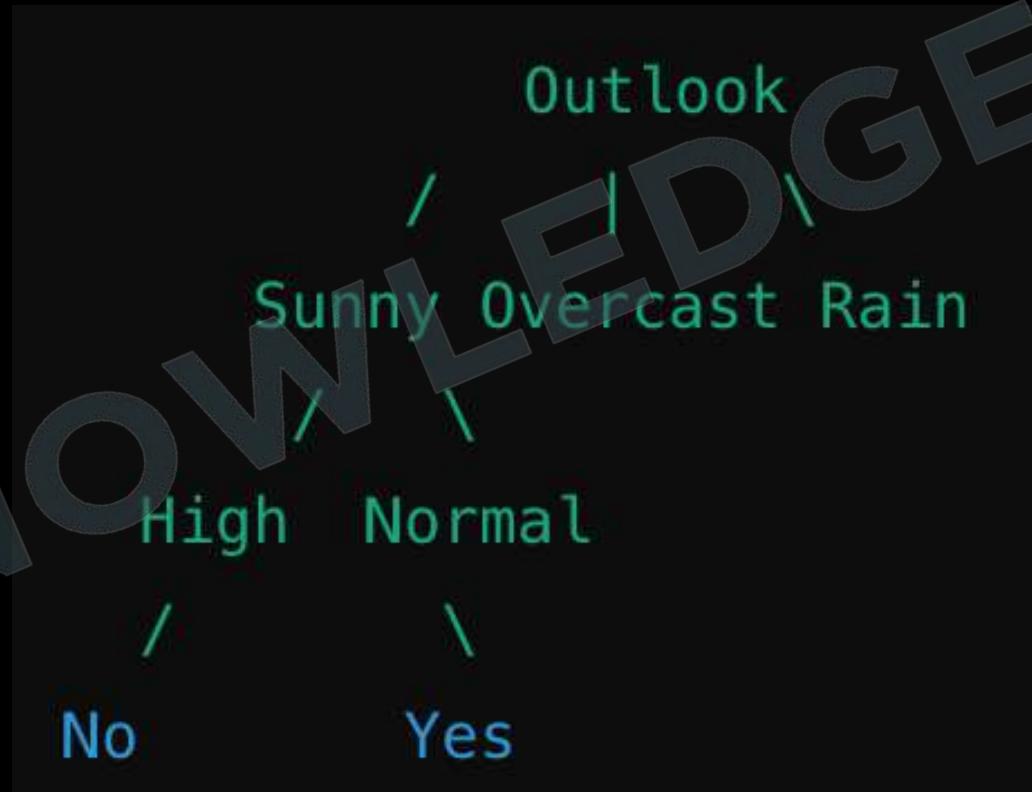
$$\text{Gain}(Sunny, Wind) = 0.971 - 0.950$$

$$\text{Gain}(Sunny, Wind) = 0.021$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

4. Choose the Attribute with the Highest Information Gain

From the calculations, "Humidity" has the highest information gain (0.971). Therefore, we choose "Humidity" to split the "Sunny" subset.



Step 2: Subset Splits

- Sunny:
 - Instances: D1, D2, D8, D9, D11
 - Calculate entropy and information gain for attributes: Temp, Humidity, Wind
 - Highest Gain: Humidity (e.g., $\text{Gain}(\text{Sunny}, \text{Humidity}) = 0.97)$

Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Overcast:

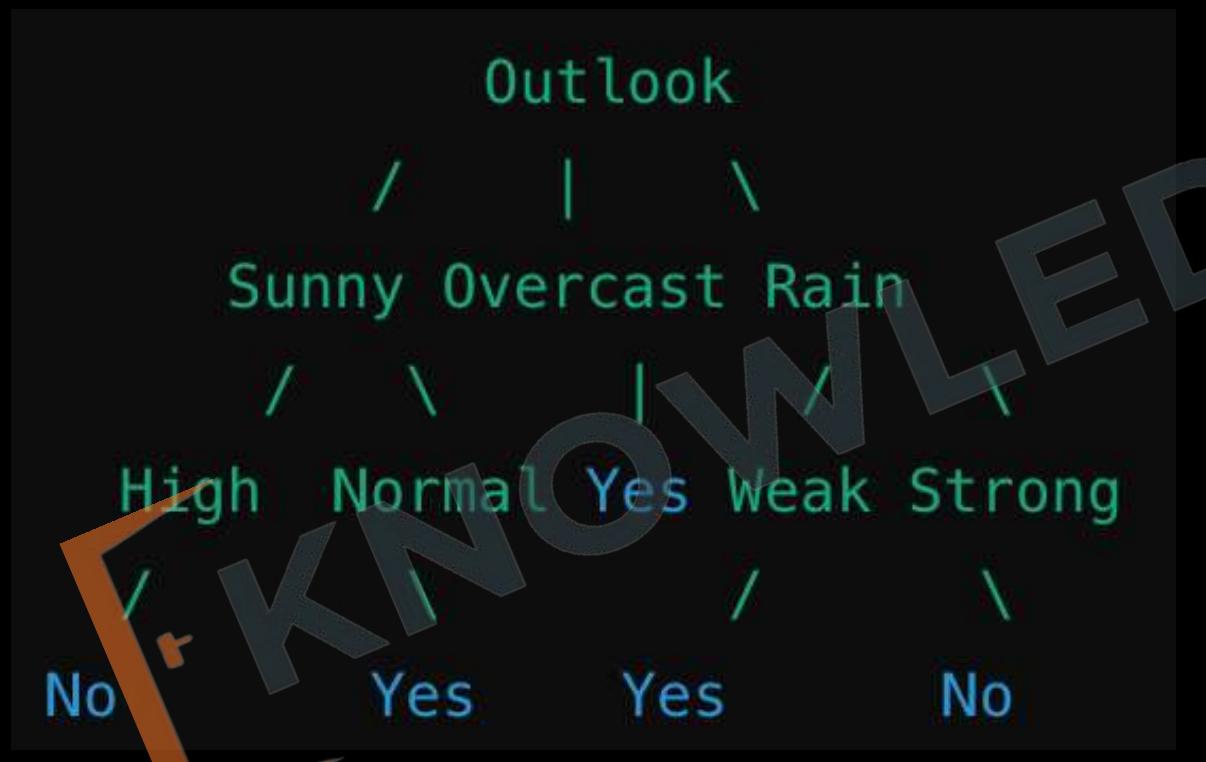
- Instances: D3, D7, D12, D13
- All instances are "Yes", so it becomes a leaf node.



Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Rain:

- Instances: D4, D5, D6, D10, D14
- Calculate entropy and information gain for attributes: Temp, Humidity, Wind
- Highest Gain: Wind (e.g., $\text{Gain}(\text{Rain}, \text{Wind}) = 0.97$)



Day	Outlook	Temp	Humidity	Wind	Play Cricket
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ID3 (Iterative Dichotomiser 3)

- **Historical Context**: Developed by Ross Quinlan in 1986. One of the earliest algorithms used for decision tree construction.
- **Algorithm**: ID3 uses a top-down, greedy approach to build a decision tree. Splits the dataset on the attribute with the highest information gain (based on entropy).
- **Usage Today**: Primarily of historical interest and educational purposes. Basis for more advanced algorithms like C4.5.
- **Advantages**: Simple and easy to understand. Good for small datasets.
- **Disadvantages**: Prone to overfitting. Handles only categorical attributes. Doesn't handle missing values well.

C4.5

- **Historical Context:** Developed by Ross Quinlan as an extension of ID3 in 1993. Became one of the most widely used decision tree algorithms.
- **Algorithm:** Uses gain ratio (normalized information gain) to select the best attribute for splitting. Can handle both categorical and continuous attributes. Prunes the tree after it is created to avoid overfitting. Handles missing values by assigning probabilities to each potential value.
- **Usage Today:** Used in various machine learning applications and often cited in academic research. Basis for the popular data mining tool, See5/C5.0.
- **Advantages:** More robust than ID3, handling both categorical and continuous data. Includes tree pruning to reduce overfitting. Handles missing values better than ID3.
- **Disadvantages:** More complex than ID3. Computationally intensive due to gain ratio calculations.

CART (Classification and Regression Trees)

- **Historical Context**: Developed by Breiman, Friedman, Olshen, and Stone in 1984. Introduced to provide a method for both classification and regression tasks.
- **Algorithm**: Uses Gini impurity (for classification) or variance reduction (for regression) to split nodes. Can handle both categorical and continuous attributes. Performs binary splits, which simplifies the tree but may increase its depth. Includes pruning to avoid overfitting.
- **Usage Today**: Widely used in various machine learning applications, especially in ensemble methods like Random Forests and Gradient Boosting Machines.
- **Advantages**: Versatile, supporting both classification and regression tasks. Handles both categorical and continuous data. Simplifies splits to binary decisions, making it easy to implement. Robust and forms the basis of many ensemble learning techniques.
- **Disadvantages**: Prone to overfitting if not pruned properly. Can create deep trees, which may be difficult to interpret.

Each algorithm has contributed significantly to the development of decision tree methods in machine learning:

- ID3 laid the foundation for decision tree algorithms but is less commonly used today due to its limitations.
- C4.5 improved on ID3 by handling a wider variety of data types and implementing pruning techniques, making it more robust and versatile.
- CART further expanded the capabilities of decision trees by supporting both classification and regression tasks, simplifying tree splits, and becoming integral to advanced ensemble methods.

Inductive Bias in Machine Learning

- **Definition:** Inductive bias refers to the set of assumptions a learning algorithm makes to generalize from specific training data to unseen data. It guides the learning process and helps the algorithm to make predictions.

Learning Algorithms with Inductive Bias

- **Decision Trees (e.g., ID3, C4.5, CART)**: Preference for shorter trees (Occam's Razor), preference for attributes with higher information gain or lower Gini impurity. Classification and regression tasks in various domains like finance, healthcare, and marketing.
- **Linear Regression**: Assumes a linear relationship between input features and the target variable. Predictive modeling in economics, biology, and engineering.
- **Neural Networks**: Assumes that complex patterns can be learned through multiple layers of non-linear transformations. Image and speech recognition, natural language processing, and game playing.
- **k-Nearest Neighbors (k-NN)**: Assumes that similar instances have similar outputs (locality-based bias). Recommendation systems, pattern recognition, and anomaly detection.
- **Support Vector Machines (SVM)** : Assumes a maximum margin hyperplane separates different classes. Text classification, image recognition, and bioinformatics.

Advantages of Inductive Bias

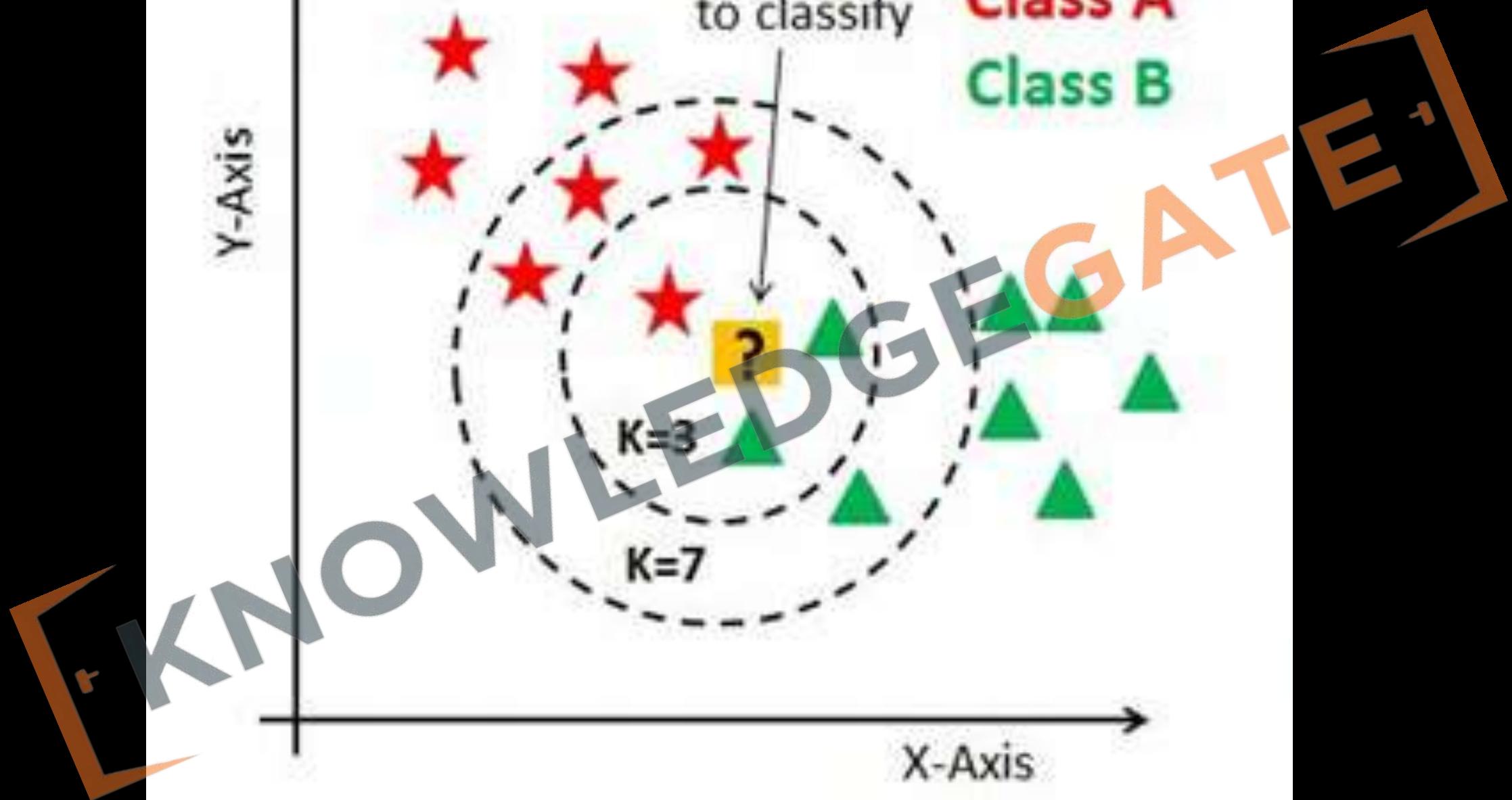
- Guides Generalization: Helps algorithms generalize from limited training data to unseen data.
- Reduces Overfitting: By incorporating domain-specific knowledge, it helps in reducing overfitting.
- Speeds Up Learning: Provides a framework that can speed up the learning process by narrowing the hypothesis space.

Disadvantages of Inductive Bias

- May Introduce Bias: If the inductive bias is incorrect, it can lead to poor generalization and biased predictions.
- Limited Flexibility: Strong inductive biases can limit the flexibility of the model, making it less adaptable to diverse data.
- Over-reliance on Assumptions: Incorrect assumptions can degrade the performance of the learning algorithm.

Instance-Based Learning in Machine Learning

- **Historical Context**: Instance-based learning has been foundational in pattern recognition and classification tasks since the early days of AI. The k-NN algorithm, one of the simplest and most effective instance-based methods, was developed in the 1950s.
- **Definition**: Instance-based learning, also known as memory-based learning or lazy learning, is a type of machine learning where the algorithm stores instances of training data and uses them to make predictions on new data. Instead of explicitly creating a model during the training phase, instance-based learning methods delay the generalization process until a query is made during the prediction phase.



Key Characteristics:

- **Lazy Learning**: No explicit model is built during training. The algorithm simply stores the training instances. Generalization occurs at the time of prediction.
- **Similarity-Based**: Predictions are made based on the similarity between the new data instance and stored training instances. Common similarity measures include Euclidean distance, Manhattan distance, and cosine similarity.
- **Local Decision Making**: Decisions are made based on the local neighbourhood of the input instance. The algorithm considers only a subset of the training data (e.g., the k-nearest neighbours) to make a prediction.

Example Algorithms:

- **k-Nearest Neighbors (k-NN):**
 - **Algorithm:** For a given input instance, the algorithm finds the k training instances closest to the input and assigns the most common label (for classification) or averages the labels (for regression).
 - **Application:** Used in recommendation systems, image recognition, and anomaly detection.

Advantages of Instance-Based Learning:

- **Simplicity**:
 - Easy to implement and understand.
 - No need for a complex model training phase.
- **Adaptability**:
 - Can adapt quickly to new data without retraining the entire model.
 - Useful for applications where the data distribution changes over time.
- **Performance**:
 - Effective for problems where the decision boundary is irregular and cannot be easily captured by a global model.

Disadvantages of Instance-Based Learning:

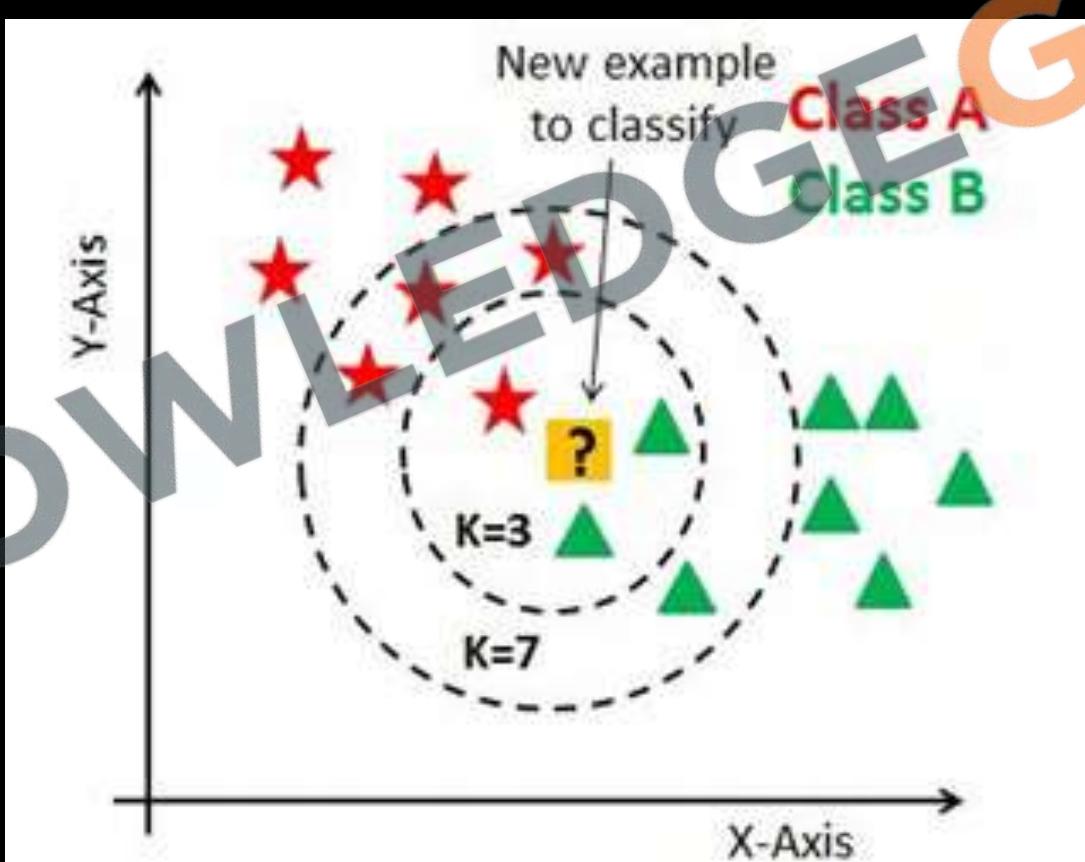
- **Computational Cost**:
 - High storage requirements since all training instances are retained.
 - Prediction can be slow for large datasets due to the need to compute distances to all stored instances.
- **Sensitivity to Noise**:
 - Can be sensitive to noisy data, as outliers can affect the prediction.
- **Feature Scaling**:
 - Performance depends on the proper scaling of features, as different scales can distort distance calculations.

K-Nearest Neighbors (KNN)

- **Historical Context:** Introduced in the 1950s by Fix and Hodges, formalized in the 1960s by Cover and Hart, KNN was initially popular for pattern recognition and statistical estimation due to its simplicity and intuitive approach.

K-Nearest Neighbors (KNN)

- K-Nearest Neighbors (KNN) is a simple, non-parametric, and lazy learning algorithm used for classification and regression tasks. KNN is an instance-based learning algorithm that classifies a data point based on how its neighbors are classified.



- Algorithm Steps:
 - **Select K:** Choose the number of neighbors (K).
 - **Calculate Distance:** Compute the distance between the new data point and all the training points (commonly used distance metrics are Euclidean, Manhattan).
 - **Find Nearest Neighbors:** Identify the K nearest neighbors to the new data point.
 - **Vote (for classification):** Each neighbor votes for their class, and the class with the most votes is assigned to the data point.
 - **Average (for regression):** The average value of the K nearest neighbors is taken as the prediction.

Euclidean Distance: $\sqrt{\sum (x_i - y_i)^2}$

Manhattan Distance: $\sum |x_i - y_i|$

- To classify the new data point (170, 57) using the K-Nearest Neighbors (KNN) algorithm and Euclidean distance, follow these steps

Calculate the Euclidean Distance:

- The Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is given by:

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

Compute Distances for All Points:

- Calculate the distance between the new point (170, 57) and each point in the dataset.

Select the Nearest Neighbors:

- Choose the $K = 3$ nearest neighbors.

Height (cm)	Weight (kg)	Classification
168	50	Underweight
181	63	Normal
175	68	Normal
172	63	Normal
173	64	Normal
173	55	Underweight
170	59	Normal
174	56	Normal
170	57	?

Distance to (168, 50):

$$d = \sqrt{(170 - 168)^2 + (57 - 50)^2} = \sqrt{2^2 + 7^2} = \sqrt{4 + 49} = \sqrt{53} \approx 7.28$$

Distance to (181, 63):

$$d = \sqrt{(170 - 181)^2 + (57 - 63)^2} = \sqrt{(-11)^2 + (-6)^2} = \sqrt{121 + 36} = \sqrt{157} \approx 12.53$$

Distance to (175, 68):

$$d = \sqrt{(170 - 175)^2 + (57 - 68)^2} = \sqrt{(-5)^2 + (-11)^2} = \sqrt{25 + 121} = \sqrt{146} \approx 12.08$$

Distance to (172, 63):

$$d = \sqrt{(170 - 172)^2 + (57 - 63)^2} = \sqrt{(-2)^2 + (-6)^2} = \sqrt{4 + 36} = \sqrt{40} \approx 6.32$$

Distance to (173, 64):

$$d = \sqrt{(170 - 173)^2 + (57 - 64)^2} = \sqrt{(-3)^2 + (-7)^2} = \sqrt{9 + 49} = \sqrt{58} \approx 7.62$$

Distance to (173, 55):

$$d = \sqrt{(170 - 173)^2 + (57 - 55)^2} = \sqrt{(-3)^2 + 2^2} = \sqrt{9 + 4} = \sqrt{13} \approx 3.61$$

Distance to (170, 59):

$$d = \sqrt{(170 - 170)^2 + (57 - 59)^2} = \sqrt{0^2 + (-2)^2} = \sqrt{4} = 2$$

Distance to (174, 56):

$$d = \sqrt{(170 - 174)^2 + (57 - 56)^2} = \sqrt{(-4)^2 + 1^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.12$$

Height (cm)	Weight (kg)	Classification
168	50	Underweight
181	63	Normal
175	68	Normal
172	63	Normal
173	64	Normal
173	55	Underweight
170	59	Normal
174	56	Normal
170	57	?

Nearest Neighbors:

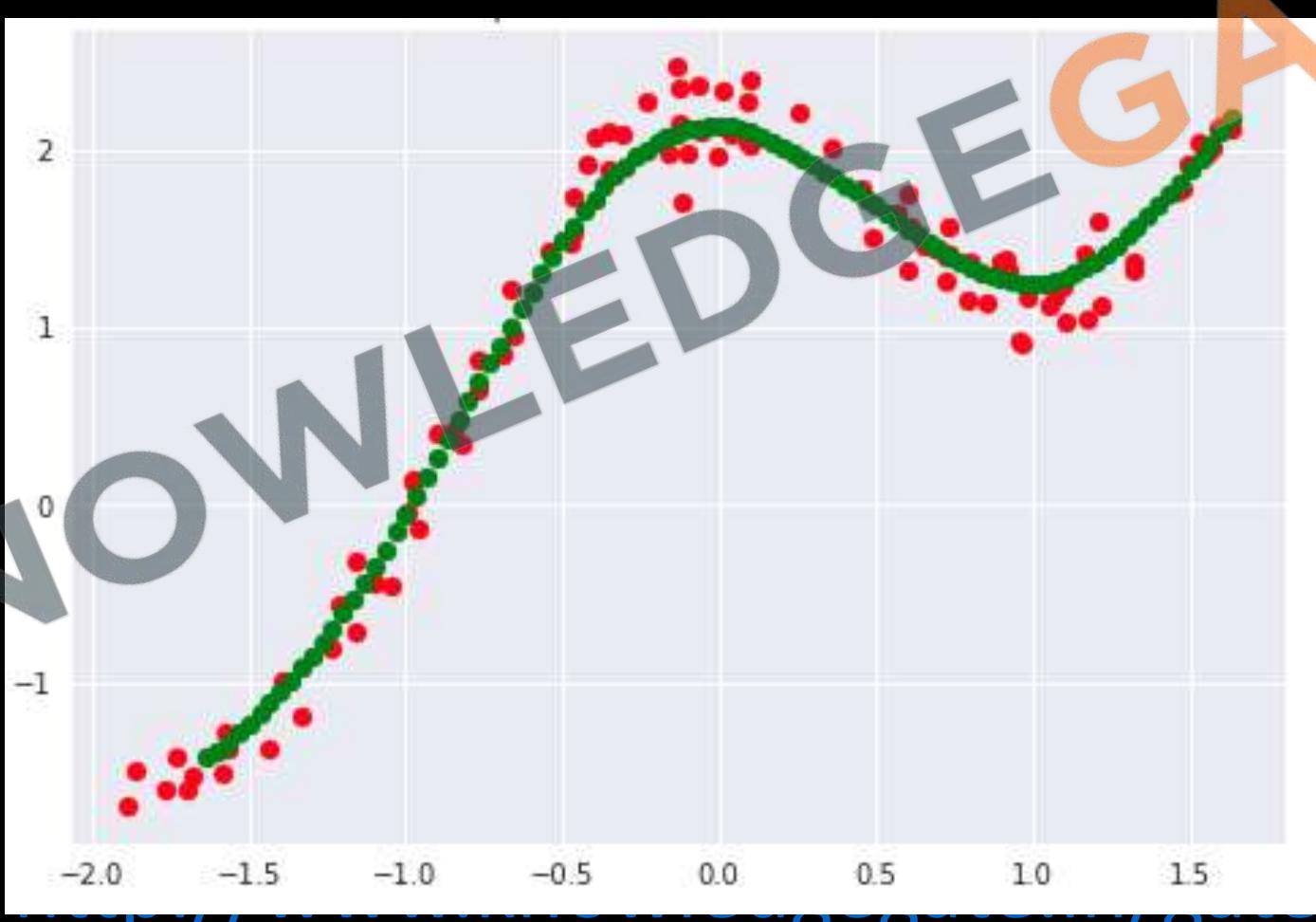
- The distances are:
 - (170, 59): 2 (Normal)
 - (173, 55): 3.61 (Underweight)
 - (174, 56): 4.12 (Normal)
- Classification:
 - The majority class among the nearest neighbors is "Normal".
 - Thus, the new data point (170, 57) is classified as "Normal".

- **Advantages:**
 - Simple and easy to implement.
 - No training phase (lazy learning), which makes it fast for small datasets.
- **Disadvantages:**
 - Computationally expensive for large datasets.
 - Sensitive to the choice of K and the distance metric.
 - Poor performance with imbalanced data.
- **Applications:**
 - Pattern recognition
 - Data mining
 - Image classification
 - Recommendation systems

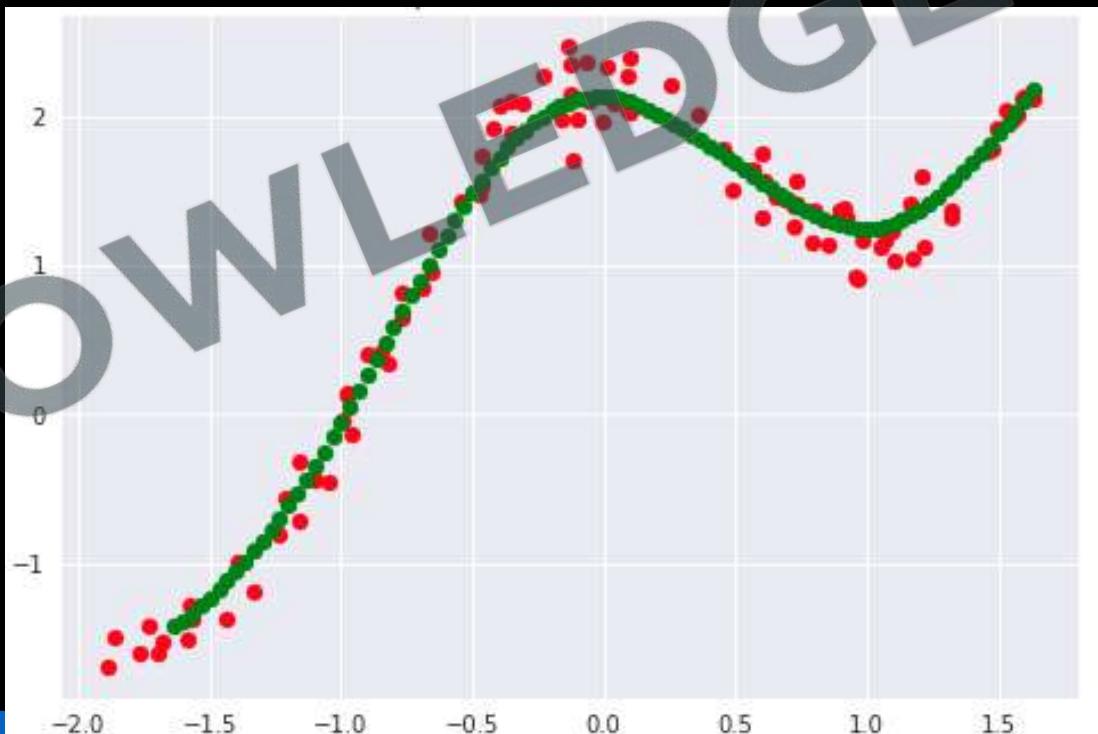
- **Modern-Day Use**: KNN is currently applied in image recognition, recommendation systems, anomaly detection, and genomics. It remains favoured for its ease of implementation and versatility, despite computational challenges, which are mitigated by techniques like KD-Trees, PCA, and distributed computing.
- **Current Research and Trends**: Ongoing research focuses on optimizing KNN for specific applications and enhancing efficiency through approximate nearest neighbour searches. Trends include integrating KNN with deep learning models, using it in edge computing and IoT, and improving performance with parallel processing and GPU acceleration.

Locally Weighted Regression (LWR)

- Locally Weighted Regression (LWR), also known as Locally Weighted Scatterplot Smoothing (LOWESS or LOESS), is a non-parametric regression method that fits multiple regressions in localized subsets of the data to produce a smooth curve through points in a scatter plot.



- **Non-Parametric**: LWR does not assume a fixed form (parametric form) for the relationship between predictors and the response variable. It adapts to the data locally.
- **Localized Fitting**: Regression is performed at each point using a subset of data that is close to the point in question. This local fitting is weighted by the distance of each data point to the point being estimated.
- **Weighting**: Points closer to the point of interest have higher weights. Commonly used weighting functions include the Gaussian kernel and the tricube kernel.



- Imagine you are a botanist studying the growth of trees in a forest. You have a small dataset of tree heights at various ages, and you want to predict the height of a tree that is 3 years old.

Age (years)	Height (meters)
1	2
2	3
3	2
4	5
5	4

Select a Query Point:

- Our query point is $x_q = 3$.

Compute Weights:

- We use a weighting function to assign weights to each data point based on their distance to x_q .
- For simplicity, let's use a Gaussian kernel with bandwidth parameter $\tau = 1$:

$$w_i = \exp\left(-\frac{(x_i - x_q)^2}{2\tau^2}\right)$$

- Compute the weights for each data point:
 - For $x_1 = 1$:

$$w_1 = \exp\left(-\frac{(1 - 3)^2}{2 \times 1^2}\right) = \exp\left(-\frac{4}{2}\right) = \exp(-2) \approx 0.1353$$

- For $x_2 = 2$:

$$w_2 = \exp\left(-\frac{(2 - 3)^2}{2 \times 1^2}\right) = \exp\left(-\frac{1}{2}\right) = \exp(-0.5) \approx 0.6065$$

- For $x_3 = 3$:

$$w_3 = \exp\left(-\frac{(3 - 3)^2}{2 \times 1^2}\right) = \exp(0) = 1$$

- For $x_4 = 4$:

$$w_4 = \exp\left(-\frac{(4 - 3)^2}{2 \times 1^2}\right) = \exp\left(-\frac{1}{2}\right) = \exp(-0.5) \approx 0.6065$$

- For $x_5 = 5$:

$$w_5 = \exp\left(-\frac{(5 - 3)^2}{2 \times 1^2}\right) = \exp\left(-\frac{4}{2}\right) = \exp(-2) \approx 0.1353$$

Age (years)	Height (meters)
1	2
2	3
3	2
4	5
5	4

- The goal of Locally Weighted Regression (LWR) is to fit a linear model to the data points in a localized region around a query point. To achieve this, we assign weights to the data points based on their distance from the query point, and then perform a weighted least squares regression. This ensures that points closer to the query point have more influence on the model than points farther away.

Formulate the Weighted Linear Regression Problem

We need to minimize the weighted sum of squared errors:

$$\sum_i w_i (y_i - (\theta_0 + \theta_1 x_i))^2$$

Using matrix notation:

- W is a diagonal matrix of weights.
- X is the matrix of input features, including a column of ones for the intercept.
- y is the vector of target values.

Matrices and Vectors

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}, \quad W = \begin{bmatrix} 0.1353 & 0 & 0 & 0 & 0 \\ 0 & 0.6065 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.6065 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix}, \quad y = \begin{bmatrix} 2 \\ 3 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

Normal Equation for Weighted Linear Regression

$$\theta = (X^T W X)^{-1} X^T W y$$

Compute θ

1. Compute $X^T W X$:

$$X^T W X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0.1353 & 0 & 0 & 0 & 0 \\ 0 & 0.6065 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.6065 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}$$

2. Compute $X^T W y$:

$$X^T W y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0.1353 & 0 & 0 & 0 & 0 \\ 0 & 0.6065 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.6065 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

3. Solve for θ :

$$\theta = (X^T W X)^{-1} (X^T W y)$$

After computing the above, we get:

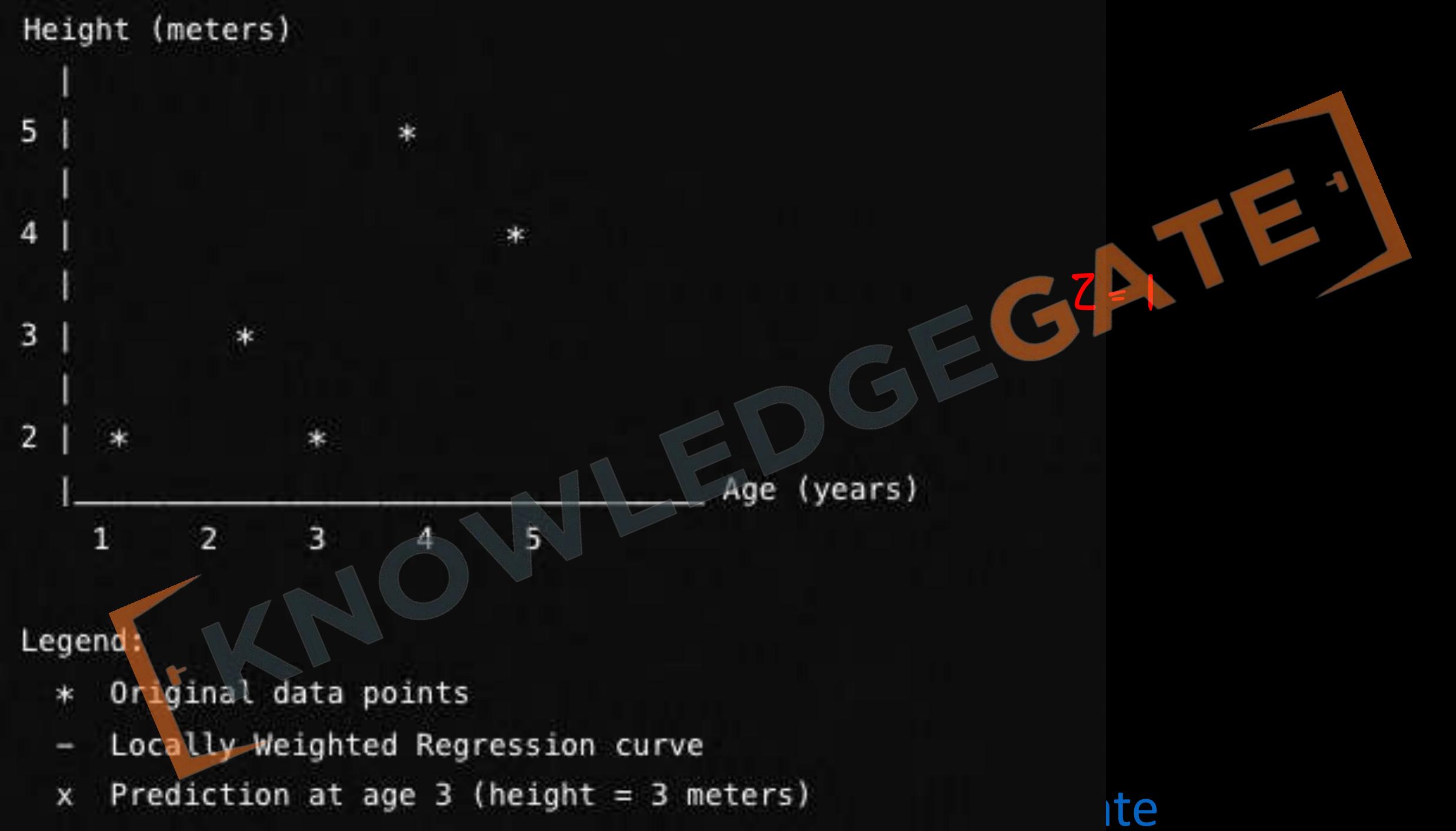
$$X^T W X = \begin{bmatrix} 2.0836 & 7.5149 \\ 7.5149 & 29.1717 \end{bmatrix}$$

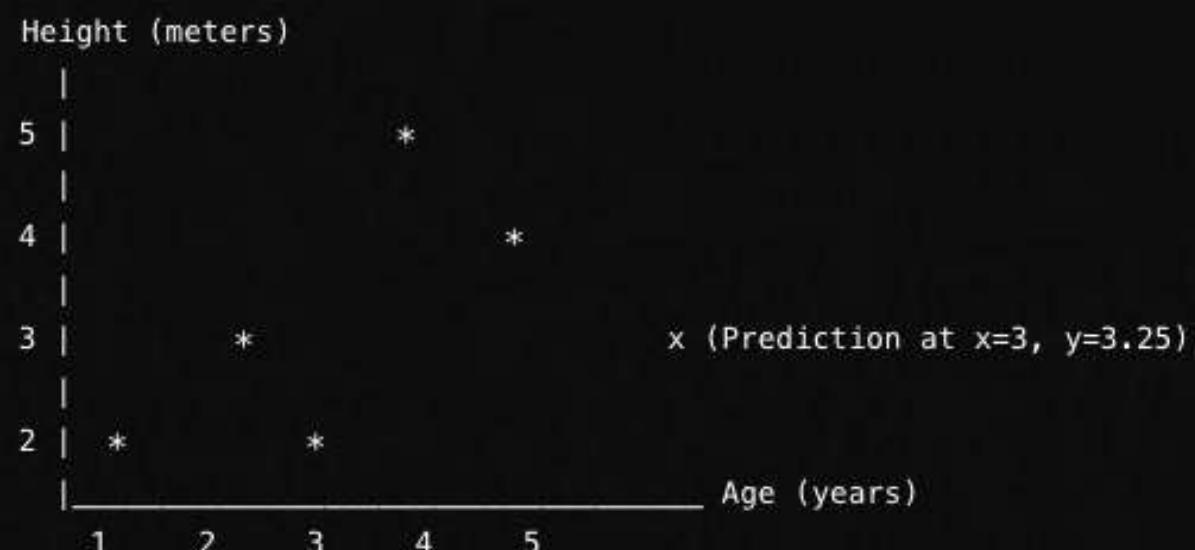
$$X^T W y = \begin{bmatrix} 12.0591 \\ 46.2269 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}$$

Thus, the fitted local regression line is:

$$y = 0.5x + 1.5$$





Legend:

- * Original data points
- Locally Weighted Regression curve
- x Prediction at age 3 (height = 3.25 meters)

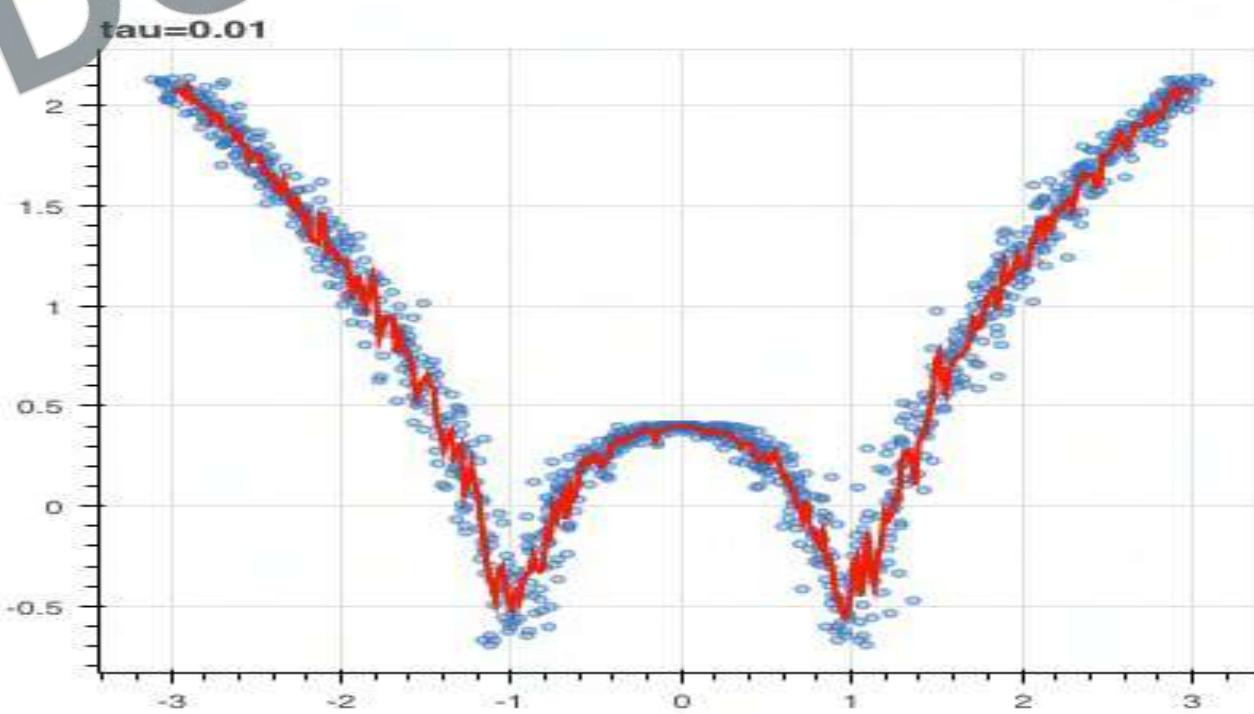
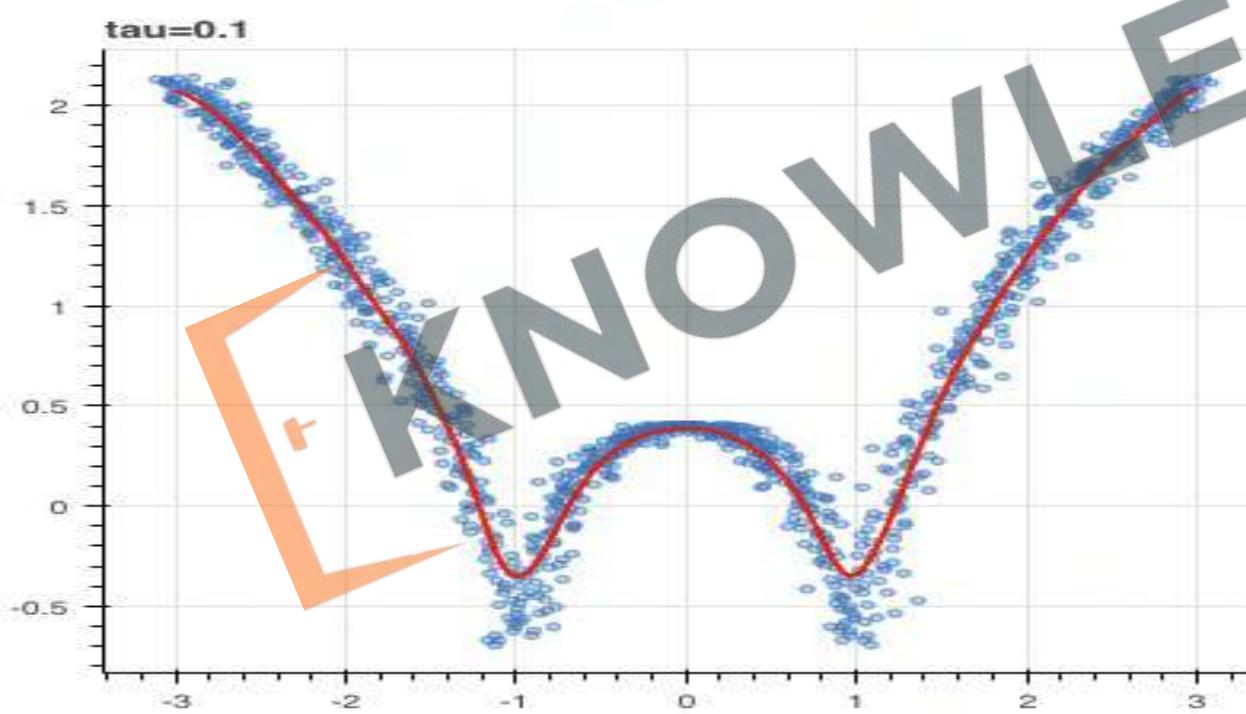
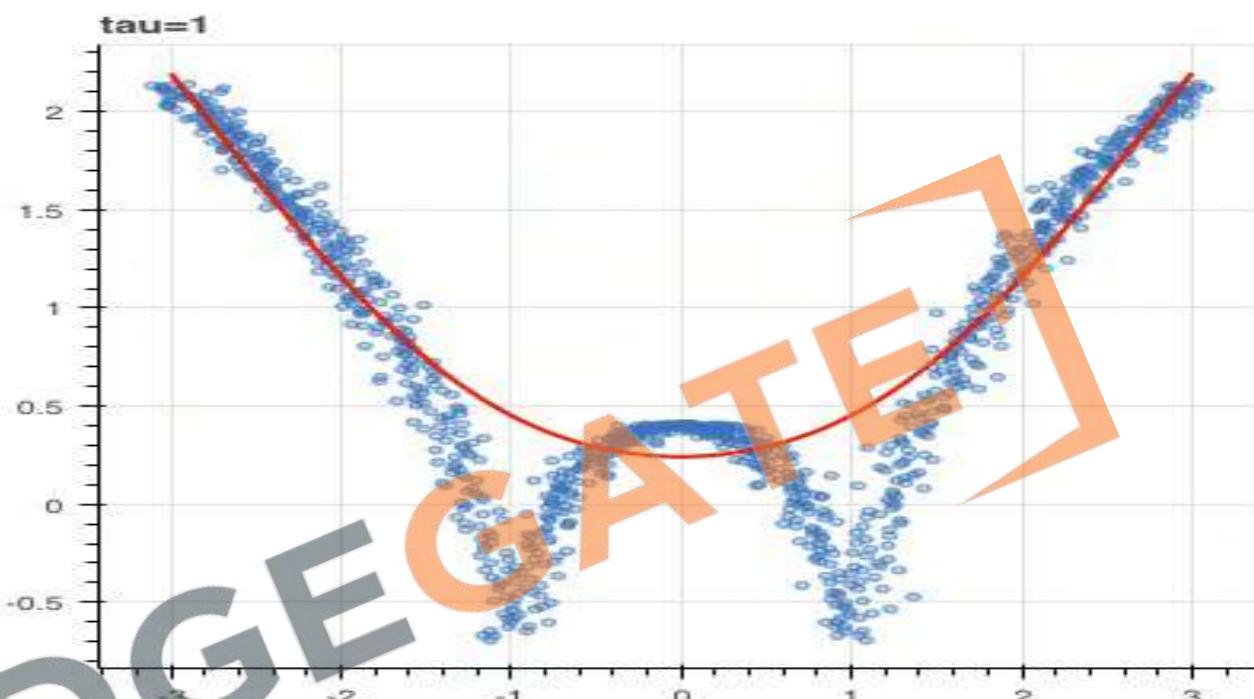
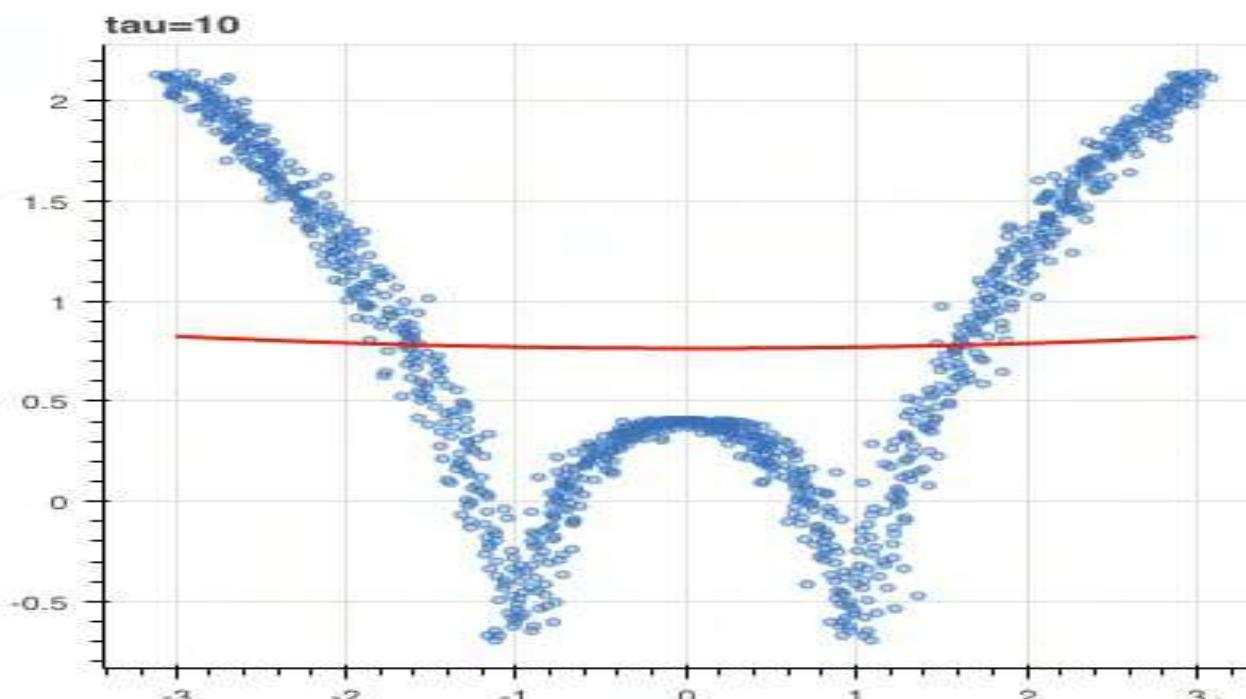


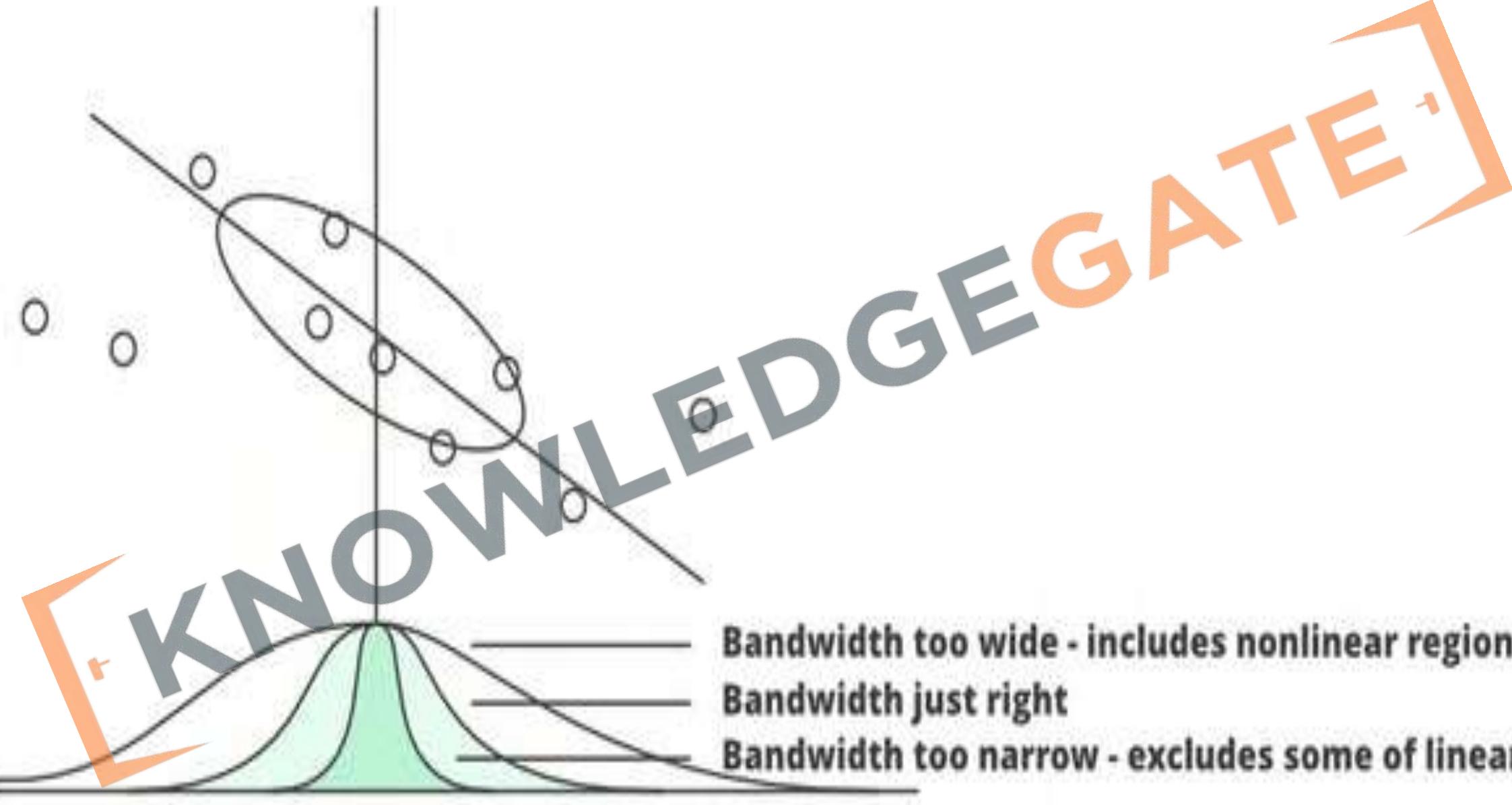
Legend:

- * Original data points
- Locally Weighted Regression curve
- x Prediction at age 3 (height = 3.4 meters)

$Z=2$

$Z=3$



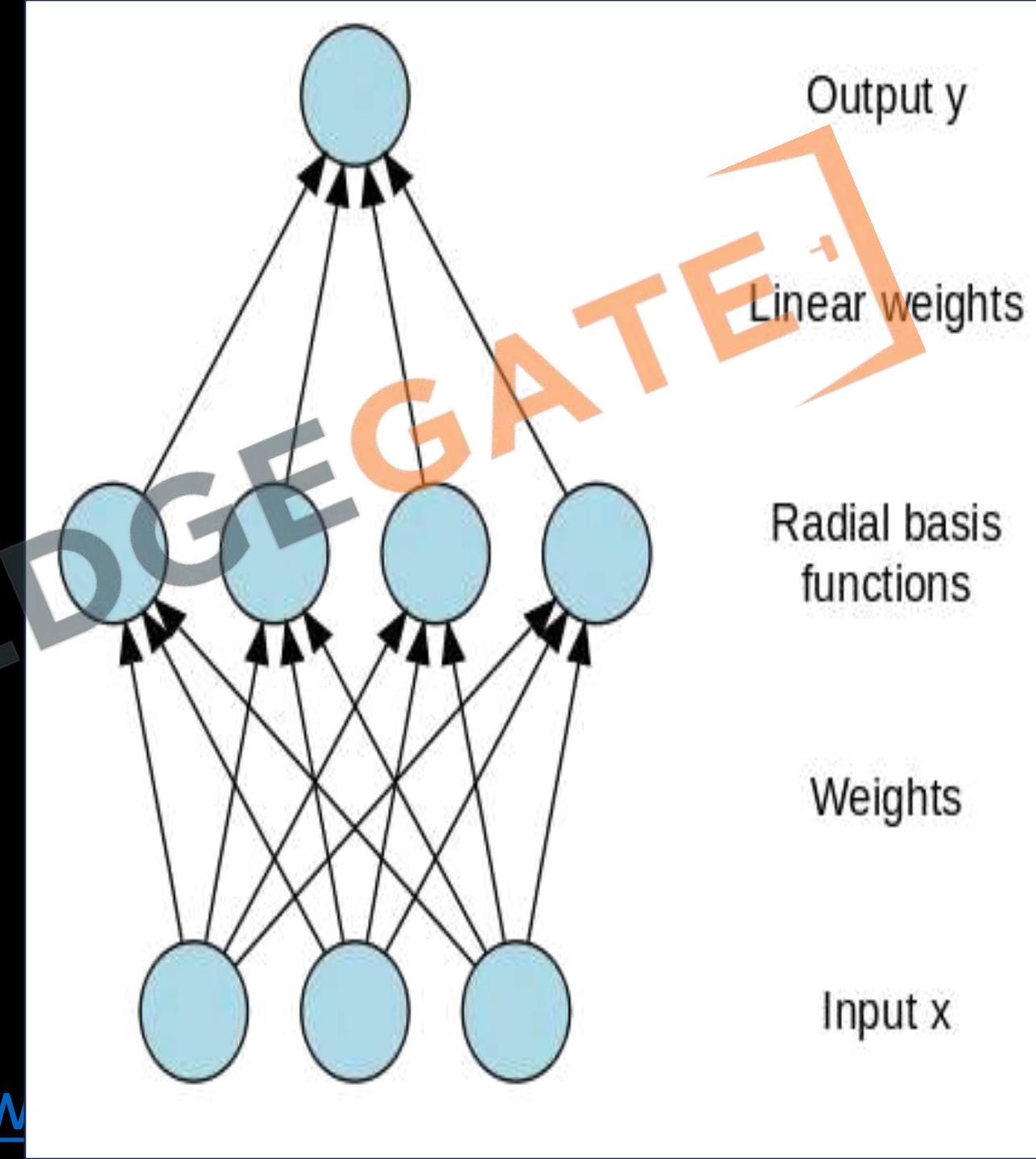


- **History and Development:**
 - Emerged from mid-20th-century non-parametric regression techniques to handle nonlinear relationships without assuming a global model.
 - Formalized in the 1970s and 1980s by statisticians like Cleveland and Devlin through methods like LOWESS and LOESS, introducing key innovations such as kernel functions and bandwidth selection.
- **Modern-Day Applications:**
 - Used in diverse fields like economics, bioinformatics, robotics, and environmental science for forecasting, gene expression analysis, trajectory planning, and modeling environmental variables.
 - Advantages include flexibility in modeling complex, nonlinear relationships and local adaptation to data variations.
- **Current Trends and Research:**
 - Ongoing research focuses on improving efficiency, especially for high-dimensional data, and integrating LWR with deep learning.
 - Trends include real-time applications, big data, and edge computing, with modern toolkits like Python's statsmodels and scikit-learn and R's loess and gam facilitating implementation.
<http://www.knowledgegate.in/gate>

Radial Basis Functions

- Radial Basis Functions are a powerful tool for modeling complex, nonlinear relationships in data. They originated from mathematical interpolation methods and have evolved to become a key component in various modern applications, particularly in machine learning and data science. RBFs offer flexibility, smoothness, and local adaptation, making them suitable for a wide range of problems where traditional linear models fall short.

<http://www.know>



Problem They Solve:

- Function Approximation, Interpolation, and Machine Learning: RBFs approximate complex functions and model intricate patterns in data, create smooth surfaces for precise interpolation (crucial in geostatistics for predicting unsampled values), and are used in machine learning for classification and regression, effectively handling nonlinear relationships between inputs and outputs.

History:

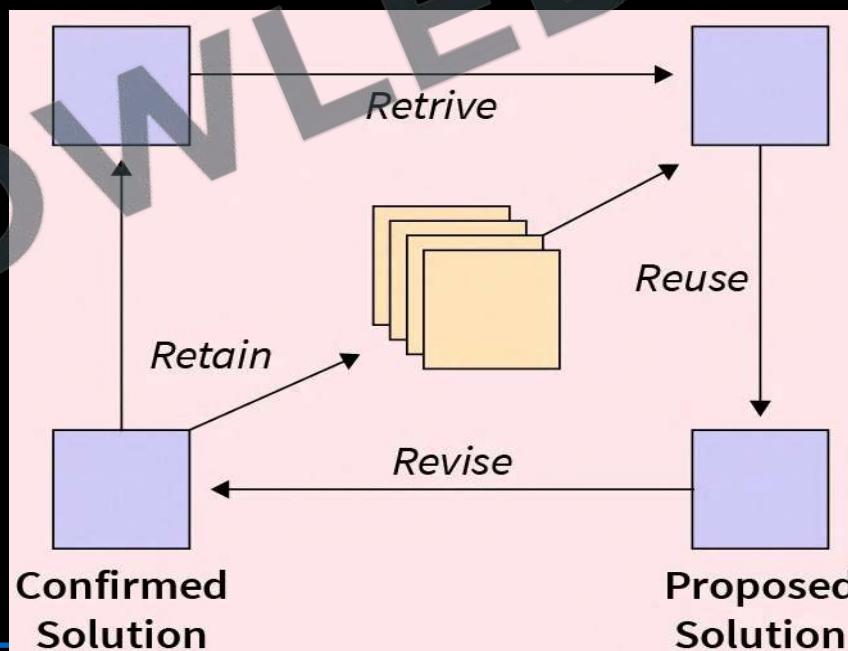
- **Origins and Development:** Rooted in mid-20th century interpolation and approximation theory, RBFs were formalized in the 1970s and 1980s by researchers like Hardy. They became popular for smooth interpolations and handling scattered data points, with RBF networks extending their applications to machine learning and artificial intelligence.

- **Applications:**
 - **Machine Learning:** RBFs are used in Radial Basis Function Networks (RBFNs) for classification, regression, and time-series prediction.
 - **Image and Signal Processing:** RBFs help in tasks like image reconstruction, noise reduction, and signal interpolation.
 - **Geostatistics:** Used for spatial interpolation and modeling, such as predicting pollution levels at unmonitored locations.
 - **Robotics:** Applied in path planning and control systems, where smooth and adaptive control strategies are required.
 - **Finance:** Used for modeling financial data and predicting market trends.

- **Problem and Significance:**
 - RBFs are used to predict air pollution levels across a city by interpolating data from a limited number of monitoring stations, providing smooth and accurate estimates even in unsampled locations.
- **Why Use RBFs:**
 - RBFs excel in smooth interpolation, adapting to local variations due to pollution sources, and handling nonlinear relationships between pollution levels and factors like traffic and industrial activity.
- **Outcome and Benefits:**
 - The resulting pollution maps help city planners and environmental agencies make informed decisions on pollution control, and increase public awareness, encouraging actions to reduce pollution.

Case-Based Learning in Machine Learning

- **History:** Originating from cognitive science in the 1980s, Case-Based Learning (CBL) in machine learning was inspired by the way humans use past experiences to solve new problems, formalized through early AI research on analogical reasoning.
- **Function or Use:** CBL solves new problems by retrieving and adapting solutions from similar past cases, making it useful for domains where similar problems recur and historical solutions are available.
- **Advantage:** CBL can handle complex and unstructured problems without needing explicit programming for every scenario, making it flexible and adaptable to a wide range of applications.



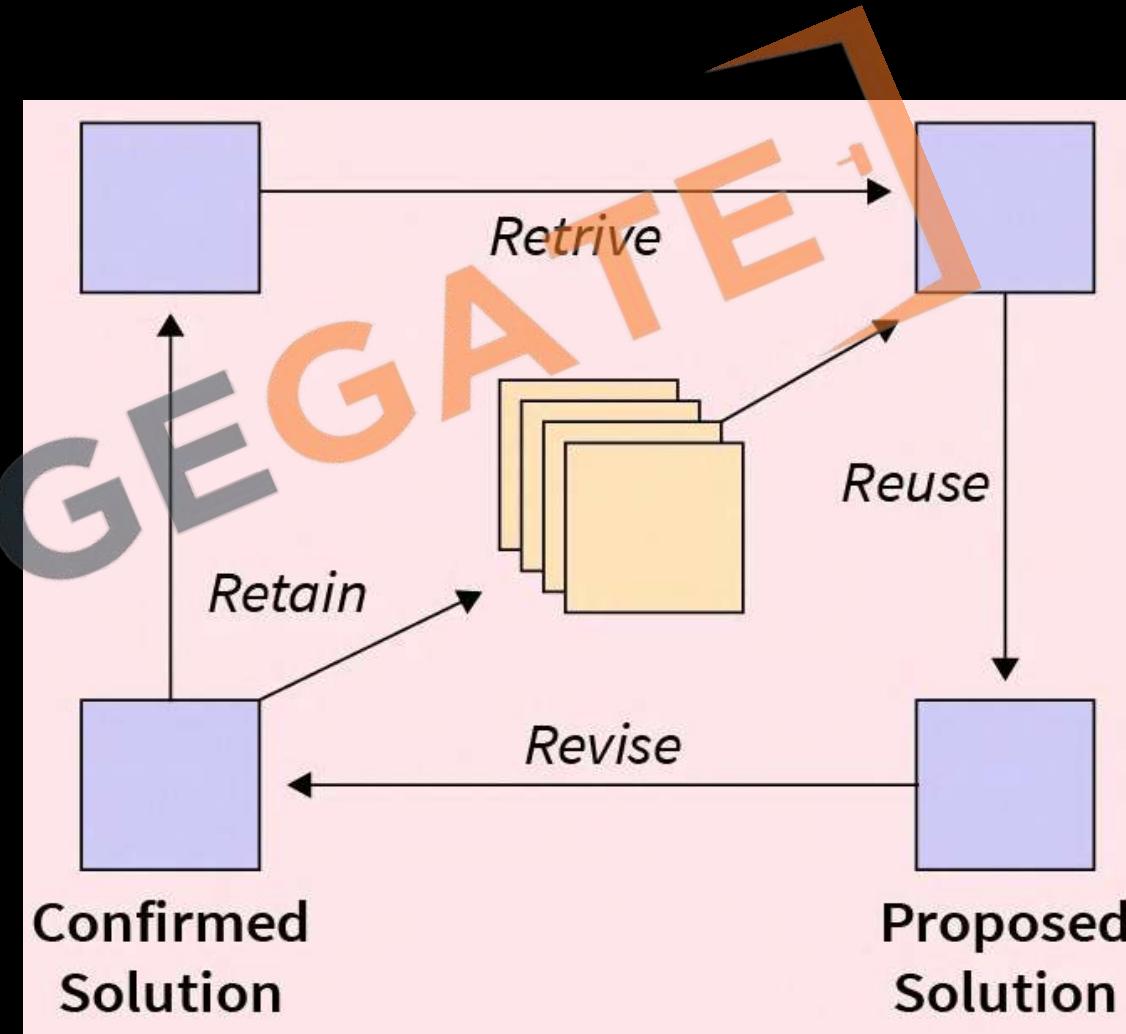
- **Advantage of CBL as a Lazy Problem Solving**: As a lazy learning method, CBL does not require a model to be built or trained beforehand, allowing it to use the most recent data for problem-solving, which can be particularly useful in dynamic environments.
- **Disadvantage or Limitation**: CBL can be computationally expensive and slow for large datasets due to the need to search through a potentially vast case library to find similar past cases.
- **Applications**: CBL is used in various fields such as medical diagnosis, legal reasoning, customer support, and maintenance troubleshooting, where leveraging past cases can significantly improve decision-making and problem-solving.
- **Case-Based Learning Cycle with Different Schemes**: The CBL cycle typically involves retrieving the most relevant cases, reusing the information to solve the current problem, revising the proposed solution if necessary, and retaining the new solution for future use, often implemented through schemes like the nearest neighbor, template-based, and knowledge-based approaches.

Medical Diagnosis Support System

- **Scenario:** A hospital uses a Case-Based Learning system to assist doctors in diagnosing rare diseases. The system contains a large database of past medical cases, including patient symptoms, diagnostic tests, treatments, and outcomes.

- **How It Works:**

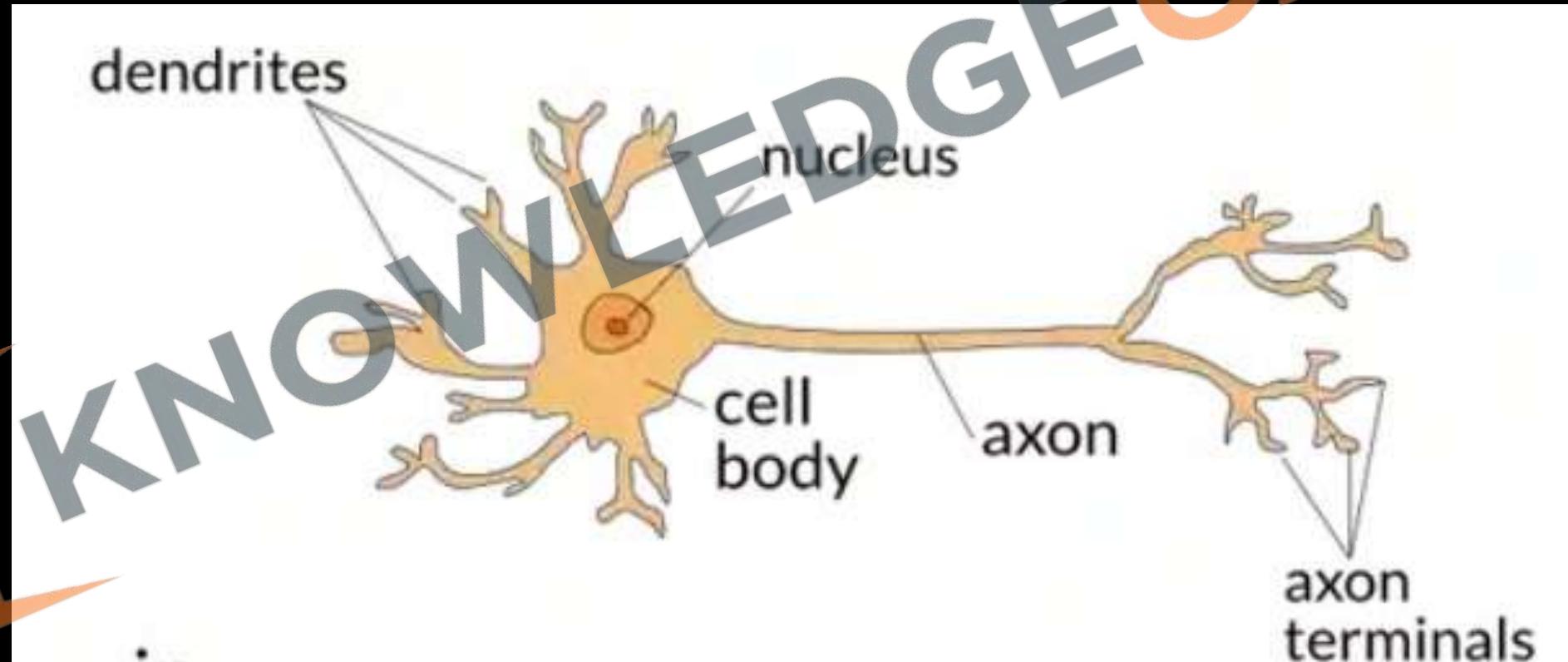
- **Retrieve:** When a new patient presents symptoms that are difficult to diagnose, the doctor inputs these symptoms into the system.
- **Reuse:** The CBL system searches its database for past cases with similar symptom profiles.
- **Revise:** It retrieves the most relevant cases and suggests potential diagnoses and treatment plans that were successful in those past cases.
- **Retain:** After the doctor reviews and possibly adjusts the suggested diagnosis and treatment, the final decision is added to the system's database as a new case.



- **(UNIT-4: ARTIFICIAL NEURAL NETWORKS)** ARTIFICIAL NEURAL NETWORKS - Perceptron's, Multilayer perceptron, Gradient descent & the Delta rule, Multilayer networks, Derivation of Backpropagation Algorithm, Generalization, Unsupervised Learning - SOM Algorithm and its variant; DEEP LEARNING - Introduction, concept of convolutional neural network, Types of layers - (Convolutional Layers, Activation function, pooling, fully connected), Concept of Convolution (1D and 2D) layers, Training of network, Case study of CNN for eg on Diabetic Retinopathy, Building a smart speaker, Self-deriving car etc.

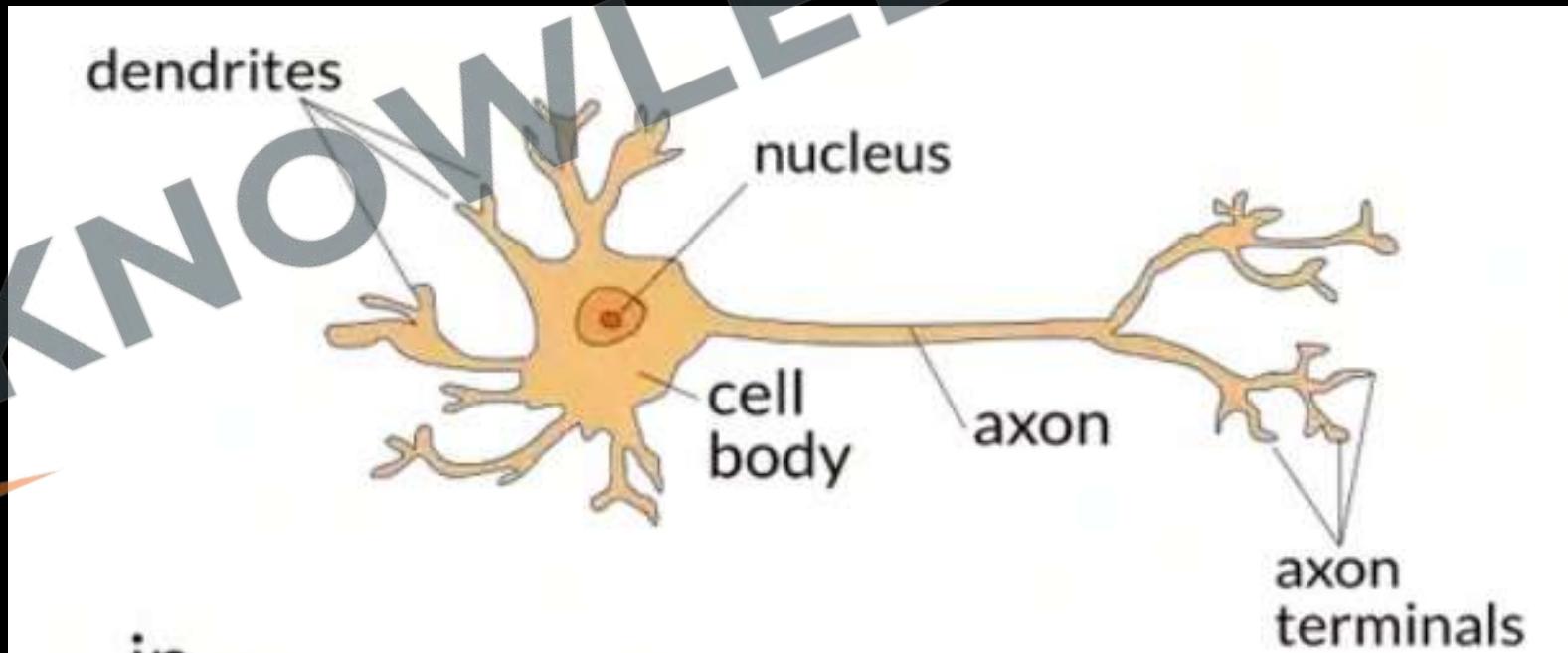
Biological Neurons and Their Structure

- Neuron, Also Called Nerve Cell:
 - Basic working unit of the brain.
 - Specialized cells responsible for sending and receiving signals from the brain.



Structure of a Biological Neuron:

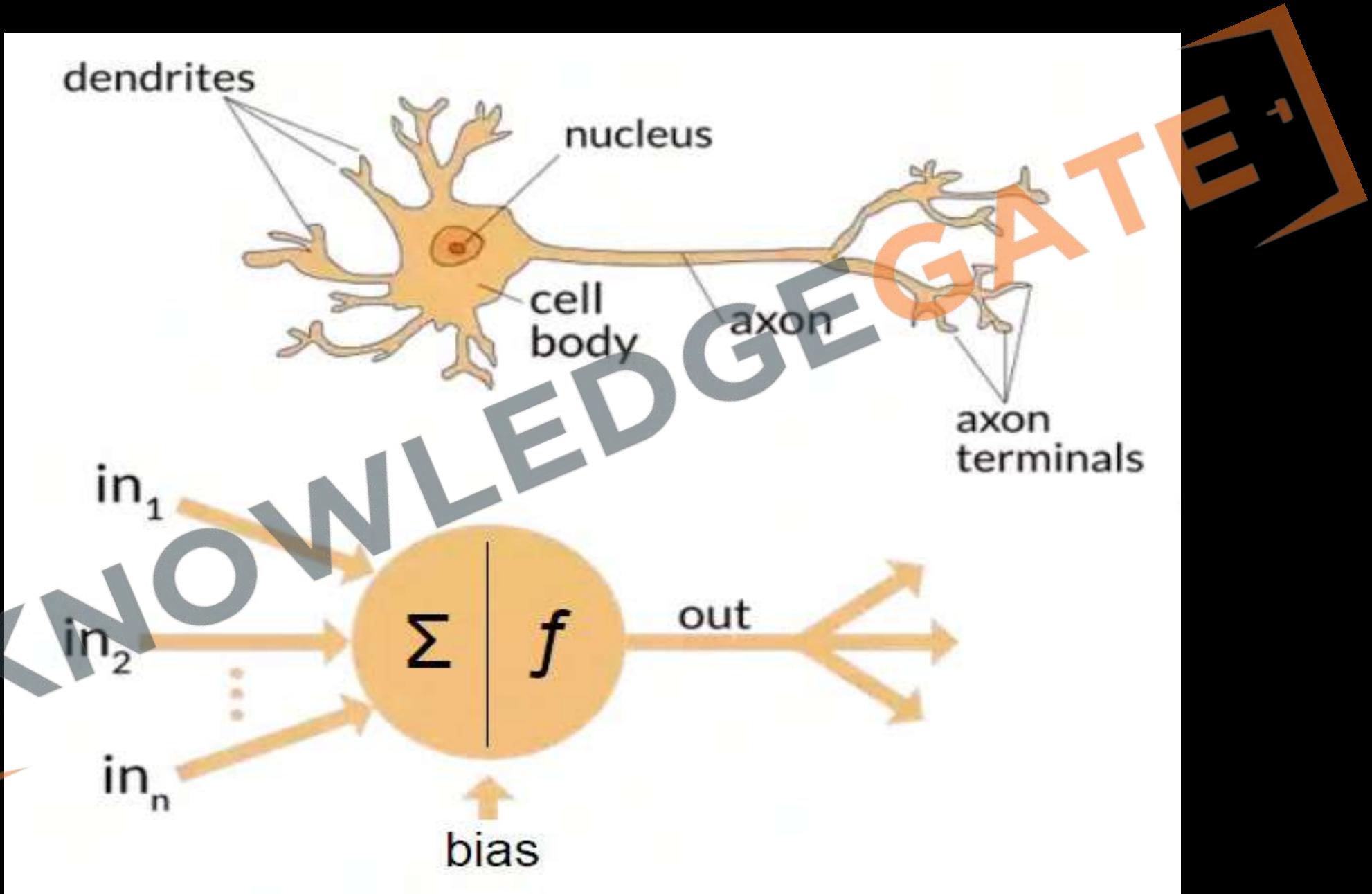
- **Dendrite (Input):**
 - Receives signals from other neurons.
- **Cell Body (Summing Unit):**
 - Sums up all the incoming input signals.
 - Contains the threshold unit.
- **Axon (Output):**
 - When the sum reaches a threshold level, the neuron generates an output signal which travels down to other neurons.



Types of Biological Neurons:

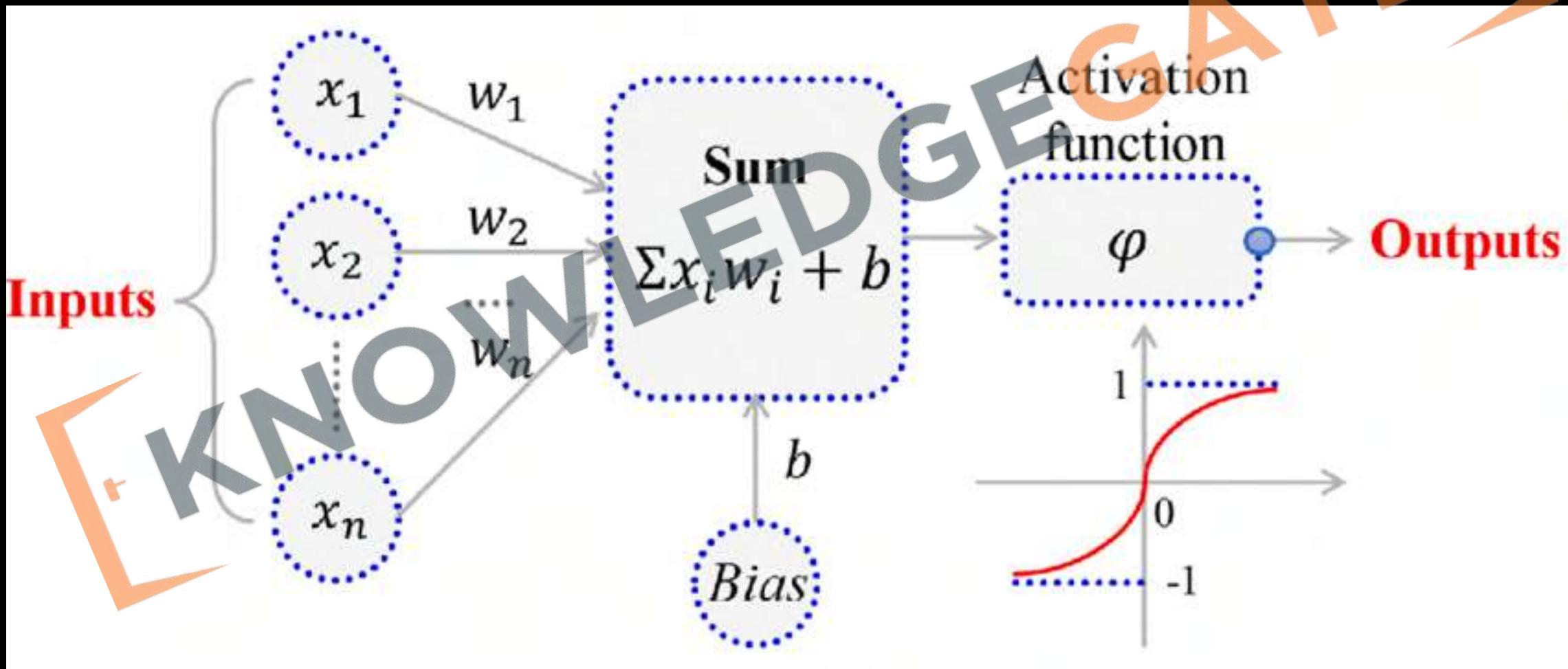
- **Sensory Neurons**:
 - Handle taste, smell, hearing, seeing, and feeling.
- **Motor Neurons**:
 - Control voluntary and involuntary movements of muscles and organs.
- **Interneurons**:
 - Intermediate neurons found in the spinal cord and brain.
 - Pass signals from sensory neurons to motor neurons.

Artificial Neuron



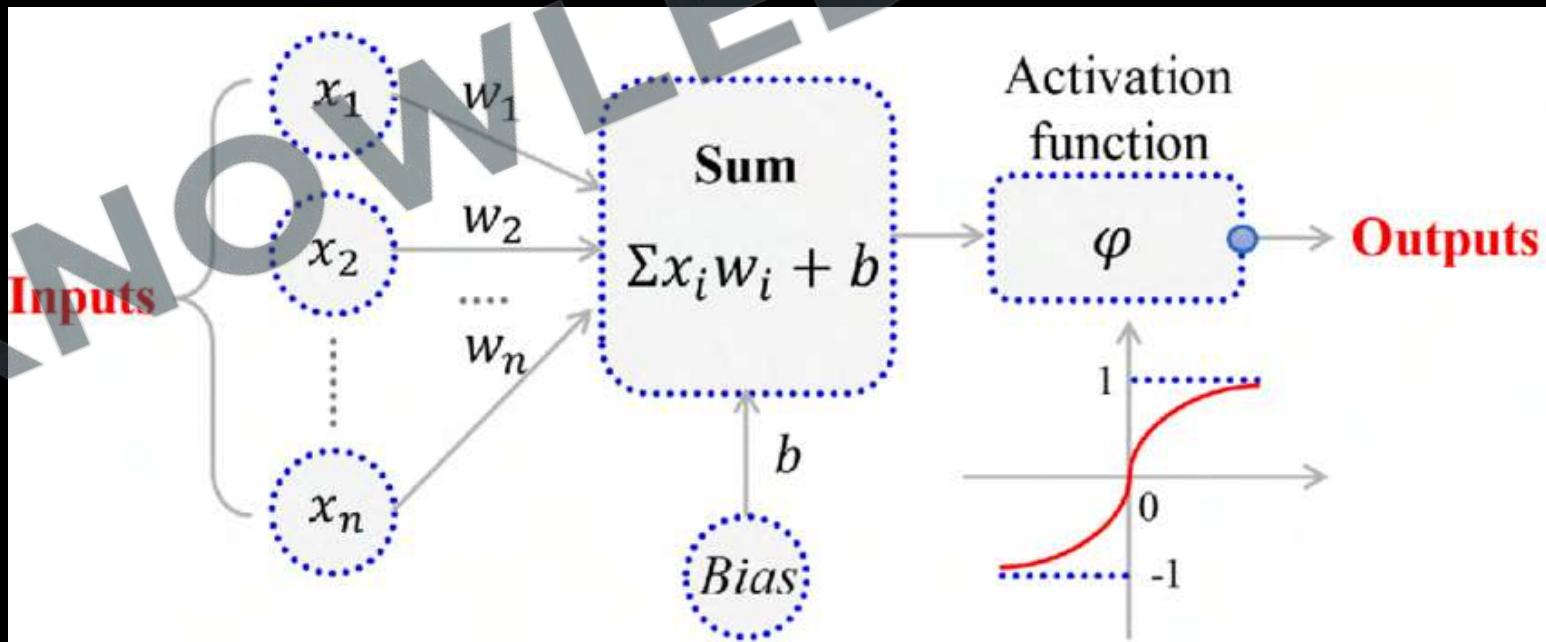
Artificial Neuron

- An artificial neuron is a mathematical function based on the model of biological neurons.



The model of an artificial neuron has five components:

- **Input Values/Input Layer:**
 - Input values are passed to a neuron using this layer. Similar to dendrites in biological neurons.
- **Synapses/Weighted Connection Links:**
 - Each connection link is characterized by a weight or strength (w_i). Each input (x_i) is multiplied by its weight to get the synaptic strength.
- **Summing Function:**
 - The summing function takes the sum of all weighted connection links: $\sum w_i x_i$. A bias value (b) is also added to this weighted sum.
- **Activation Function:**
 - The activation function decides whether or not a neuron is fired based on the output value produced.
- **Output/Output Layer:**
 - The output layer gives the final output of a neuron which can then be passed to other neurons in the network.

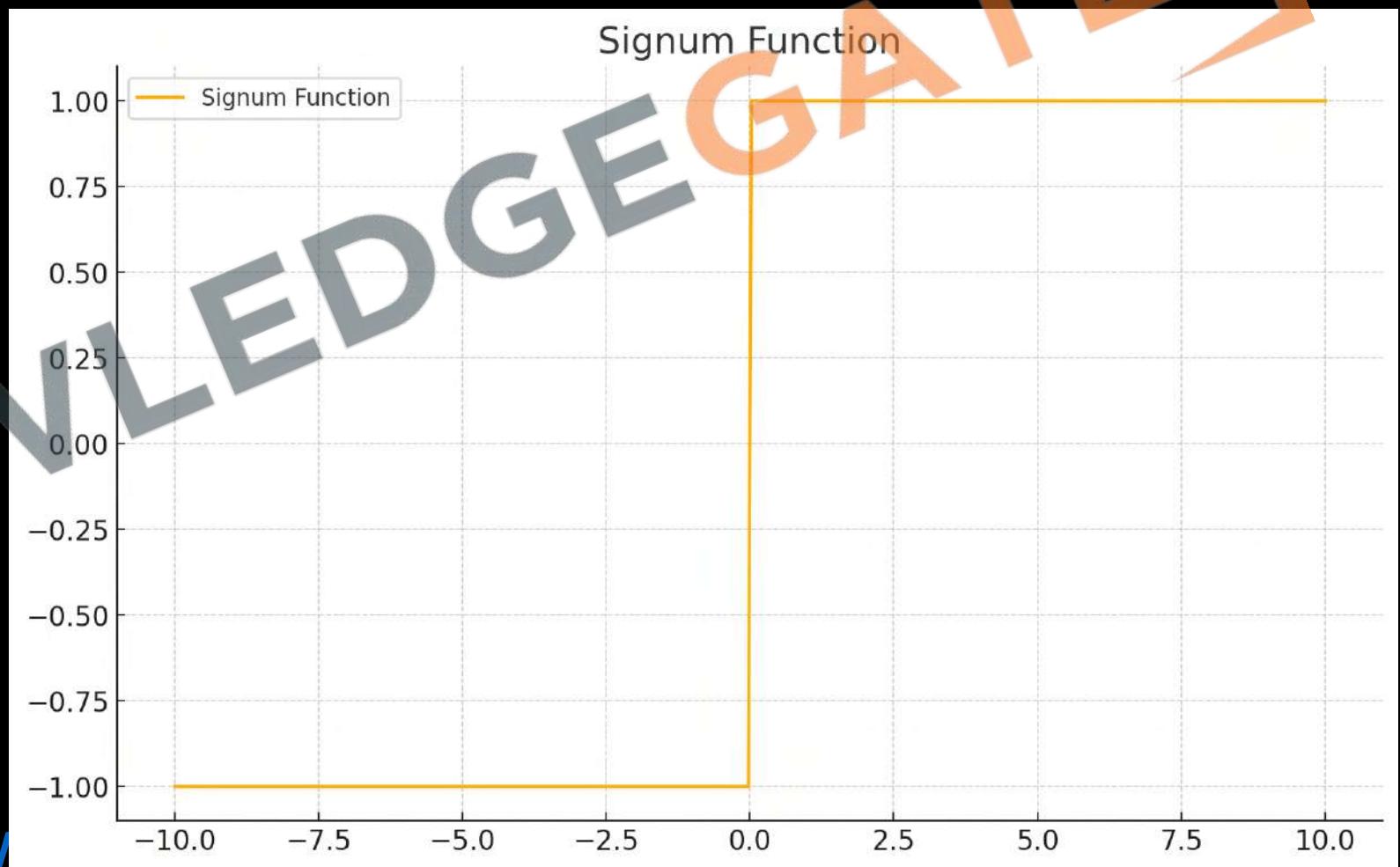


Activation Functions in Neural Networks

- Activation functions are mathematical functions that determine the output of a neural network. They decide whether a neuron should be activated ("fired") or not, introducing non-linearity into the model, which is crucial for handling complex data.

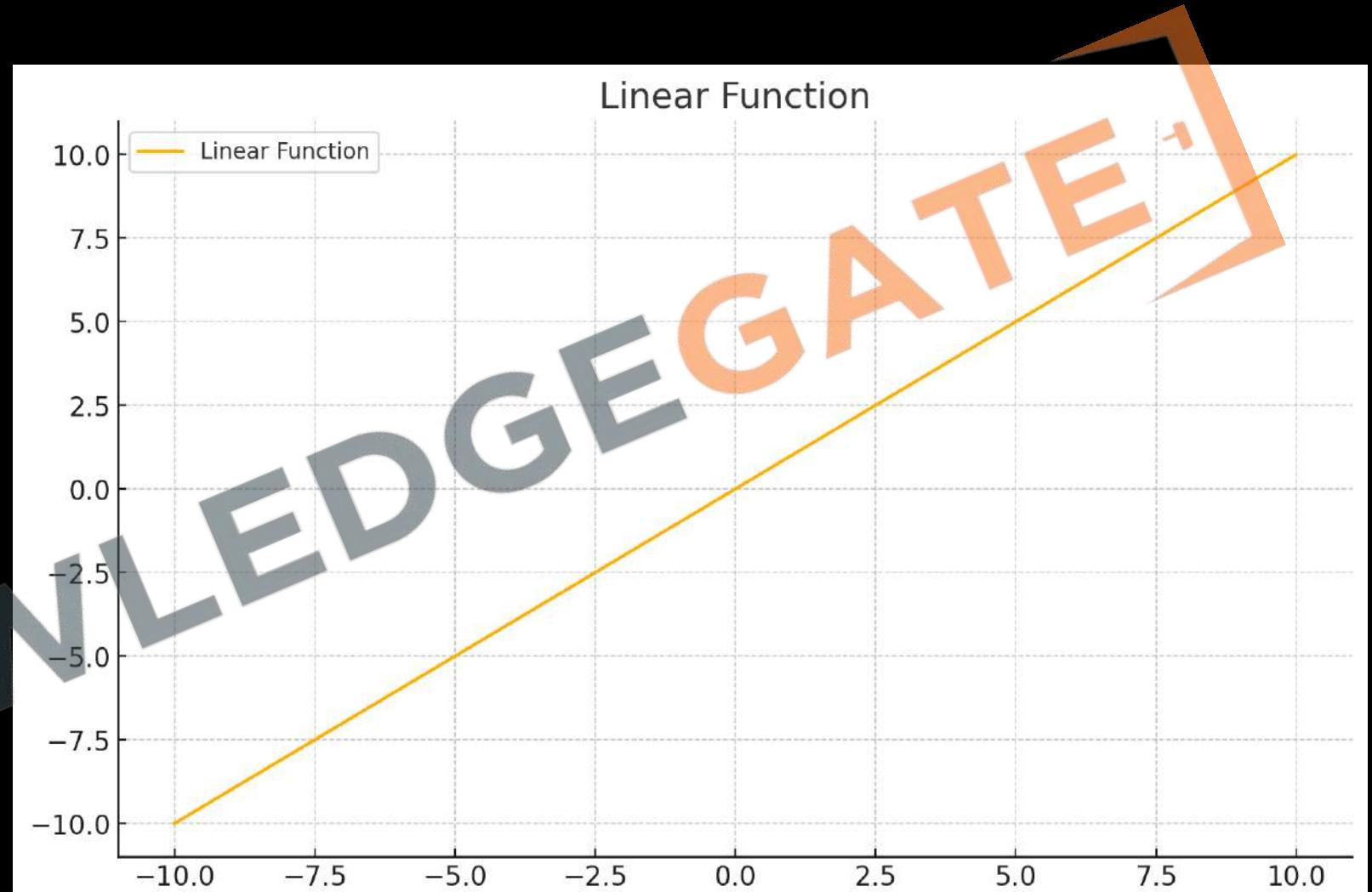
Step Function:

- $$z = f(y) = \begin{cases} 0, & y \leq 0 \\ 1, & y > 0 \end{cases}$$



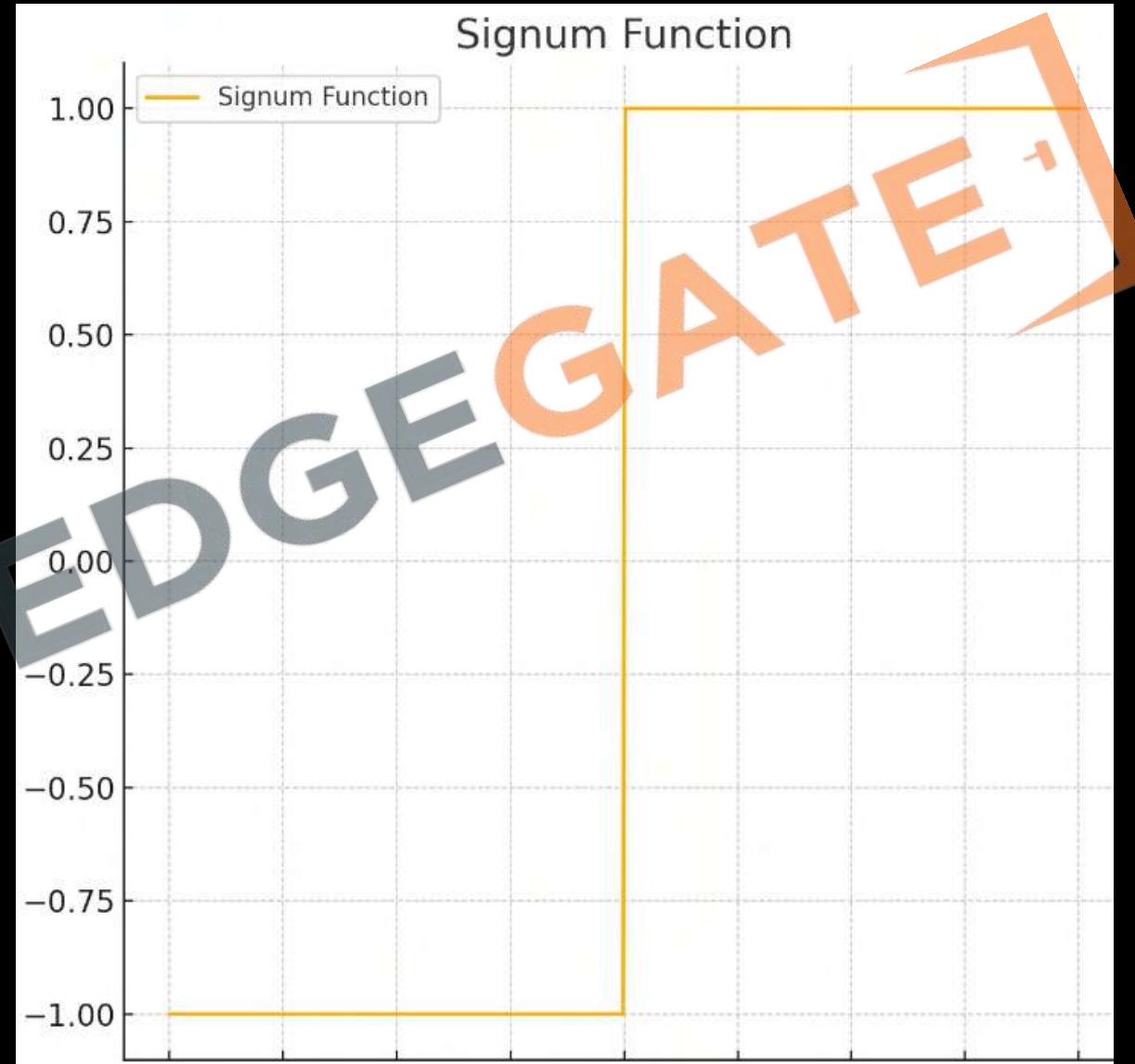
Linear Function:

- $z = f(y) = y$



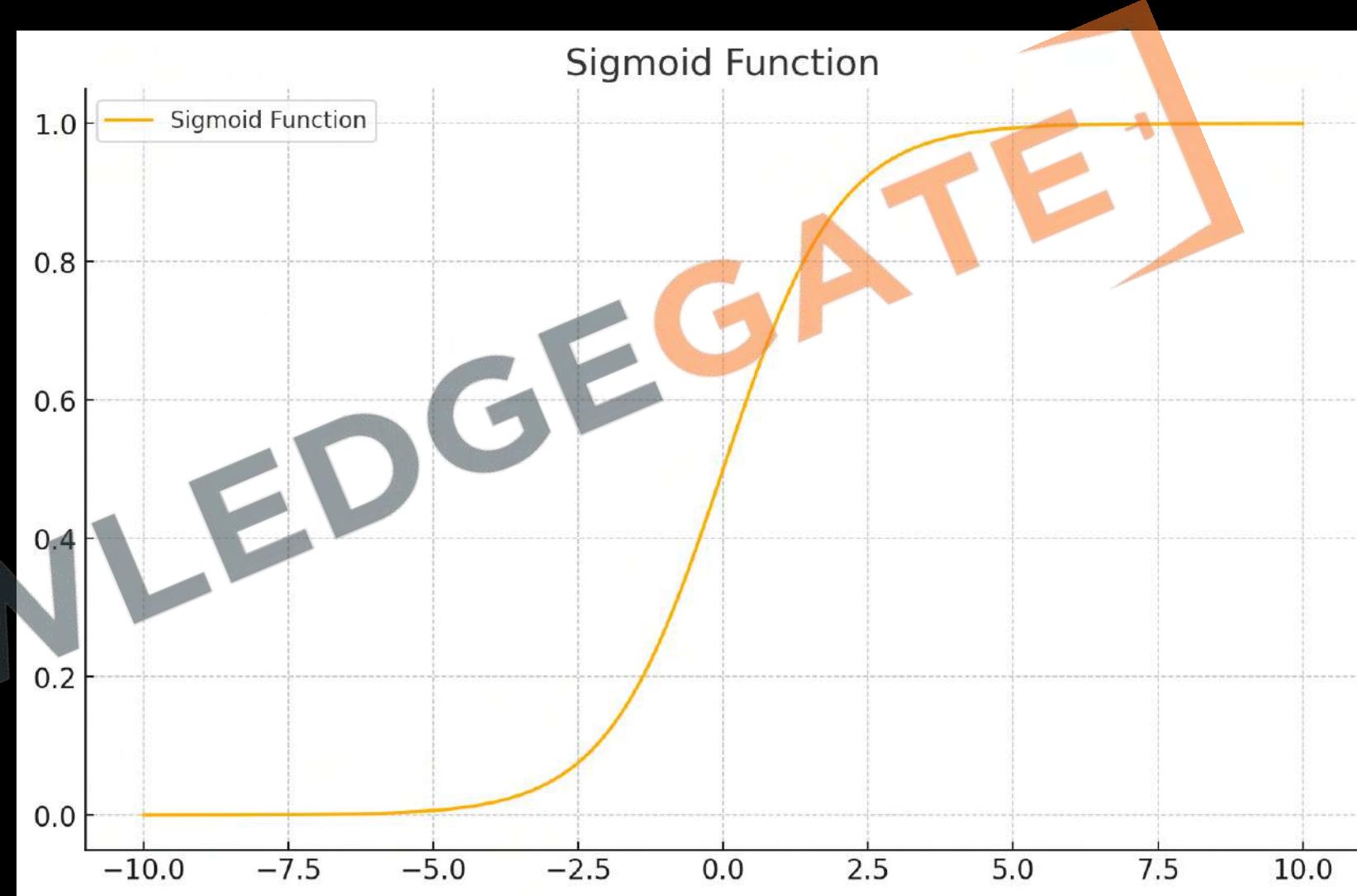
Signum Function:

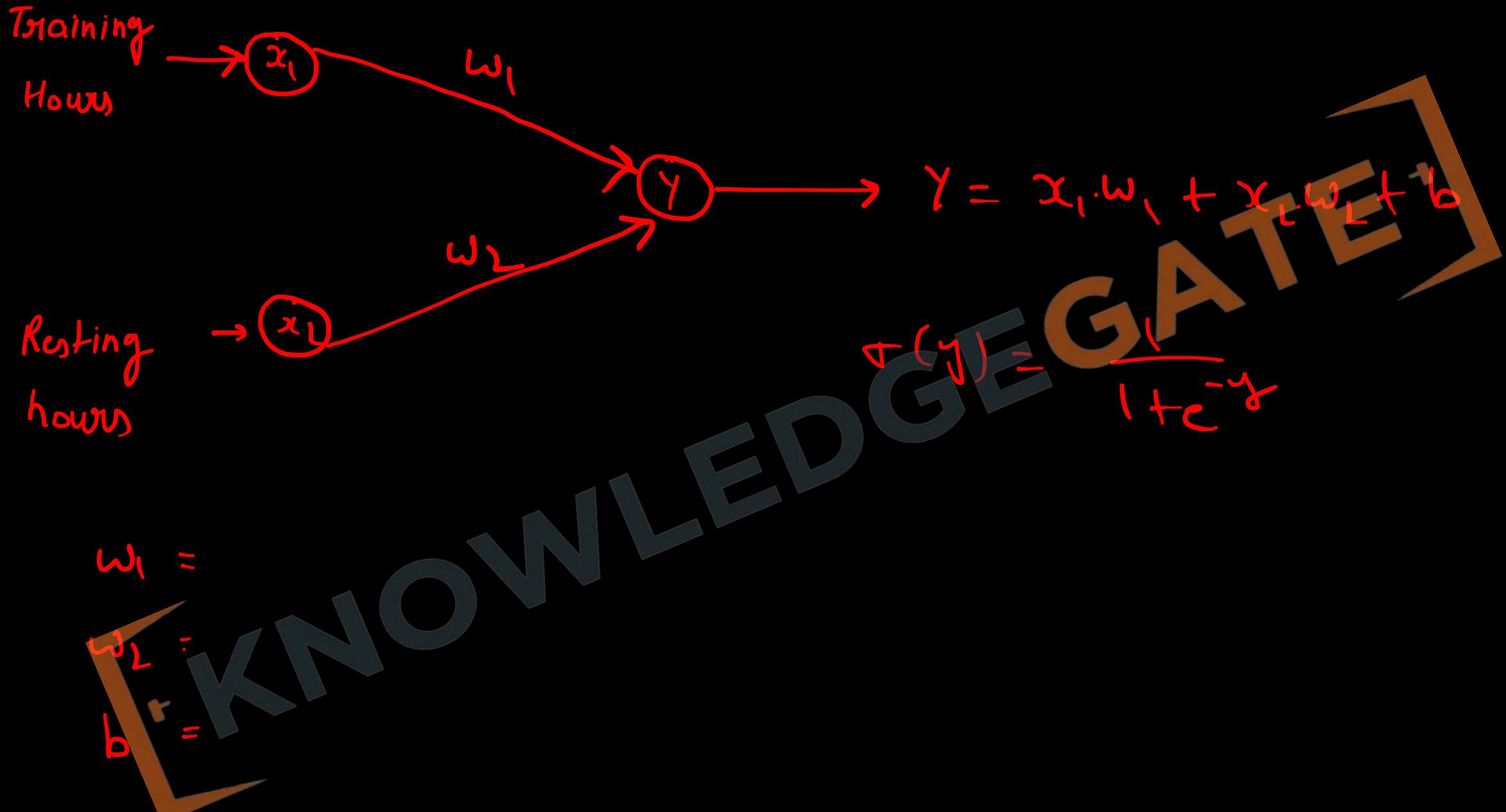
- $$z = f(y) = \begin{cases} -1, & y < 0 \\ 0, & y = 0 \\ 1, & y > 0 \end{cases}$$



Sigmoid (Logistic) Function:

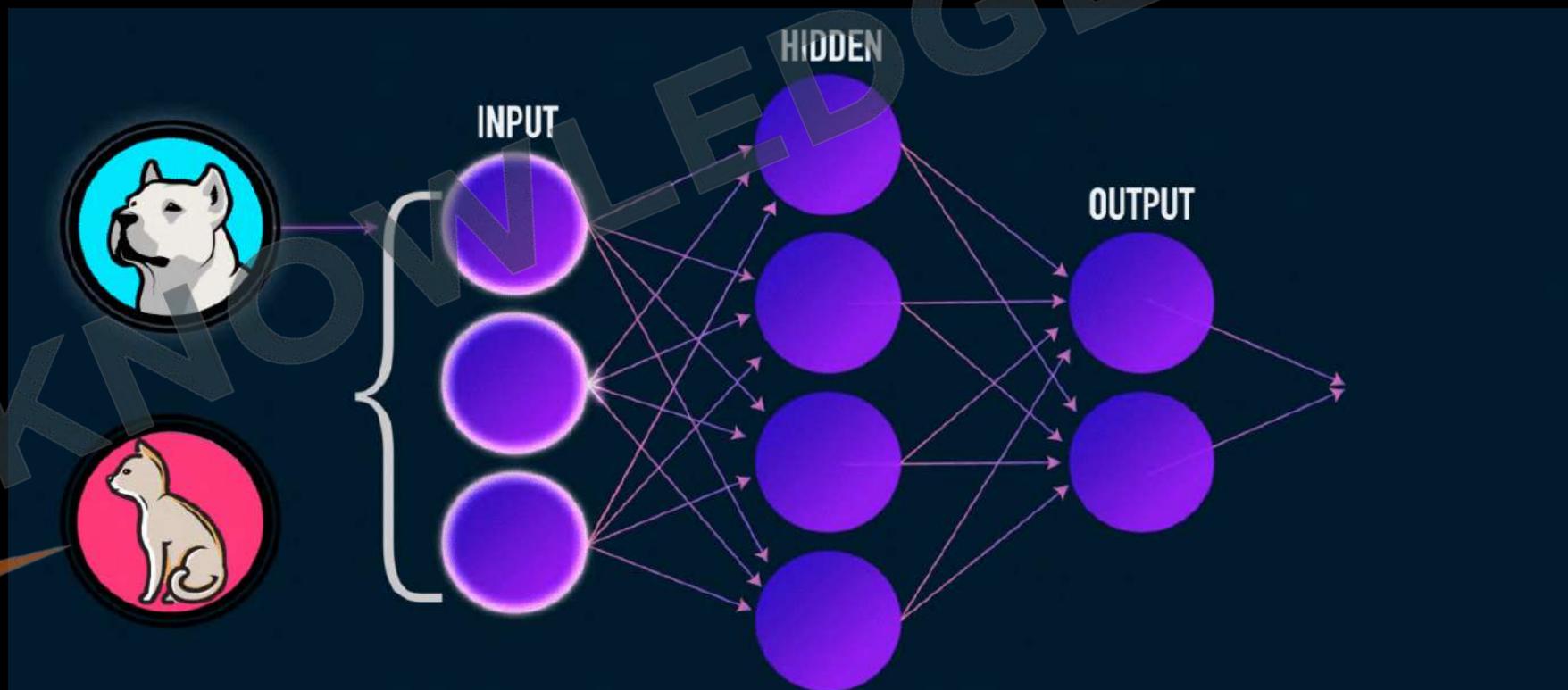
- $z = f(y) = \frac{1}{1+e^{-y}}$





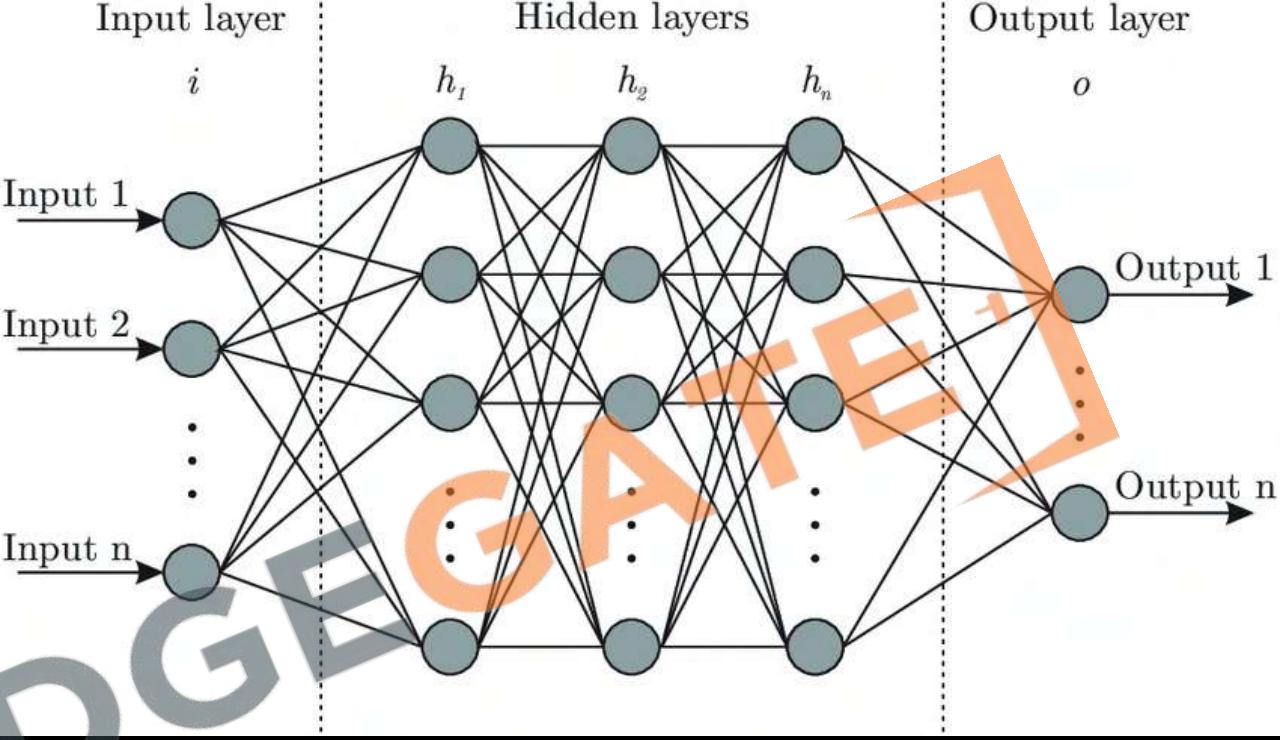
Artificial Neural Network (ANN)

- An Artificial Neural Network (ANN) is a computing system designed to simulate the way the human brain analyzes and processes information.
- It is a data processing system consisting of a large number of interconnected processing elements called artificial neurons (nodes). These neurons are connected through connection links.
- ANNs have self-learning capabilities, allowing them to improve as more data becomes available.



Architecture of ANN:

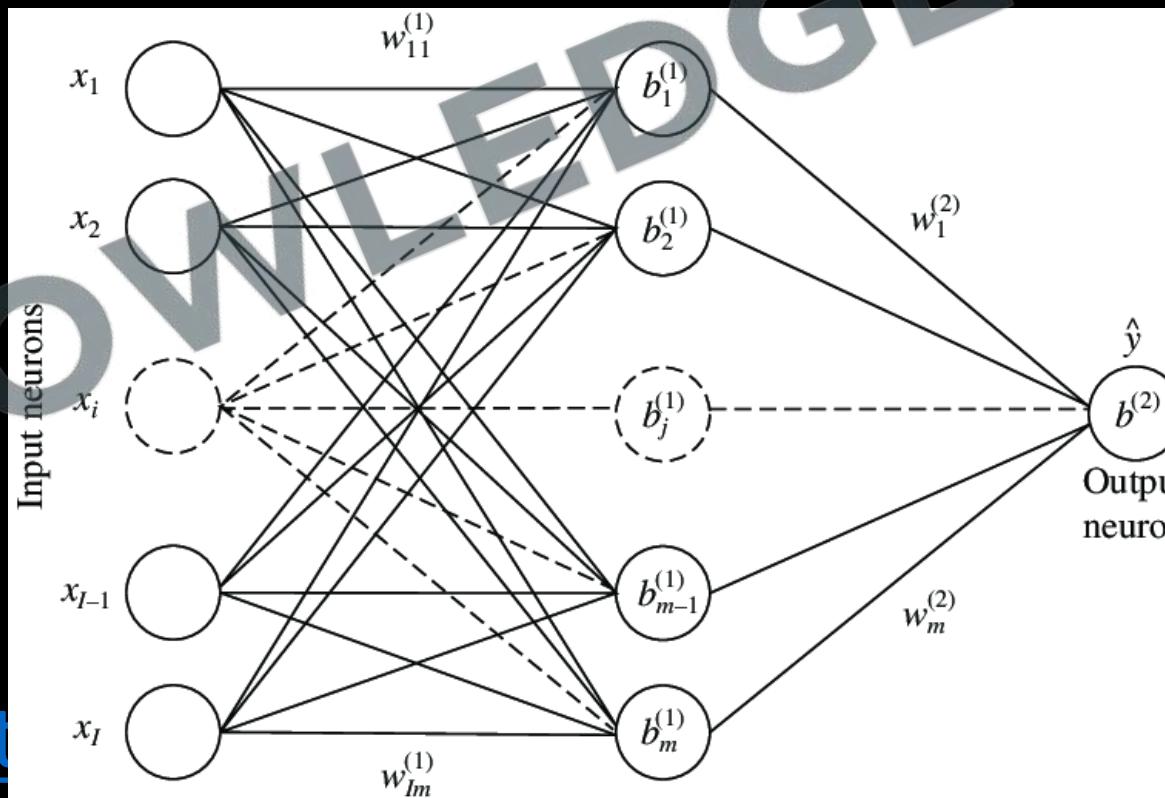
- Input Layer:
 - Contains nodes (artificial neurons) that receive input from the outside world.
- Output Layer:
 - Contains nodes that provide the output of the network.
- Hidden Layers:
 - Located between the input and output layers.
 - Comprise hidden nodes or neurons not directly visible to the external system.
 - Hidden layers are responsible for intermediate computations and feature extraction.
 - There can be zero or more hidden layers depending on the complexity of the problem.
 - One hidden layer is sufficient for many problems, but deeper networks (with more hidden layers) can model more complex patterns.



Classification of Artificial Neural Networks (ANN)

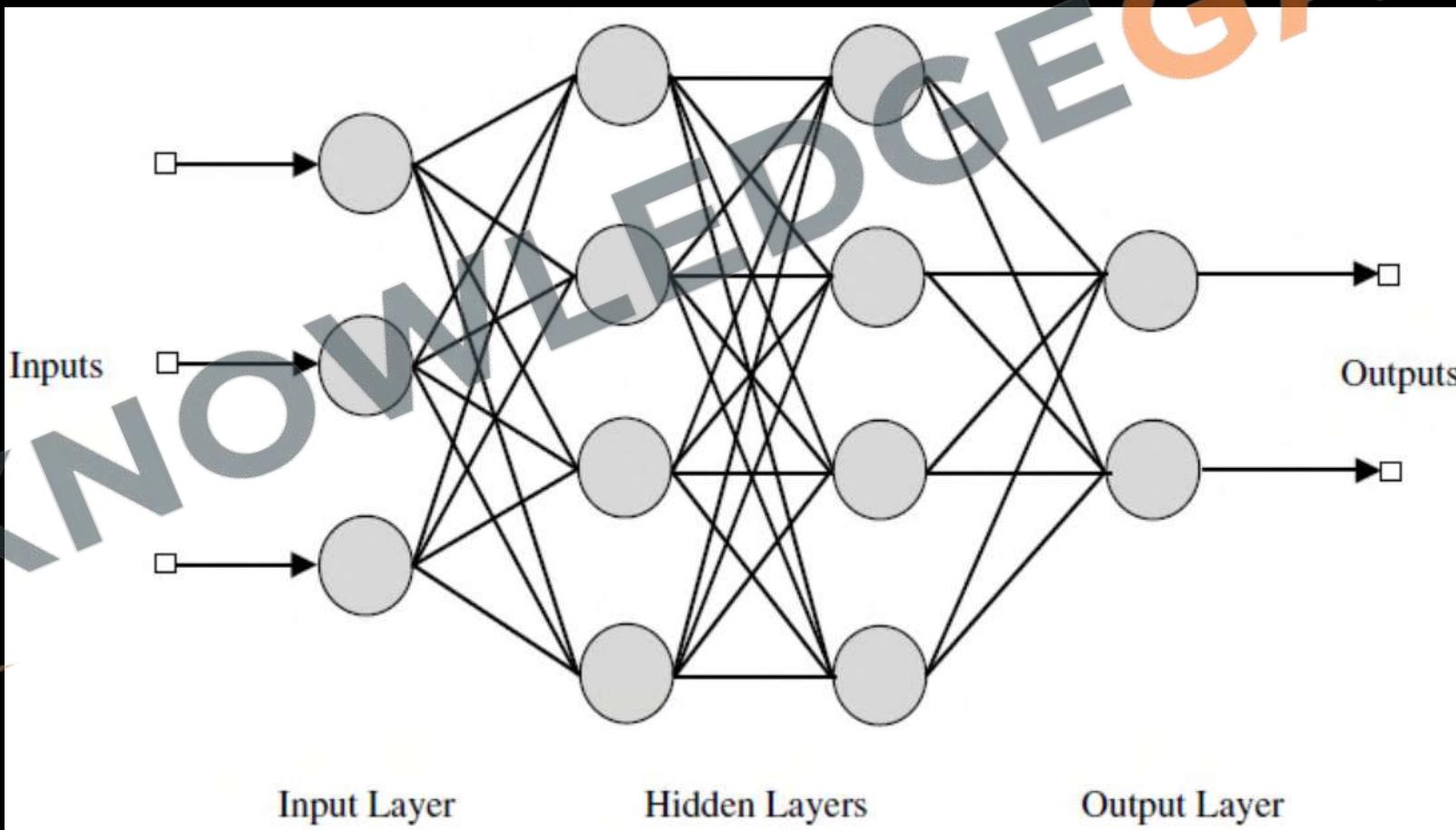
- Single Layer Feed-forward Network:

- Single Layer: Only one computational layer (output layer), making it a single-layer ANN, but it consists of two layers in total (input and output layers).
- Feed-forward Network: Information flows from the input layer to the output layer without any feedback loops.

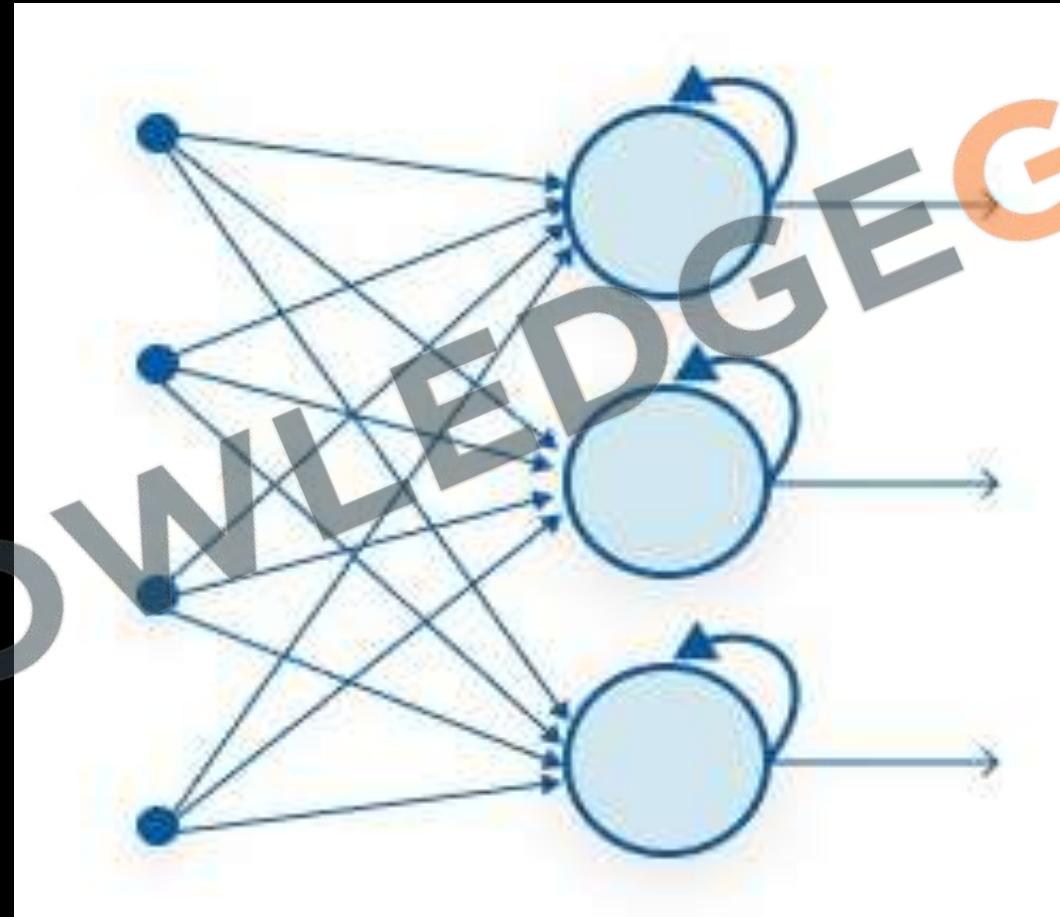


- **Multilayer Feed-forward Network:**

- **Multilayer**: Includes an input layer, one or more intermediate (hidden) layers, and an output layer.
- Hidden layers perform intermediate computations before passing the information to the output layer.



- **Recurrent Network:**
 - These networks include at least one feedback loop, differing from feed-forward architectures. Can be single-layer or multi-layer recurrent networks.



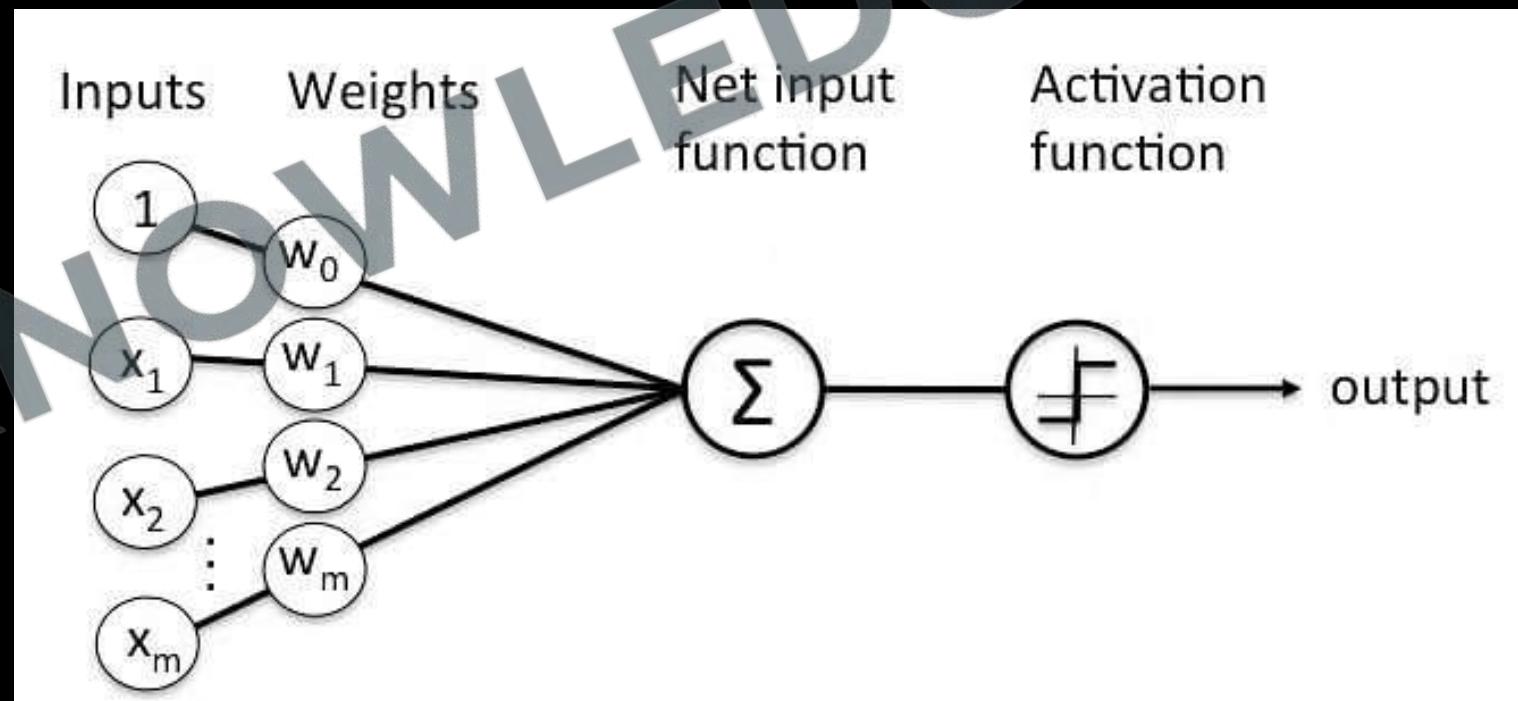
Recurrent Neural Network

<http://>

gate

(Artificial Neuron) Perceptron

- **Definition:** The simplest form of a neural network. Consists of a single neuron with adjustable synaptic weights and bias.
- **History:** Introduced by Frank Rosenblatt in 1957.
- **Components:** One or more inputs, a processing unit, and a single output. Used for supervised learning of binary classifiers. Enables neurons to learn and process elements in the training set one at a time.



Perceptron Function

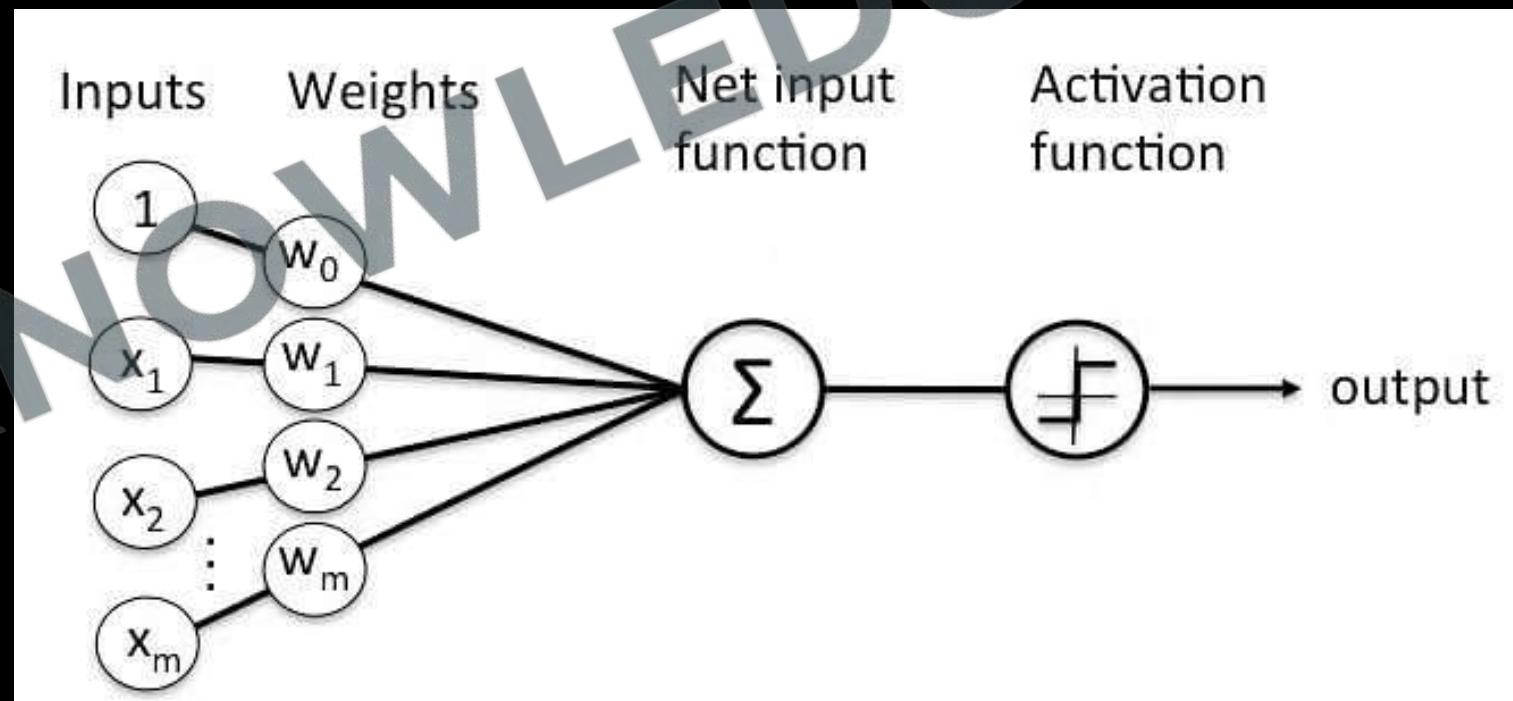
- **Definition:** Maps input x , multiplied with the learned weight coefficient, to generate an output value $f(x)$.
- Formula:
$$f(x) = \begin{cases} 1, & \text{if } wx + b > 0 \\ 0, & \text{if } wx + b \leq 0 \end{cases}$$
- Where:
 - w : Values of weights.
 - b : Bias.
 - x : Values of inputs.
 - n : Number of inputs to the perceptron.

Output:

- The output can be represented as 0 or 1, or -1 and 1, depending on the activation function used. For e.g.

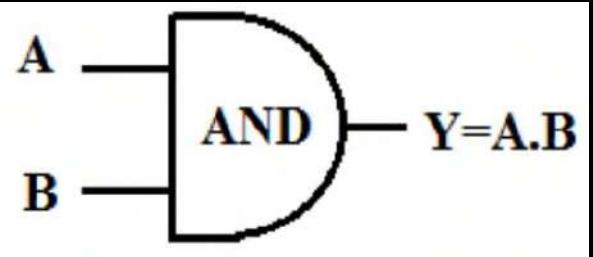
$$f(x_1, x_2, \dots, x_n) = 1 \text{ if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0$$

$$f(x_1, x_2, \dots, x_n) = 0 \text{ if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n \leq 0$$



AND gate using neural networks

- Steps in Perceptron Algorithm:
 - Initialize weight values and bias.
 - Forward propagate the inputs.
 - Check the error.
 - Backpropagate and adjust weights and bias.
 - Repeat for all training examples.



Truth Table		
Input		Output
A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = \begin{cases} 1, & \text{if } wx + b > 0 \\ 0, & \text{if } wx + b \leq 0 \end{cases}$$

$$Y = w_1x_1 + w_2x_2 + b$$

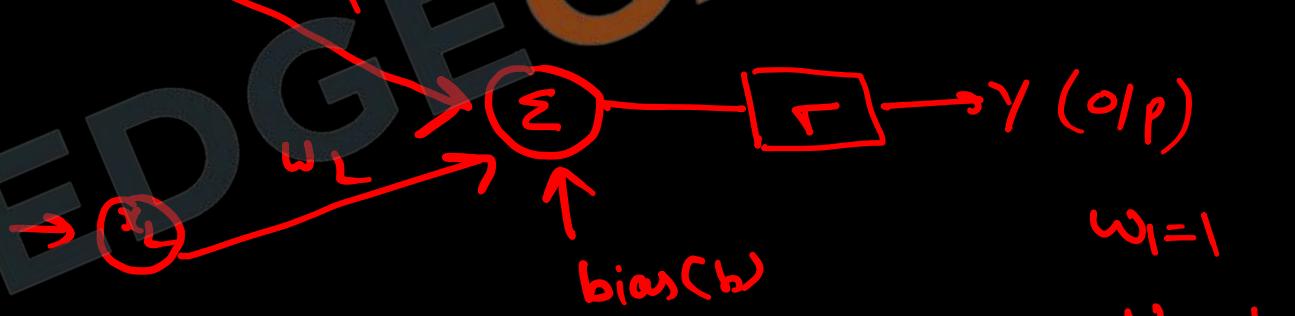
Row 1 $\rightarrow 1 \cdot 0 + 1 \cdot 0 + (-1) = -1 \rightarrow 0 \quad \checkmark$

Row 2 $\rightarrow 1 \cdot 0 + 1 \cdot 1 + (-1) = 0 \rightarrow 0 \quad \checkmark$

Row 3 $\rightarrow 1 \cdot 1 + 1 \cdot 0 + (-1) = 0 \rightarrow 0 \quad \checkmark$

Row 4 $\rightarrow 1 \cdot 1 + 1 \cdot 1 + (-1) = 1 \rightarrow 1$

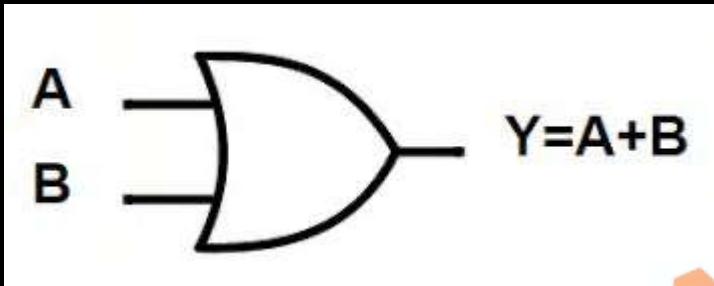
<http://www.knowledgegate.in/gate>



$$\begin{aligned} & w_1 \cdot x_1 + w_2 \cdot x_2 + b \\ & 1 \cdot x_1 + 1 \cdot x_2 - 1 \end{aligned}$$

OR gate using neural networks

- Steps in Perceptron Algorithm:
 - Initialize weight values and bias.
 - Forward propagate the inputs.
 - Check the error.
 - Backpropagate and adjust weights and bias.
 - Repeat for all training examples.



Truth Table		
Input		Output
A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

$$Y = \begin{cases} 1, & \text{if } wx + b > 0 \\ 0, & \text{if } wx + b \leq 0 \end{cases}$$

$$\omega_1 = 1$$

$$\omega_L = 1$$

$$b = -1$$

$$\omega_1 x_1 + \omega_L x_L + b$$

$$1 \cdot 0 + 1 \cdot 0 + (-1) = -1 \rightarrow 0$$

$$1 \cdot 0 + 1 \cdot 1 + (-1) = 0 \rightarrow 0$$

$$1 \cdot 1 + 1 \cdot 0 + (-1) = 0 \rightarrow 0$$

$$1 \cdot 1 + 1 \cdot 1 + (-1) = 1 \rightarrow 1$$

$$\omega_1 x_1 + \omega_L x_L + b$$

$$\omega_L = 1$$

$$2 \cdot 0 + 1 \cdot 0 + (-1) = -1 \rightarrow 0$$

$$2 \cdot 0 + 1 \cdot 1 + (-1) = 0 \rightarrow 0$$

$$2 \cdot 1 + 1 \cdot 0 + (-1) = 1 \rightarrow 1$$

$$2 \cdot 1 + 1 \cdot 1 + (-1) = 2 \rightarrow 1$$

$$\omega_1 = 2$$

$$\omega_L = 2$$

$$\omega_1 x_1 + \omega_L x_L + b$$

$$2 \cdot 0 + 2 \cdot 0 + (-1) = -1 \rightarrow 0$$

$$2 \cdot 0 + 2 \cdot 1 + (-1) = 1 \rightarrow 1$$

$$2 \cdot 1 + 2 \cdot 0 + (-1) = 1 \rightarrow 1$$

$$2 \cdot 1 + 2 \cdot 1 + (-1) = 3 \rightarrow 1$$



formula to change weights

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = h \cdot (t - o) \cdot x_i$$

h = learning rate ($0.1 \leftrightarrow 1$)

t = target o/p

o = actual o/p

x_i = i/o associated with w_i

$$\Delta b = h \cdot c = .1 \times 1 = 0.1$$

$$b = b + \Delta b = -1 + 0.1 = -0.9$$

$$w_1 \leftarrow w_1 + \Delta w_1$$

Row L ($x_1 = 0, x_L = 1$)

$$\Delta w_1 = h \cdot (t - o) \cdot x_1$$

$$\Delta w_1 = h \cdot e \cdot x_1$$

$$= 1 \times (1 - 0) \cdot 0$$

$$= 1 \times 1 \times 0$$

$$= 0$$

$$w_1 = 1 + 0 = 1$$

$$w_L \leftarrow w_L + \Delta w_L$$

Row 2 ($x_1 = 0, x_L = 1$)

$$\Delta w_L = h \cdot (t - o) \cdot x_2$$

$$\Delta w_L = h \cdot e \cdot x_L$$

$$= 1 \times 1 \times 1$$

$$= 1$$

$$w_L = 1 + 1 = 1.1$$

$$w_1 = 1$$

$$\rightarrow w_L = 1.1$$

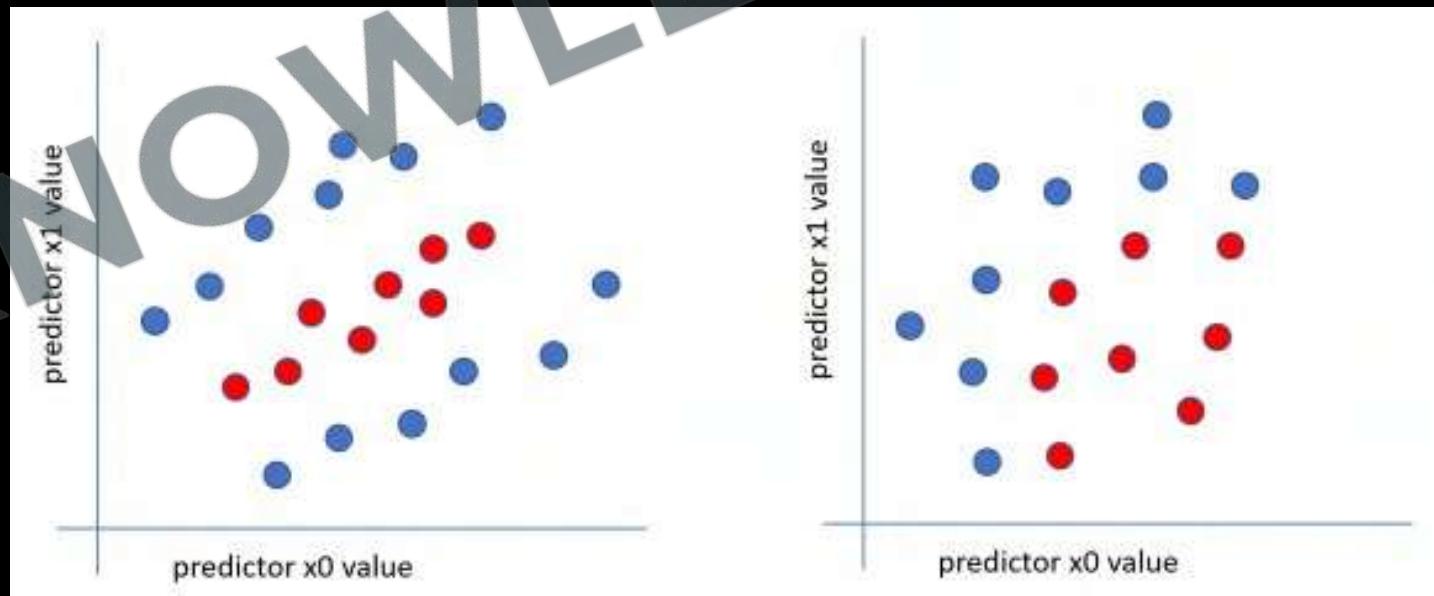
$$b = -0.9$$

Linearly Separable Data

- **Definition**: Data that can be perfectly separated by a straight line.
- **Algorithm**: Perceptron adjusts weights and bias to minimize classification error.
- **Outcome**: Converges to find an optimal decision boundary.

Non-Linearly Separable Data

- **Definition**: Data that cannot be perfectly separated by a single straight line.
- **Algorithm Limitation**: Perceptron may not converge, continuously adjusting weights and bias without finding a perfect solution.



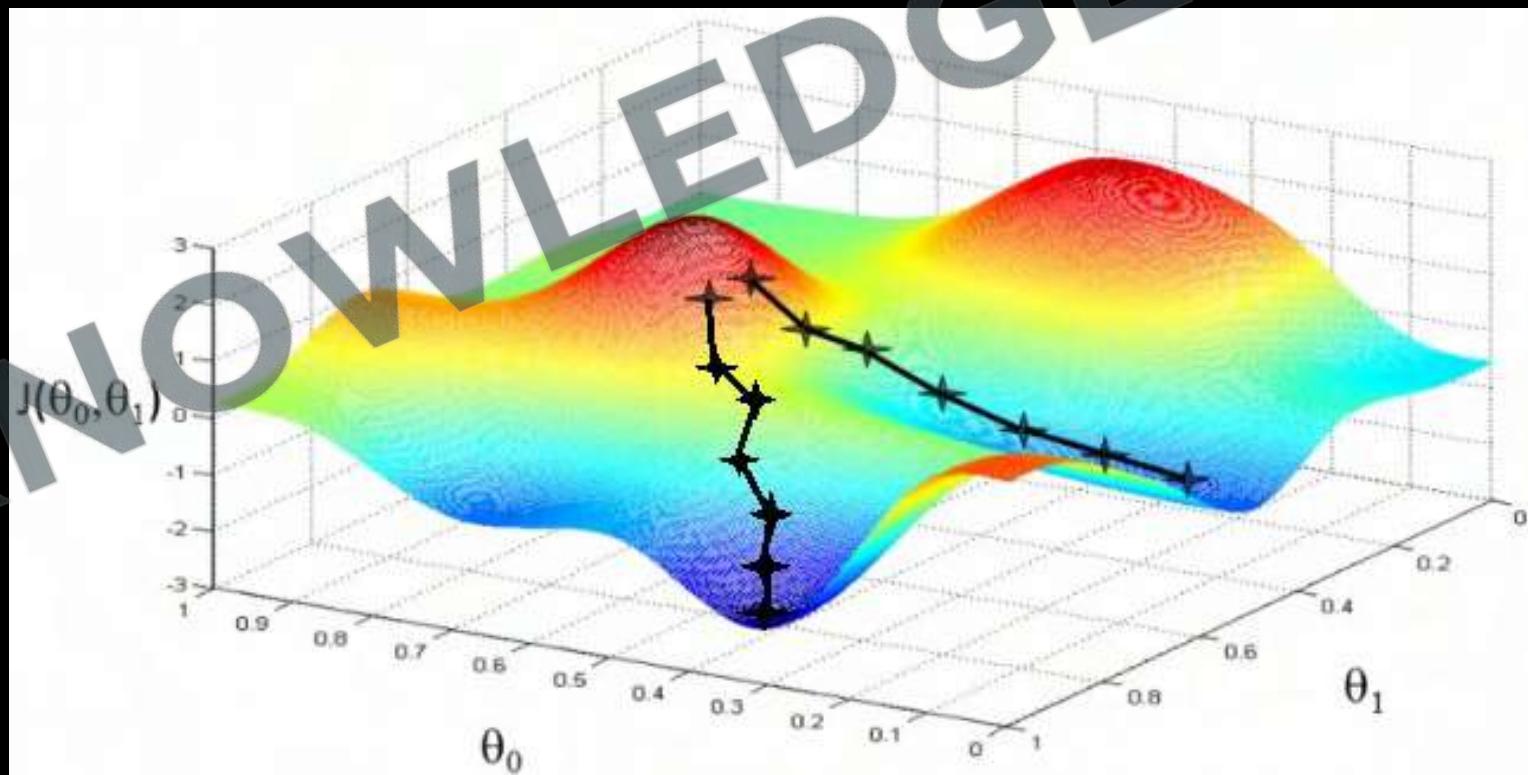
Key Point

- The Perceptron rule is effective for linearly separable data.
- For non-linearly separable data, more advanced methods like the delta rule (gradient descent) or multi-layer neural networks are required.

<http://www.knowledgegate.in/gate>

Gradient Descent Learning Algorithm

- It is the most used algorithm to train neural networks.
- An optimization algorithm to find model parameters (coefficients, weights) that minimize the error on the training dataset.
- It achieves this by making changes to the model that move it along a gradient/slope of errors towards a minimum error value.



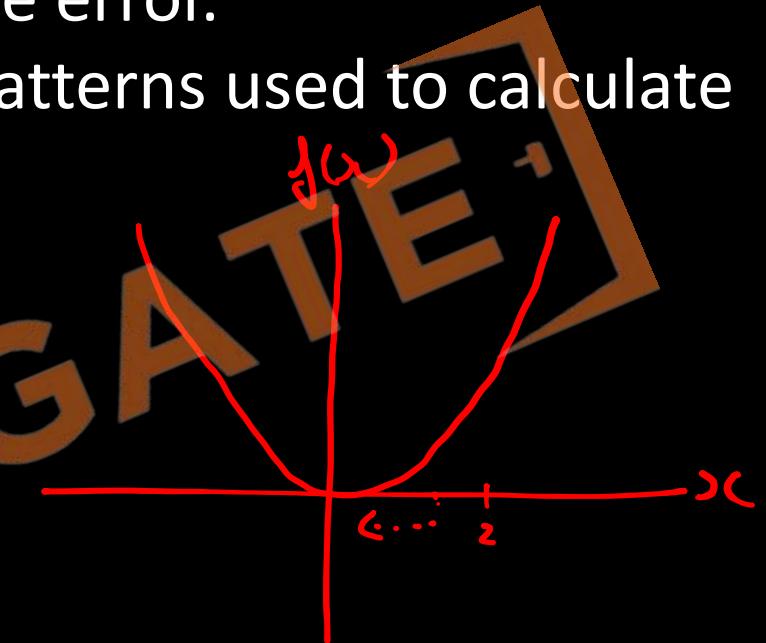
- Working:

- Adjusts model parameters iteratively to minimize the error.
- Gradient descent can vary in terms of the training patterns used to calculate errors and update the model.

$$f(x) = x^2$$

$$L.R = .1$$

$$\frac{d(x^2)}{dx} = 2x = \text{slope}$$



New value = old value - Slope * Learning Rate

$$= 2 - 4 * .1$$

$$= 2 - 0.4$$

$$w_i \leftarrow w_i + \Delta w_i \quad \text{gradient of } F \quad o_d = \sum w_i x_i$$

$$\Delta\omega_i = -\hbar \nabla E(\omega) \cdot \vec{r}$$

$\nabla E(w) \rightarrow$ derivative of error wrt weights

also called as gradient

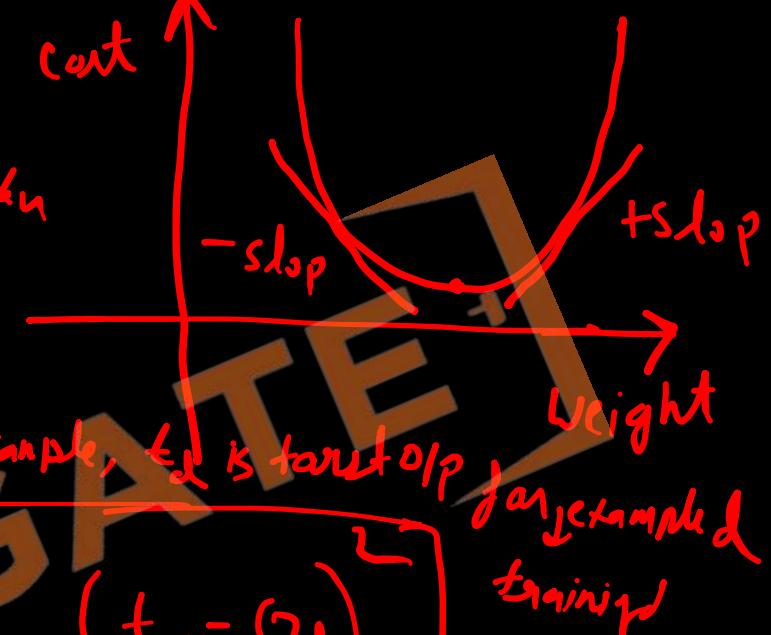
$$\nabla E(w) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

$$\frac{\partial F}{\partial w_i} = \frac{\partial}{\partial w_i} \left(\frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \right)$$

$$= \frac{1}{2} \sum \frac{\partial}{\partial w_i} \sum (L_d - o_d)^2$$

$\frac{1}{2} \times \sum_{d=1}^D (t_d - o_d)^2$ putting it in eqn (1)

$$= \frac{\omega_0 x_0 + \omega_1 x_1 + \omega_L x_L + \dots + \omega_n x_n}{\sum_{i=1}^n}$$



$$E = \frac{1}{2} \sum_{d \in D} (t_d - G_d)$$

$$= \sum (t_d - \theta_d) \frac{\partial}{\partial w_i} (t_d - \omega_d \cdot x_d)$$

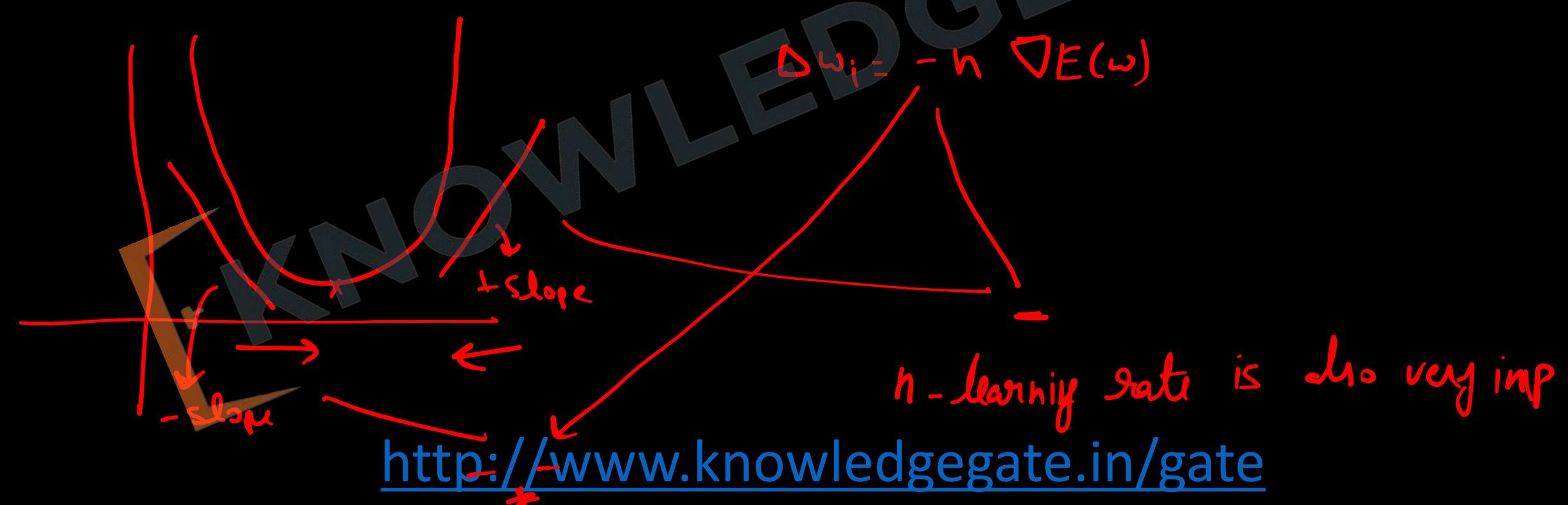
$$= \sum (t_d - O_d) (-x_d)$$

Putting it in our (1)

$$\Delta w_i = n \sum_{d \in D} (t_d - o_d) x_d$$

↓ ↓ ↓ ↓
 learning rate target o/p actual o/p i/o for respective weight

This rule also provide the basis for Back Propagation alg.



The main practical challenges in applying gradient descent are:

- Converging to a local minimum can be quite slow, often requiring many thousands of gradient descent steps.
- With multiple local minima in the error surface, there is no guarantee that the procedure will find the global minimum.

Types of Gradient Descent Algorithms

- **Batch Gradient Descent:**
 - Calculates the error for each example in the training dataset but updates the model after all training examples are evaluated.
 - Decreased update frequency results in more stable error gradients and computational efficiency but can be slow and expensive for large datasets.

- Stochastic Gradient Descent (SGD):

- Calculates the error for random examples in the training dataset instead of all examples.
- Faster than batch gradient descent as it updates the model parameters more frequently.

$$w_i = w_i + h(\epsilon - o)x_i \quad \text{based on current example}$$

- **Mini Batch Gradient Descent:**

- Splits the training dataset into smaller batches to calculate model error and update parameters.
- The update frequency is higher than batch gradient descent but less than stochastic gradient descent.
- Commonly used in deep learning due to a balance of efficiency and performance.

Delta Learning Rule

- **Definition:**
 - Developed by Widrow & Hoff. States that the modification in the weight of a node is equal to the multiplication of the error and the input.
- **Key Concepts:**
 - **Error:** The difference between the target and the actual output value.
 - Also Known As: LMS (Least Mean Square) Learning Rule.

Formula:

- $\Delta w = \eta \times (\text{error}) \times (\text{input})$
- $\Delta w = \eta \times (\text{target} - \text{actual}) \times \text{input}$
- New weight: $w_{\text{new}} = w_{\text{old}} + \Delta w$
- η : Learning rate.

- Characteristics:

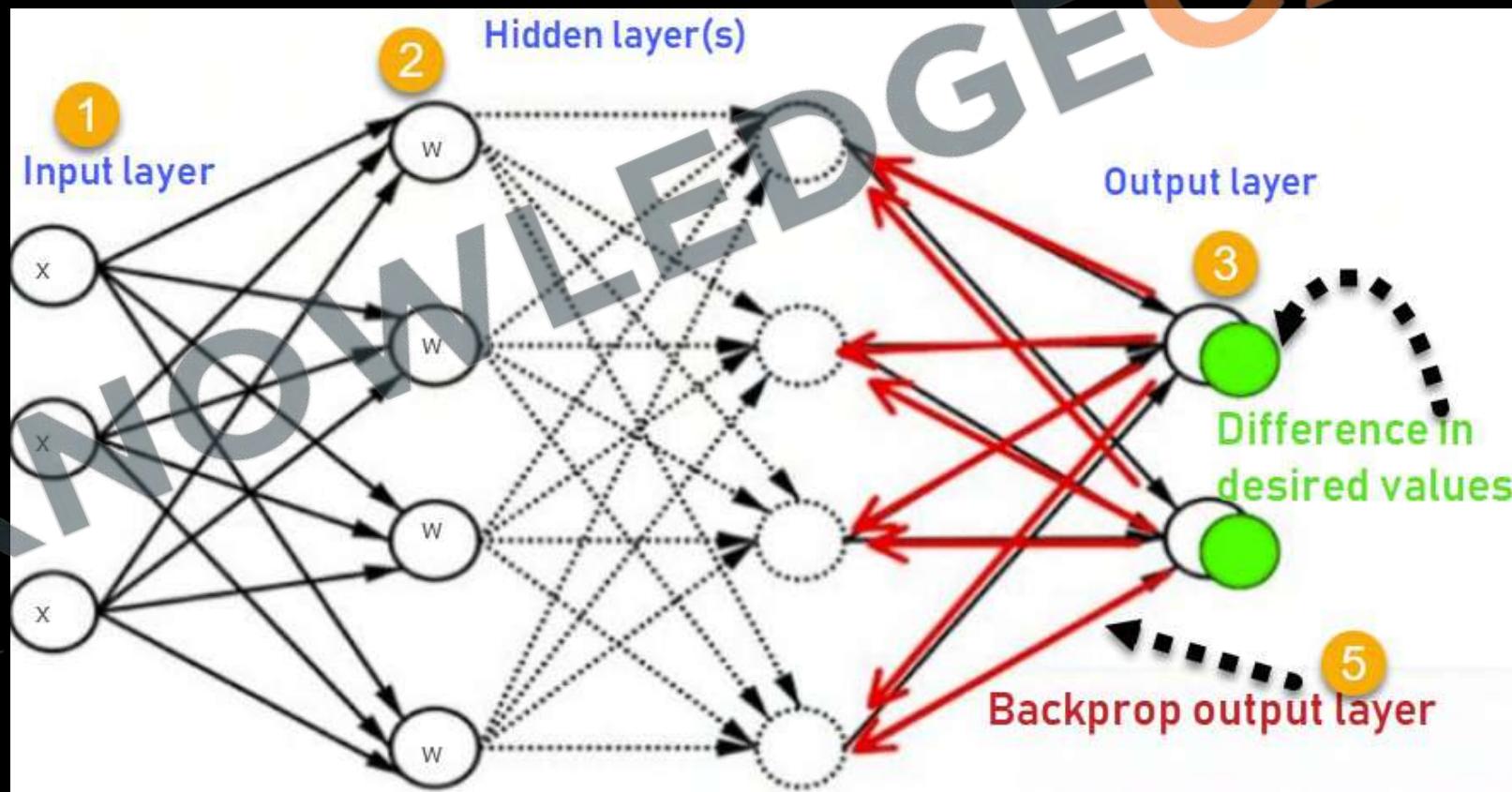
- Used in supervised training models.
- Independent of the activation function.
- Update rule for a single layer perceptron.

Steps in Delta Learning Rule

- Initialize weights with random values.
- Apply perceptron to each training sample i :
 - If sample i is misclassified, modify all weights:
 - $w_j = w_i + \eta(t_i - y_i)x_{ij}$
- Continue until all samples are correctly classified or the process converges.

Back Propagation

- Definition:
 - A standard method for training artificial neural networks.
 - Repeatedly adjusts the weights of the connections in the network to minimize the difference between actual output and desired output.



- **Steps in Back Propagation Algorithm:**
 - Inputs X arrive through the precomputed path.
 - Neurons are modelled using randomly assigned weights W .
 - Calculate the output for each neuron from the input layer, through the hidden layers, to the output layer.
 - Calculate the error in the outputs: Error = Actual Output - Desired
 - Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.
 - Repeat the process until the desired output is achieved.

Advantages of Back Propagation

- **Speed**: Fast, simple, and easy to program.
- **Flexibility**: Does not require prior knowledge about the network.
- **Utility**: Mainly useful for deep neural networks working on error-prone projects such as image or speech recognition.

Step 1: Forward Pass

Input to Hidden Layer

- Inputs: $x_1 = 0.3, x_2 = 0.8$
- Weights:
 - $W_{13} = 0.1$
 - $W_{14} = 0.5$
 - $W_{23} = 0.9$
 - $W_{24} = 0.5$

Calculate the input to hidden neurons H_3 and H_4 :

$$H_3 = \sigma(x_1 \cdot W_{13} + x_2 \cdot W_{23})$$

$$H_4 = \sigma(x_1 \cdot W_{14} + x_2 \cdot W_{24})$$

Where σ is the sigmoid function:

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

Calculations:

$$H_3 = \sigma(0.3 \cdot 0.1 + 0.8 \cdot 0.9)$$

$$H_3 = \sigma(0.03 + 0.72)$$

$$H_3 = \sigma(0.75)$$

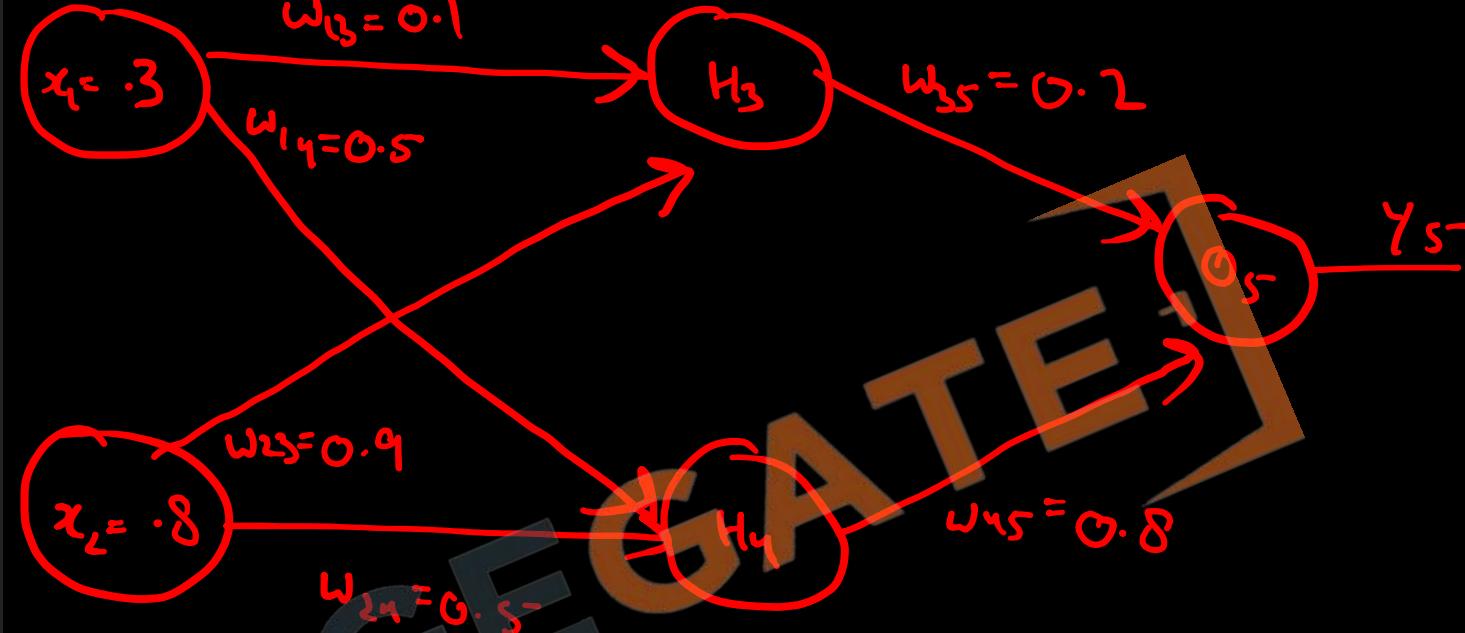
$$H_3 = \frac{1}{1+e^{-0.75}} \approx 0.679$$

$$H_4 = \sigma(0.3 \cdot 0.5 + 0.8 \cdot 0.5)$$

$$H_4 = \sigma(0.15 + 0.4)$$

$$H_4 = \sigma(0.55)$$

$$H_4 = \frac{1}{1+e^{-0.55}} \approx 0.634$$



Hidden to Output Layer

- Weights:

- $W_{35} = 0.2$
- $W_{45} = 0.8$

Calculate the input to output neuron O_5 :

$$O_5 = \sigma(H_3 \cdot W_{35} + H_4 \cdot W_{45})$$

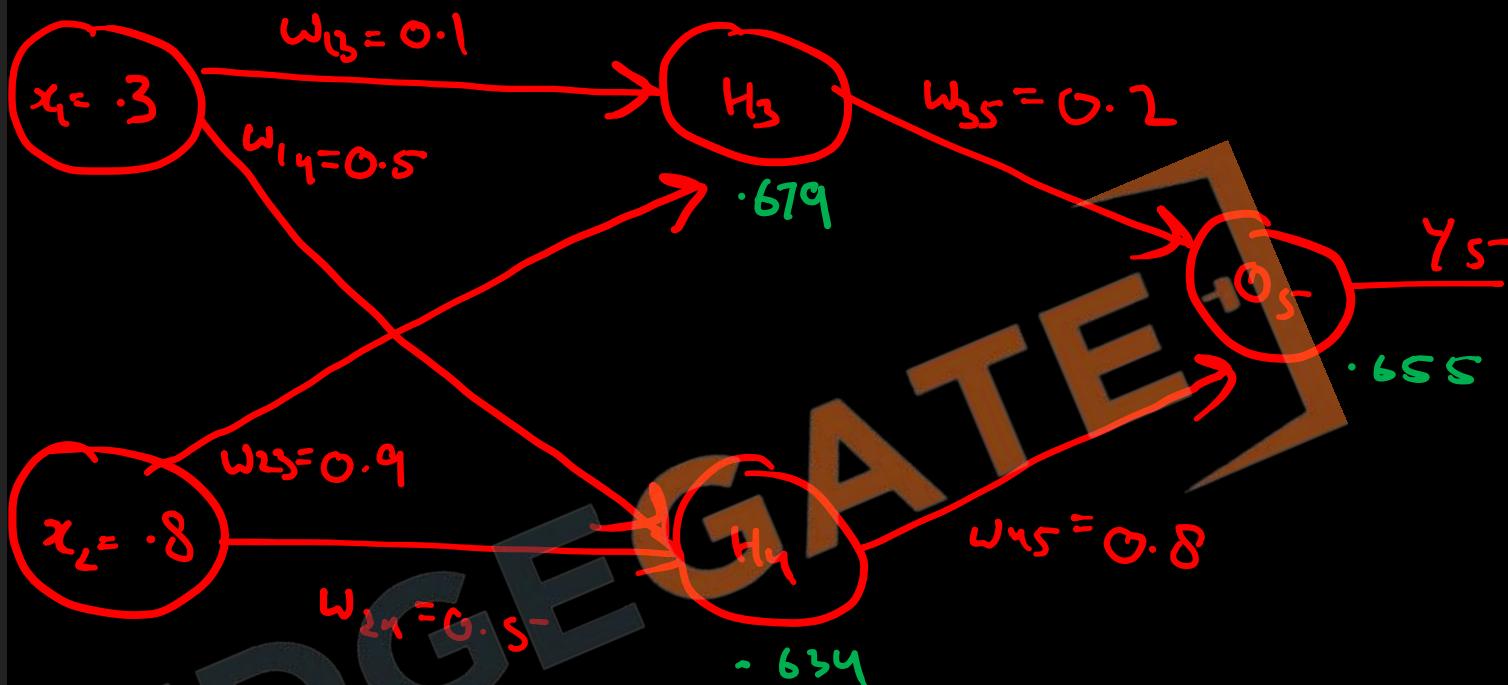
Calculations:

$$O_5 = \sigma(0.679 \cdot 0.2 + 0.634 \cdot 0.8)$$

$$O_5 = \sigma(0.1358 + 0.5072)$$

$$O_5 = \sigma(0.643)$$

$$O_5 = \frac{1}{1+e^{-0.643}} \approx 0.655$$



Step 2: Backward Pass

Given:

- Actual output $y = 0.5$
- Learning rate $\eta = 1$

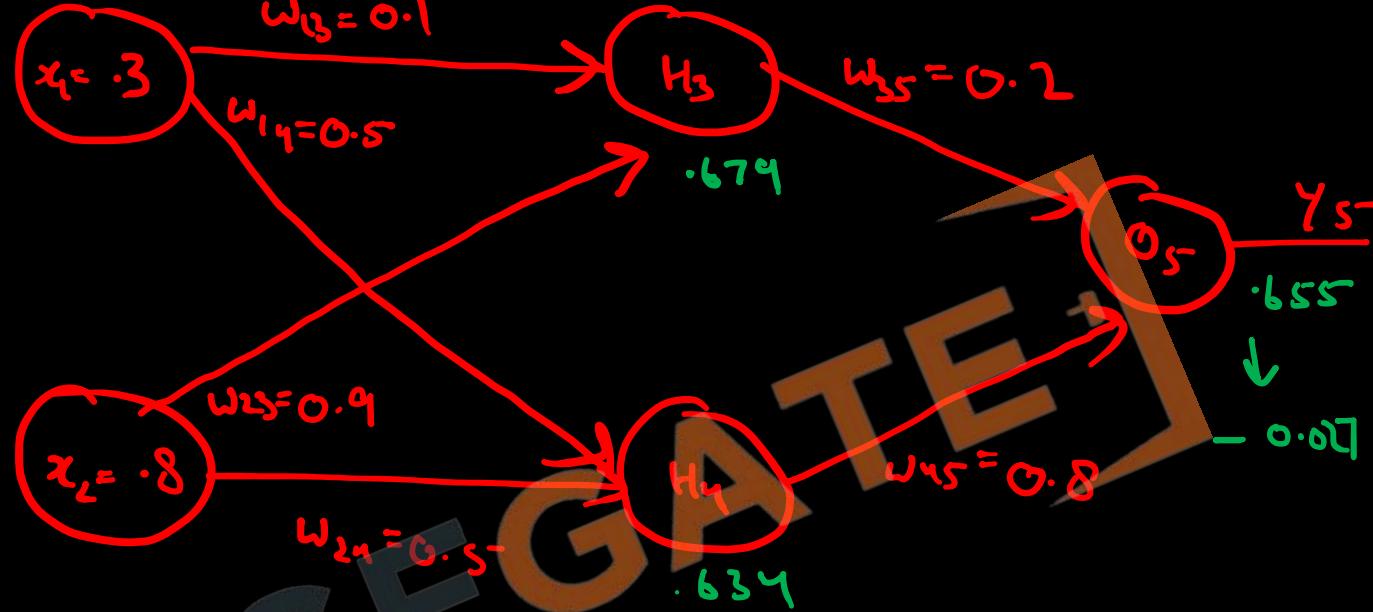
Calculate the error and update weights:

Error at Output Neuron

$$\delta_5 = (y - O_5) \cdot O_5 \cdot (1 - O_5)$$

$$\delta_5 = (0.5 - 0.655) \cdot 0.655 \cdot (1 - 0.655)$$

$$\delta_5 = -0.027$$



Update Weights from Hidden to Output Layer

$$\Delta W_{35} = \eta \cdot \delta_5 \cdot H_3$$

$$\Delta W_{35} = 1 \cdot -0.027 \cdot 0.679$$

$$\Delta W_{35} = -0.018$$

$$W_{35} = 0.2 - 0.018 = 0.182$$

$$\Delta W_{45} = \eta \cdot \delta_5 \cdot H_4$$

$$\Delta W_{45} = 1 \cdot -0.027 \cdot 0.634$$

$$\Delta W_{45} = -0.017$$

$$W_{45} = 0.8 - 0.017 = 0.783$$

Error at Hidden Neurons

$$\delta_3 = \delta_5 \cdot W_{35} \cdot H_3 \cdot (1 - H_3)$$

$$\delta_4 = \delta_5 \cdot W_{45} \cdot H_4 \cdot (1 - H_4)$$

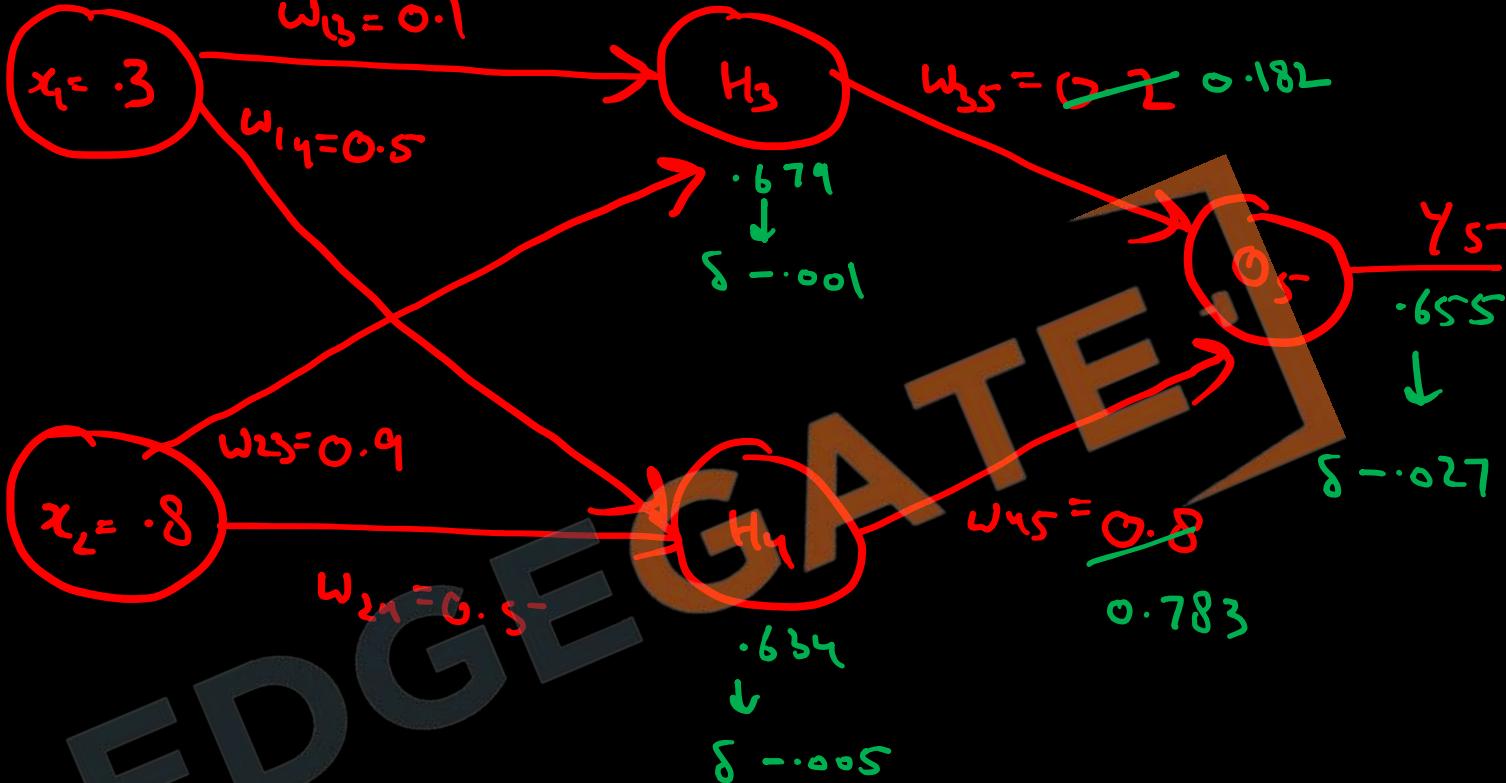
Calculations:

$$\delta_3 = -0.027 \cdot 0.2 \cdot 0.679 \cdot (1 - 0.679)$$

$$\delta_3 = -0.001$$

$$\delta_4 = -0.027 \cdot 0.8 \cdot 0.634 \cdot (1 - 0.634)$$

$$\delta_4 = -0.005$$



Update Weights from Input to Hidden Layer

$$\Delta W_{13} = \eta \cdot \delta_3 \cdot x_1$$

$$\Delta W_{13} = 1 \cdot -0.001 \cdot 0.3$$

$$\Delta W_{13} = -0.0003$$

$$W_{13} = 0.1 - 0.0003 = 0.0997$$

$$\Delta W_{23} = \eta \cdot \delta_3 \cdot x_2$$

$$\Delta W_{23} = 1 \cdot -0.001 \cdot 0.8$$

$$\Delta W_{23} = -0.0008$$

$$W_{23} = 0.9 - 0.0008 = 0.8992$$

$$\Delta W_{14} = \eta \cdot \delta_4 \cdot x_1$$

$$\Delta W_{14} = 1 \cdot -0.005 \cdot 0.3$$

$$\Delta W_{14} = -0.0015$$

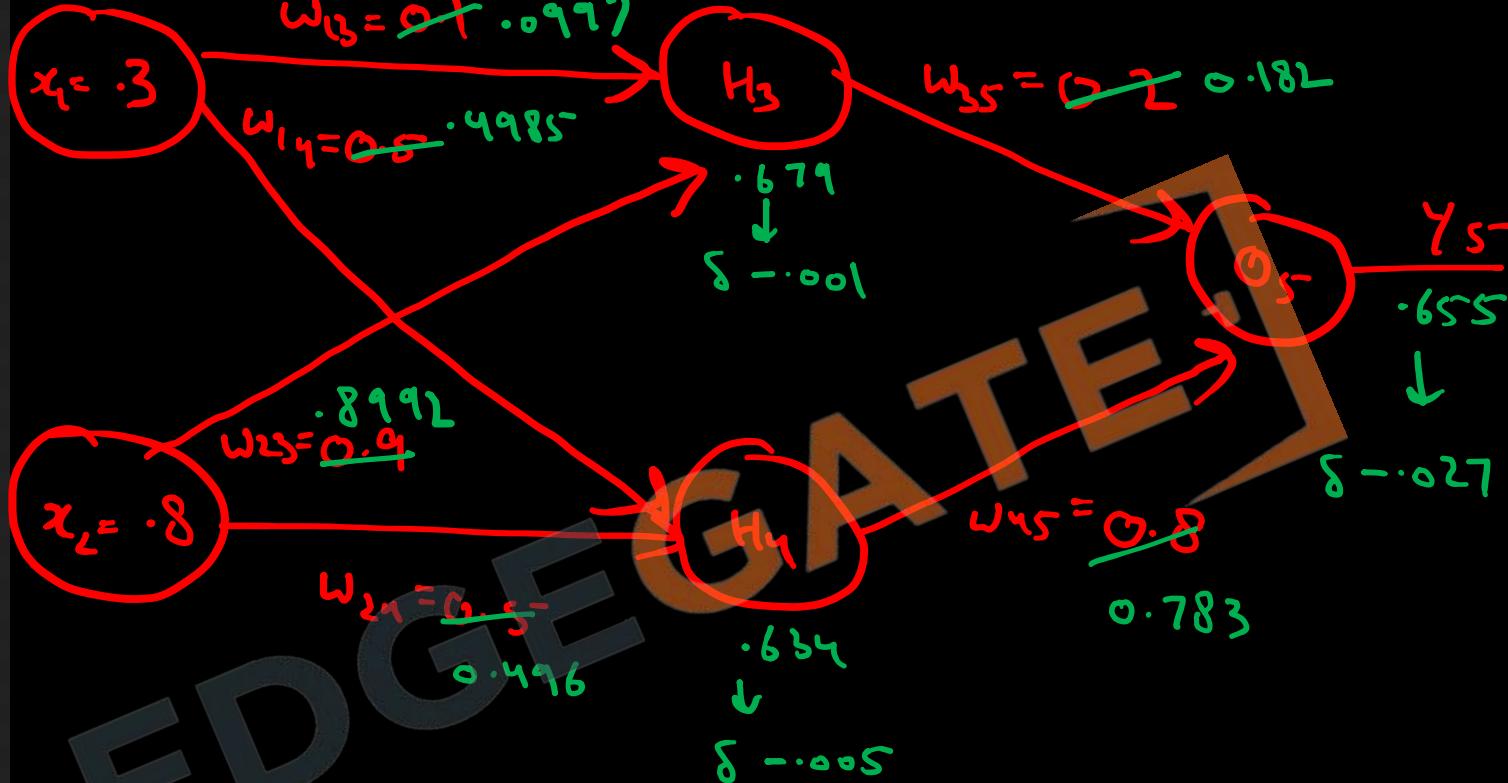
$$W_{14} = 0.5 - 0.0015 = 0.4985$$

$$\Delta W_{24} = \eta \cdot \delta_4 \cdot x_2$$

$$\Delta W_{24} = 1 \cdot -0.005 \cdot 0.8$$

$$\Delta W_{24} = -0.004$$

$$W_{24} = 0.5 - 0.004 = 0.496$$



Step 3: Perform Another Forward Pass

Using the updated weights:

New Weights:

- $W_{13} = 0.0997$
- $W_{14} = 0.4985$
- $W_{23} = 0.8992$
- $W_{24} = 0.496$
- $W_{35} = 0.182$
- $W_{45} = 0.783$

Forward Pass with Updated Weights

Calculate new hidden layer activations:

$$H_3 = \sigma(0.3 \cdot 0.0997 + 0.8 \cdot 0.8992)$$

$$H_3 = \sigma(0.02991 + 0.71936)$$

$$H_3 = \sigma(0.74927)$$

$$H_3 \approx 0.678$$

$$H_4 = \sigma(0.3 \cdot 0.4985 + 0.8 \cdot 0.496)$$

$$H_4 = \sigma(0.14955 + 0.3968)$$

$$H_4 = \sigma(0.54635)$$

$$H_4 \approx 0.633$$

Calculate new output layer activation:

$$O_5 = \sigma(H_3 \cdot W_{35} + H_4 \cdot W_{45})$$

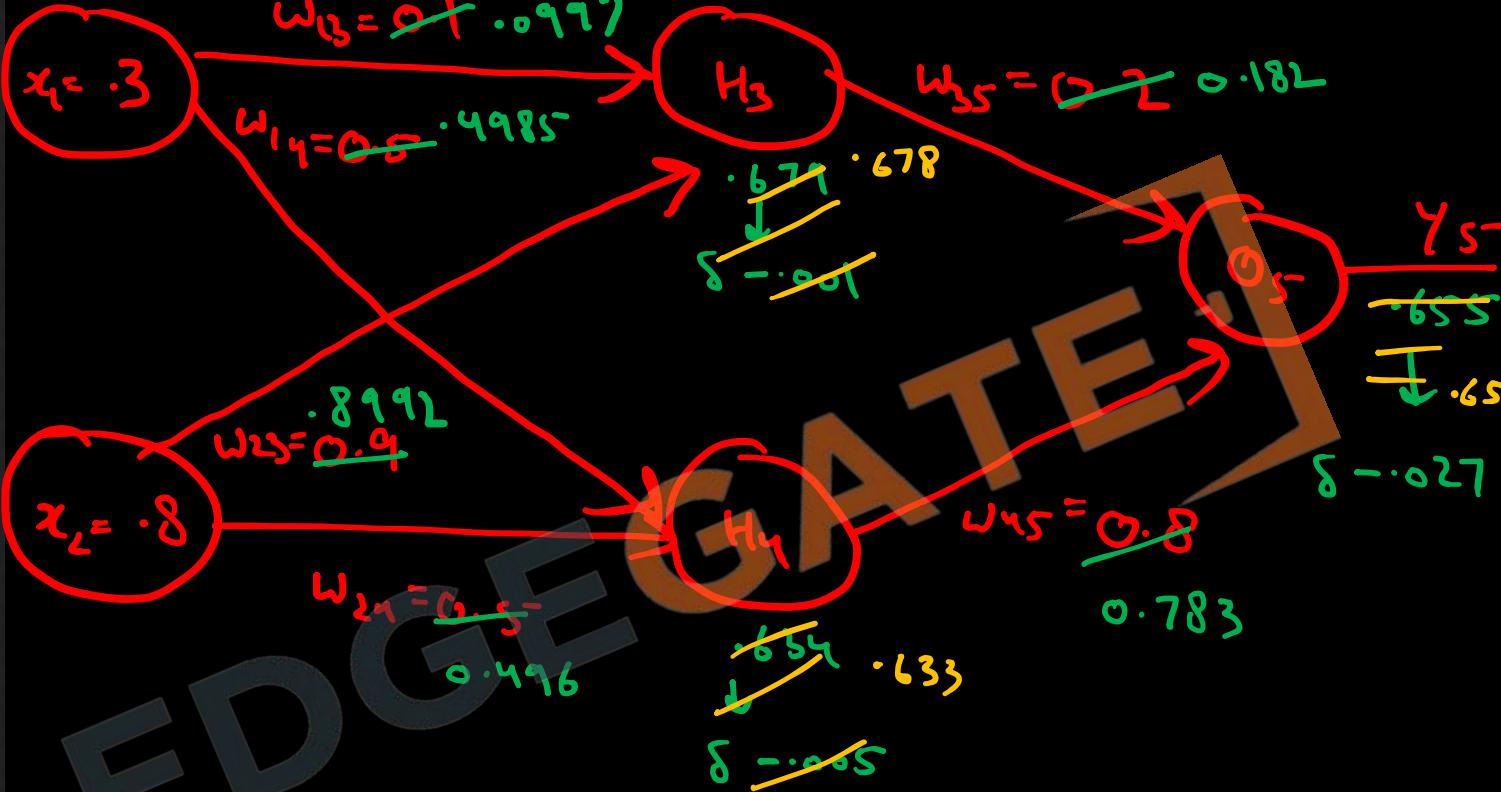
$$O_5 = \sigma(0.678 \cdot 0.182 + 0.633 \cdot 0.783)$$

$$O_5 = \sigma(0.123396 + 0.495189)$$

$$O_5 = \sigma(0.618585)$$

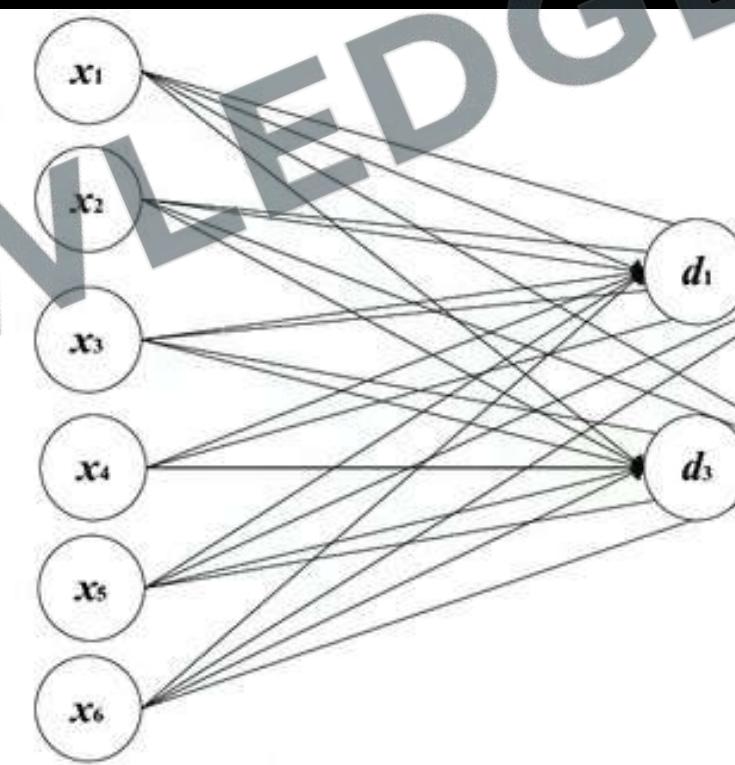
$$O_5 \approx 0.650$$

Thus, the output y after performing another forward pass with the updated weights is approximately 0.650.



Self-Organizing Maps (SOM) / Kohonene Self Organizing Maps

- Developed by T. Kohonen in 1982. Named "Self-organizing" because no supervision is required. Learn through unsupervised competitive learning.
- A Kind of neural network which uses Competitive learning algorithm to maps multidimensional data into lower dimensions for easier interpretation specially for complex problems. It will contain only two layers
 - Input layer (n units)
 - Output layer (m units)



Example of Dimensionality Reduction Using SOMs

Scenario: Visualizing customer data with features like age, income, spending score, and product preferences.

Goal: Reduce dimensionality to visualize patterns in customer behavior.

1. High-Dimensional Data:

Each customer is represented as a vector:

$$\mathbf{c}_1 = [35, 50000, 85, 1, 0, 0, 1, \dots]$$

(age, income, spending score, product preferences)

2. Using SOM for Dimensionality Reduction:

- Train a SOM with the customer data.
- Project the high-dimensional data onto a 2D grid, preserving relationships.

3. 2D Grid Visualization:

The SOM organizes customers into clusters:



egate.in/gate

4. Interpreting Clusters:

- **Cluster 1:** Young, high income, high spending.
- **Cluster 2:** Middle-aged, moderate income and spending.
- **Cluster 3:** Older, low income, high spending on specific products.

5. Applications:

- **Marketing:** Target personalized campaigns.
- **Product Development:** Develop products for different customer segments.
- **Customer Support:** Tailor strategies for different clusters.

SOM Training Algorithm:

1. **Initialization:** Choose random values for the initial weights w_{ij} .
2. **Sampling:** Take a sample training input vector $x(t)$ from the input layer.
3. **Matching:** Find the winning neuron from the output layer that has the weight vector closest to the input vector. It can be calculated by taking the square of the Euclidean distance for each output unit and finding the output unit that has the minimum Euclidean distance from the input vector.

$$D_j = \sum_{i=1}^n (x_i - w_{ij})^2 \quad \text{for each } j = 1 \text{ to } m$$

4. **New Weight Calculation:** Find the new weights between the input vector sample and the winning output unit (neuron).

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \eta(t) \cdot (x_i - w_{ij}(\text{old}))$$

where $\eta(t)$ is the learning rate.

5. **Continuation:** Repeat steps 2 to 4 until weight updation is negligible (i.e., new weights are similar to old weights or feature map stops changing).

Training Samples:

$x_1 (0, 1, 0, 1)$

$x_2 (0, 1, 1, 1)$

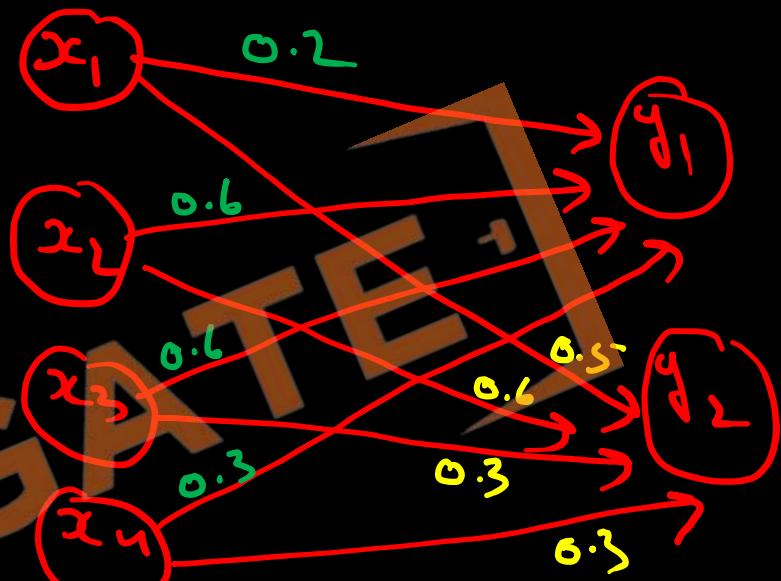
$x_3 (0, 0, 0, 0)$

$x_4 (1, 0, 0, 1)$

Learning rate = 0.5

initial weight matrix

$$\begin{bmatrix} \text{Unit 1} \\ \text{Unit 2} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.6 & 0.6 & 0.3 \\ 0.5 & 0.6 & 0.3 & 0.3 \end{bmatrix}$$



To compute the Euclidean distance between the input vector $\mathbf{x}_1 = [0, 1, 0, 1]$ and the weight vector of Unit 1, we use the formula for Euclidean distance:

$$D = \sqrt{\sum_{i=1}^n (x_i - w_i)^2}$$

Given:

- $\mathbf{x}_1 = [0, 1, 0, 1]$
- Weight vector for Unit 1: $\mathbf{w}_{\text{unit1}} = [0.2, 0.6, 0.6, 0.3]$

Let's compute the Euclidean distance step-by-step:

1. Compute the difference for each component:

- $(0 - 0.2)^2 = 0.04$
- $(1 - 0.6)^2 = 0.16$
- $(0 - 0.6)^2 = 0.36$
- $(1 - 0.3)^2 = 0.49$

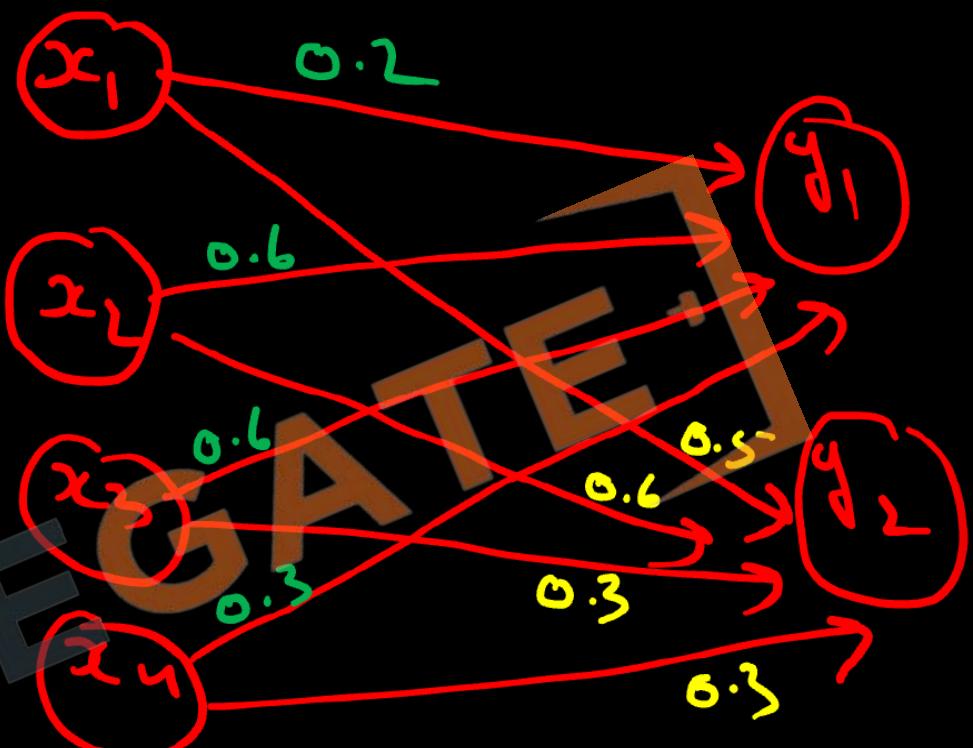
2. Sum the squared differences:

$$0.04 + 0.16 + 0.36 + 0.49 = 1.05$$

3. Take the square root of the sum:

$$D = \sqrt{1.05} \approx 1.0247$$

So, the Euclidean distance between \mathbf{x}_1 and the weight vector of Unit 1 is approximately 1.0247.



To compute the Euclidean distance between the input vector $\mathbf{x}_1 = [0, 1, 0, 1]$ and the weight vector of Unit 2, we will use the same formula:

$$D = \sqrt{\sum_{i=1}^n (x_i - w_i)^2}$$

Given:

- $\mathbf{x}_1 = [0, 1, 0, 1]$
- Weight vector for Unit 2: $\mathbf{w}_{\text{unit2}} = [0.5, 0.6, 0.3, 0.3]$

Let's compute the Euclidean distance step-by-step:

1. Compute the difference for each component:

- $(0 - 0.5)^2 = 0.25$
- $(1 - 0.6)^2 = 0.16$
- $(0 - 0.3)^2 = 0.09$
- $(1 - 0.3)^2 = 0.49$

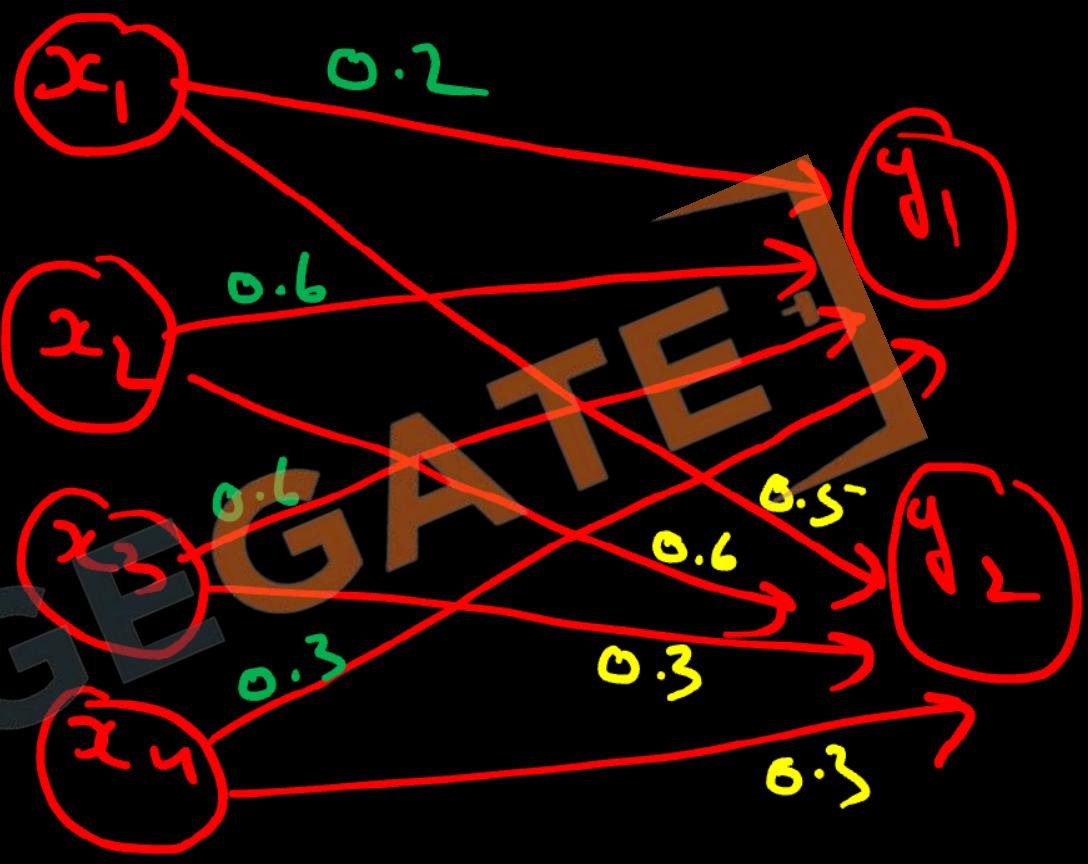
2. Sum the squared differences:

$$0.25 + 0.16 + 0.09 + 0.49 = 0.99$$

3. Take the square root of the sum:

$$D = \sqrt{0.99} \approx 0.995$$

So, the Euclidean distance between \mathbf{x}_1 and the weight vector of Unit 2 is approximately 0.995.



To update the weights of Unit 2 after determining that it is the winning unit, we use the formula:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \eta \cdot (x_i - w_{ij}(\text{old}))$$

Given:

- $x_1 = [0, 1, 0, 1]$
- Initial weights for Unit 2: $w_{\text{unit2}} = [0.5, 0.6, 0.3, 0.3]$
- Learning rate (η) = 0.5

Let's update each weight step-by-step:

1. For the first weight w_{21} :

$$w_{21}(\text{new}) = 0.5 + 0.5 \cdot (0 - 0.5) = 0.5 + 0.5 \cdot (-0.5) = 0.5 - 0.25 = 0.25$$

2. For the second weight w_{22} :

$$w_{22}(\text{new}) = 0.6 + 0.5 \cdot (1 - 0.6) = 0.6 + 0.5 \cdot 0.4 = 0.6 + 0.2 = 0.8$$

3. For the third weight w_{23} :

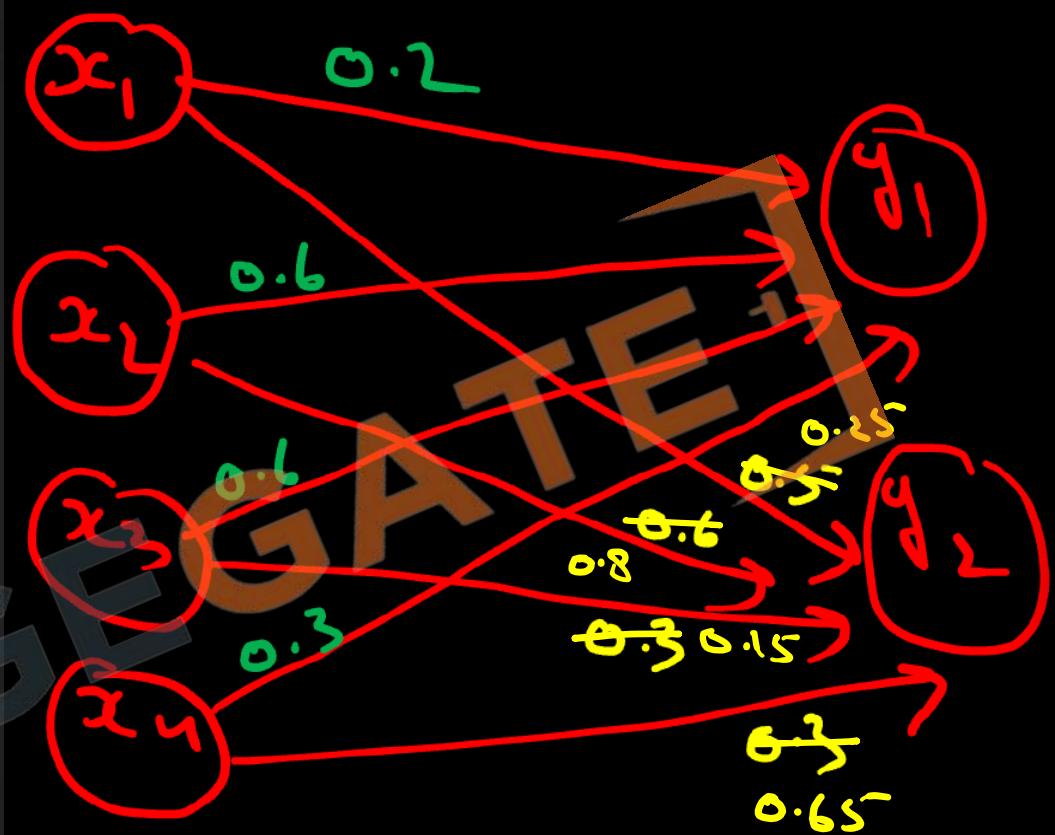
$$w_{23}(\text{new}) = 0.3 + 0.5 \cdot (0 - 0.3) = 0.3 + 0.5 \cdot (-0.3) = 0.3 - 0.15 = 0.15$$

4. For the fourth weight w_{24} :

$$w_{24}(\text{new}) = 0.3 + 0.5 \cdot (1 - 0.3) = 0.3 + 0.5 \cdot 0.7 = 0.3 + 0.35 = 0.65$$

So, the updated weights for Unit 2 are:

$$w_{\text{unit2}}(\text{new}) = [0.25, 0.8, 0.15, 0.65]$$



To continue training with $\mathbf{x}_2 = [0, 1, 1, 1]$, we first need to compute the Euclidean distances between \mathbf{x}_2 and the updated weight vectors for both Unit 1 and Unit 2.

Euclidean Distance Calculation

1. Distance between \mathbf{x}_2 and Unit 1:

- Weight vector for Unit 1: $\mathbf{w}_{\text{unit1}} = [0.2, 0.6, 0.6, 0.3]$
- $\mathbf{x}_2 = [0, 1, 1, 1]$

$$D_1 = \sqrt{(0 - 0.2)^2 + (1 - 0.6)^2 + (1 - 0.6)^2 + (1 - 0.3)^2}$$

$$D_1 = \sqrt{0.04 + 0.16 + 0.16 + 0.49} = \sqrt{0.85} \approx 0.922$$

2. Distance between \mathbf{x}_2 and Unit 2:

- Updated weight vector for Unit 2: $\mathbf{w}_{\text{unit2}} = [0.25, 0.8, 0.15, 0.65]$
- $\mathbf{x}_2 = [0, 1, 1, 1]$

$$D_2 = \sqrt{(0 - 0.25)^2 + (1 - 0.8)^2 + (1 - 0.15)^2 + (1 - 0.65)^2}$$

$$D_2 = \sqrt{0.0625 + 0.04 + 0.7225 + 0.1225} = \sqrt{0.9475} \approx 0.973$$

Determine the Winning Unit

- The Euclidean distance for Unit 1 is approximately 0.922.
- The Euclidean distance for Unit 2 is approximately 0.973.

Since Unit 1 has the smaller distance, Unit 1 is the winning unit this time.



Update Weights for Unit 1

Using the same update formula:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \eta \cdot (x_i - w_{ij}(\text{old}))$$

Given:

- $x_2 = [0, 1, 1, 1]$
- Initial weights for Unit 1: $w_{\text{unit1}} = [0.2, 0.6, 0.6, 0.3]$
- Learning rate (η) = 0.5

1. For the first weight w_{11} :

$$w_{11}(\text{new}) = 0.2 + 0.5 \cdot (0 - 0.2) = 0.2 + 0.5 \cdot (-0.2) = 0.2 - 0.1 = 0.1$$

2. For the second weight w_{12} :

$$w_{12}(\text{new}) = 0.6 + 0.5 \cdot (1 - 0.6) = 0.6 + 0.5 \cdot 0.4 = 0.6 + 0.2 = 0.8$$

3. For the third weight w_{13} :

$$w_{13}(\text{new}) = 0.6 + 0.5 \cdot (1 - 0.6) = 0.6 + 0.5 \cdot 0.4 = 0.6 + 0.2 = 0.8$$

4. For the fourth weight w_{14} :

$$w_{14}(\text{new}) = 0.3 + 0.5 \cdot (1 - 0.3) = 0.3 + 0.5 \cdot 0.7 = 0.3 + 0.35 = 0.65$$

So, the updated weights for Unit 1 are:

$$w_{\text{unit1}}(\text{new}) = [0.1, 0.8, 0.8, 0.65]$$



To continue training with $\mathbf{x}_3 = [0, 0, 0, 0]$, we first need to compute the Euclidean distances between \mathbf{x}_3 and the updated weight vectors for both Unit 1 and Unit 2.

Euclidean Distance Calculation

1. Distance between \mathbf{x}_3 and Unit 1:

- Updated weight vector for Unit 1: $\mathbf{w}_{\text{unit1}} = [0.1, 0.8, 0.8, 0.65]$
- $\mathbf{x}_3 = [0, 0, 0, 0]$

$$D_1 = \sqrt{(0 - 0.1)^2 + (0 - 0.8)^2 + (0 - 0.8)^2 + (0 - 0.65)^2}$$

$$D_1 = \sqrt{0.01 + 0.64 + 0.64 + 0.4225} = \sqrt{1.7125} \approx 1.309$$

2. Distance between \mathbf{x}_3 and Unit 2:

- Updated weight vector for Unit 2: $\mathbf{w}_{\text{unit2}} = [0.25, 0.8, 0.15, 0.65]$
- $\mathbf{x}_3 = [0, 0, 0, 0]$

$$D_2 = \sqrt{(0 - 0.25)^2 + (0 - 0.8)^2 + (0 - 0.15)^2 + (0 - 0.65)^2}$$

$$D_2 = \sqrt{0.0625 + 0.64 + 0.0225 + 0.4225} = \sqrt{1.1475} \approx 1.071$$

Determine the Winning Unit

- The Euclidean distance for Unit 1 is approximately 1.309.
- The Euclidean distance for Unit 2 is approximately 1.071.

Since Unit 2 has the smaller distance, Unit 2 is the winning unit this time.



Update Weights for Unit 2

Using the same update formula:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \eta \cdot (x_i - w_{ij}(\text{old}))$$

Given:

- $x_3 = [0, 0, 0, 0]$
- Updated weights for Unit 2: $w_{\text{unit2}} = [0.25, 0.8, 0.15, 0.65]$
- Learning rate (η) = 0.5

1. For the first weight w_{21} :

$$w_{21}(\text{new}) = 0.25 + 0.5 \cdot (0 - 0.25) = 0.25 + 0.5 \cdot (-0.25) = 0.25 - 0.125 = 0.125$$

2. For the second weight w_{22} :

$$w_{22}(\text{new}) = 0.8 + 0.5 \cdot (0 - 0.8) = 0.8 + 0.5 \cdot (-0.8) = 0.8 - 0.4 = 0.4$$

3. For the third weight w_{23} :

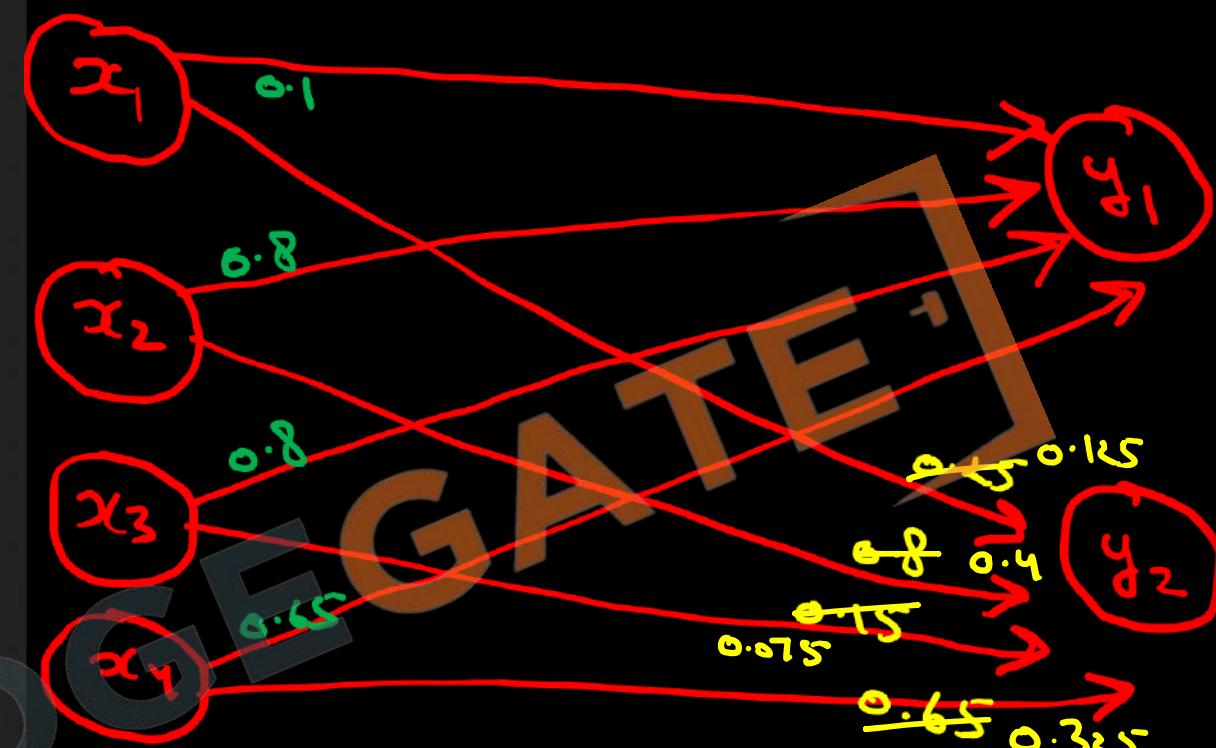
$$w_{23}(\text{new}) = 0.15 + 0.5 \cdot (0 - 0.15) = 0.15 + 0.5 \cdot (-0.15) = 0.15 - 0.075 = 0.075$$

4. For the fourth weight w_{24} :

$$w_{24}(\text{new}) = 0.65 + 0.5 \cdot (0 - 0.65) = 0.65 + 0.5 \cdot (-0.65) = 0.65 - 0.325 = 0.325$$

So, the updated weights for Unit 2 are:

$$w_{\text{unit2}}(\text{new}) = [0.125, 0.4, 0.075, 0.325]$$



To continue training with $\mathbf{x}_4 = [1, 0, 0, 1]$, we first need to compute the Euclidean distances between \mathbf{x}_4 and the updated weight vectors for both Unit 1 and Unit 2.

Euclidean Distance Calculation

1. Distance between \mathbf{x}_4 and Unit 1:

- Updated weight vector for Unit 1: $\mathbf{w}_{\text{unit1}} = [0.1, 0.8, 0.8, 0.65]$
- $\mathbf{x}_4 = [1, 0, 0, 1]$

$$D_1 = \sqrt{(1 - 0.1)^2 + (0 - 0.8)^2 + (0 - 0.8)^2 + (1 - 0.65)^2}$$

$$D_1 = \sqrt{(0.9)^2 + (-0.8)^2 + (-0.8)^2 + (0.35)^2}$$

$$D_1 = \sqrt{0.81 + 0.64 + 0.64 + 0.1225} = \sqrt{2.2125} \approx 1.487$$

2. Distance between \mathbf{x}_4 and Unit 2:

- Updated weight vector for Unit 2: $\mathbf{w}_{\text{unit2}} = [0.125, 0.4, 0.075, 0.325]$
- $\mathbf{x}_4 = [1, 0, 0, 1]$

$$D_2 = \sqrt{(1 - 0.125)^2 + (0 - 0.4)^2 + (0 - 0.075)^2 + (1 - 0.325)^2}$$

$$D_2 = \sqrt{(0.875)^2 + (-0.4)^2 + (-0.075)^2 + (0.675)^2}$$

$$D_2 = \sqrt{0.765625 + 0.16 + 0.005625 + 0.455625} = \sqrt{1.386875} \approx 1.178$$

Determine the Winning Unit

- The Euclidean distance for Unit 1 is approximately 1.487.
- The Euclidean distance for Unit 2 is approximately 1.178.

Since Unit 2 has the smaller distance, Unit 2 is the winning unit again.



Update Weights for Unit 2

Using the same update formula:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \eta \cdot (x_i - w_{ij}(\text{old}))$$

Given:

- $x_4 = [1, 0, 0, 1]$
- Updated weights for Unit 2: $w_{\text{unit2}} = [0.125, 0.4, 0.075, 0.325]$
- Learning rate (η) = 0.5

1. For the first weight w_{21} :

$$w_{21}(\text{new}) = 0.125 + 0.5 \cdot (1 - 0.125) = 0.125 + 0.5 \cdot 0.875 = 0.125 + 0.4375 = 0.5625$$

2. For the second weight w_{22} :

$$w_{22}(\text{new}) = 0.4 + 0.5 \cdot (0 - 0.4) = 0.4 + 0.5 \cdot (-0.4) = 0.4 - 0.2 = 0.2$$

3. For the third weight w_{23} :

$$w_{23}(\text{new}) = 0.075 + 0.5 \cdot (0 - 0.075) = 0.075 + 0.5 \cdot (-0.075) = 0.075 - 0.0375 = 0.0375$$

4. For the fourth weight w_{24} :

$$w_{24}(\text{new}) = 0.325 + 0.5 \cdot (1 - 0.325) = 0.325 + 0.5 \cdot 0.675 = 0.325 + 0.3375 = 0.6625$$

So, the updated weights for Unit 2 are:

$$w_{\text{unit2}}(\text{new}) = [0.5625, 0.2, 0.0375, 0.6625]$$



After the training process with the given samples x_1, x_2, x_3, x_4 , the final weights for Unit 1 and Unit 2 are:

Final Weights

Unit 1

Updated after training with x_2 :

$$w_{\text{unit1}}(\text{new}) = [0.1, 0.8, 0.8, 0.65]$$

Unit 2

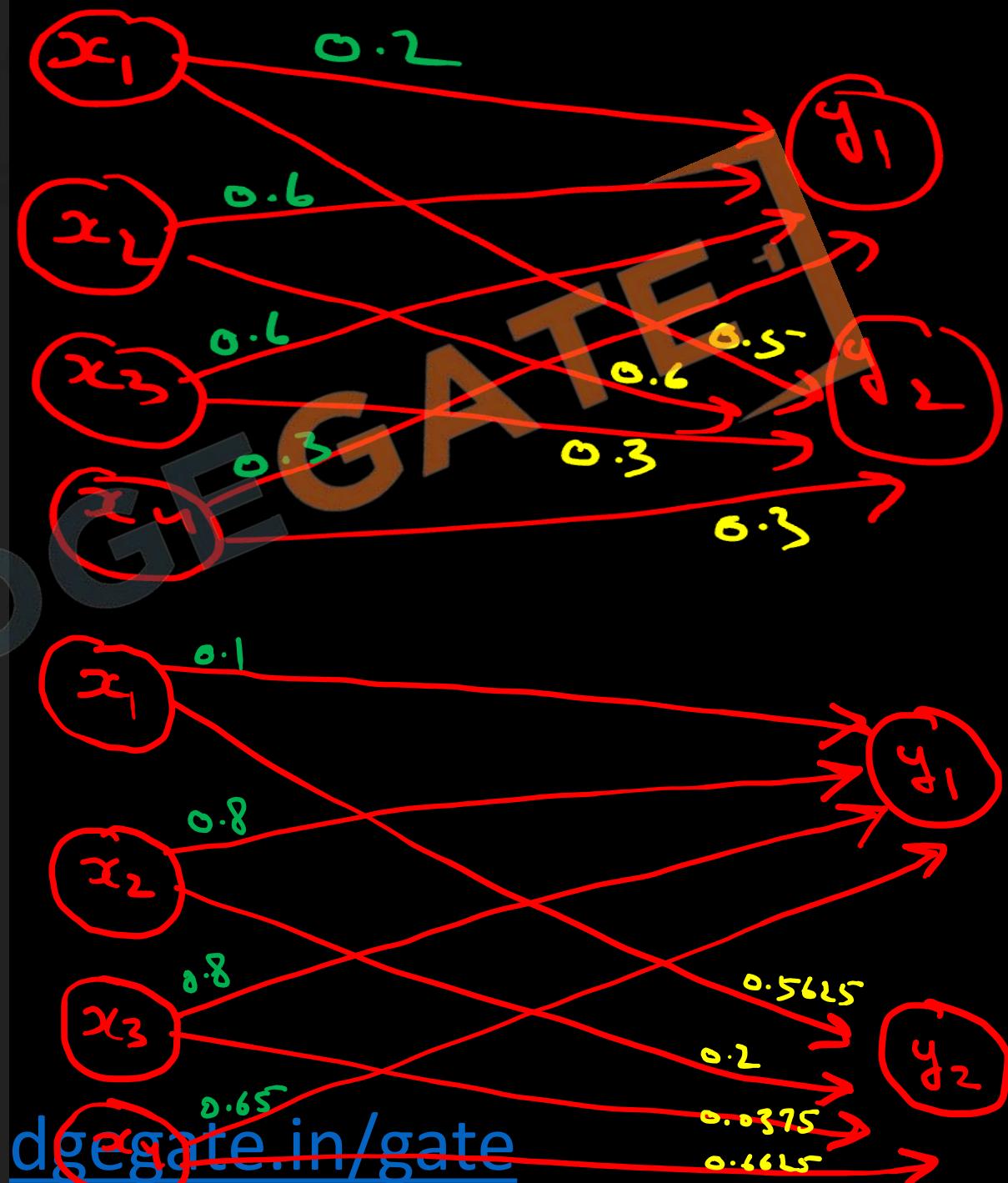
Updated after training with x_1, x_3, x_4 :

$$w_{\text{unit2}}(\text{new}) = [0.5625, 0.2, 0.0375, 0.6625]$$

Summary

- Unit 1 Weights: $[0.1, 0.8, 0.8, 0.65]$
- Unit 2 Weights: $[0.5625, 0.2, 0.0375, 0.6625]$

These are the final weights for both units after the complete training process.



Deep Learning Overview

- Deep learning is a subset of machine learning that involves neural networks with many layers (hence "deep"). These models are capable of learning from large amounts of data and can perform complex tasks by automatically learning features and representations.

<http://www.knowledgegate.in/gate>

Historical Context

- **1940s-1950s**: Early neural network models like the Perceptron were developed.
- **1980s**: Backpropagation algorithm was introduced, enabling the training of multi-layer neural networks.
- **2000s**: Deep learning gained popularity due to increased computational power (GPUs), large datasets, and algorithmic advancements.
- **2012**: Breakthrough in image recognition with AlexNet winning the ImageNet competition, showcasing the power of deep neural networks.

Current Use Deep learning is widely used across various domains:

- **Computer Vision**: Image classification, object detection, facial recognition (e.g., self-driving cars use deep learning for environment perception).
- **Natural Language Processing (NLP)**: Language translation, sentiment analysis, chatbots (e.g., Google Translate, GPT-3).
- **Speech Recognition**: Transcribing spoken words into text (e.g., virtual assistants like Siri, Alexa).
- **Healthcare**: Medical image analysis, drug discovery (e.g., detecting anomalies in X-rays, predicting patient outcomes).
- **Finance**: Fraud detection, algorithmic trading (e.g., identifying fraudulent transactions, stock price prediction).

Advantages

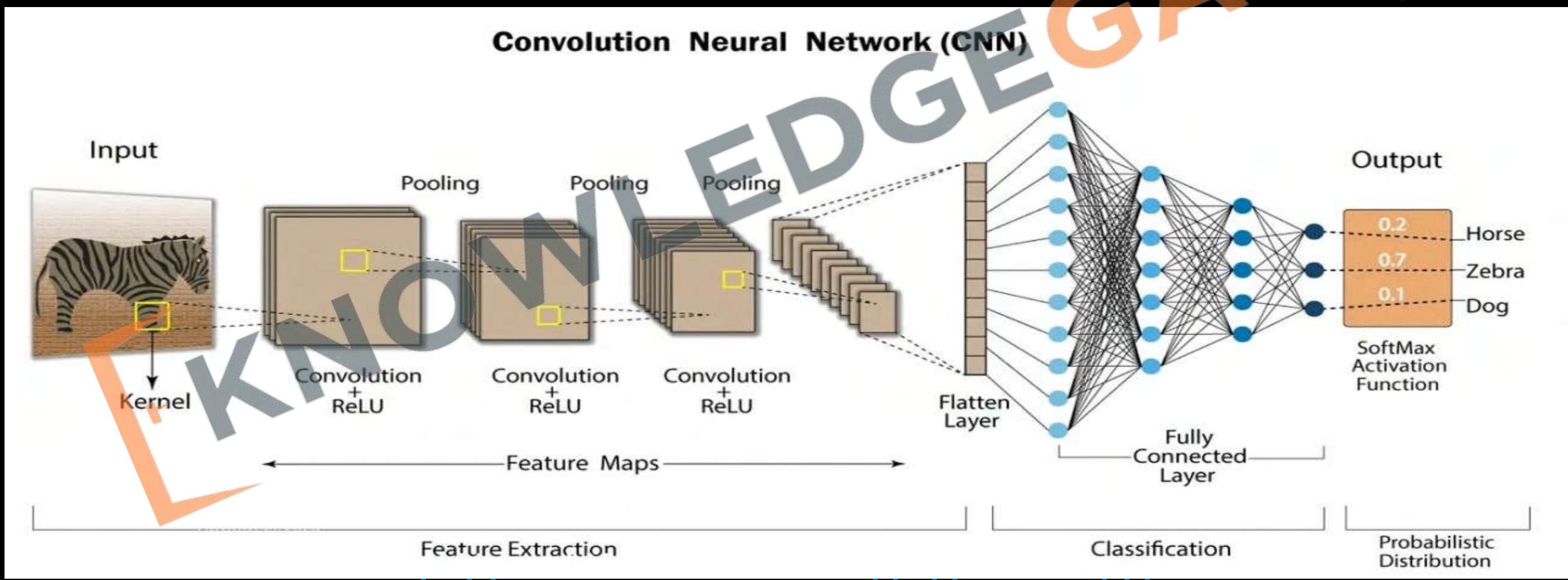
- High Performance: Exceptional accuracy and performance on complex tasks.
- Automatic Feature Extraction: Automatically learns relevant features from raw data.
- Versatility: Applicable to a wide range of problems (vision, speech, text, etc.).

Disadvantages

- Data-Hungry: Requires large amounts of labeled data for training.
- Computationally Intensive: Demands significant computational resources (GPUs, TPUs).
- Black Box: Lack of interpretability; difficult to understand how decisions are made.
- Overfitting: Prone to overfitting if not properly regularized or if insufficient data is available.

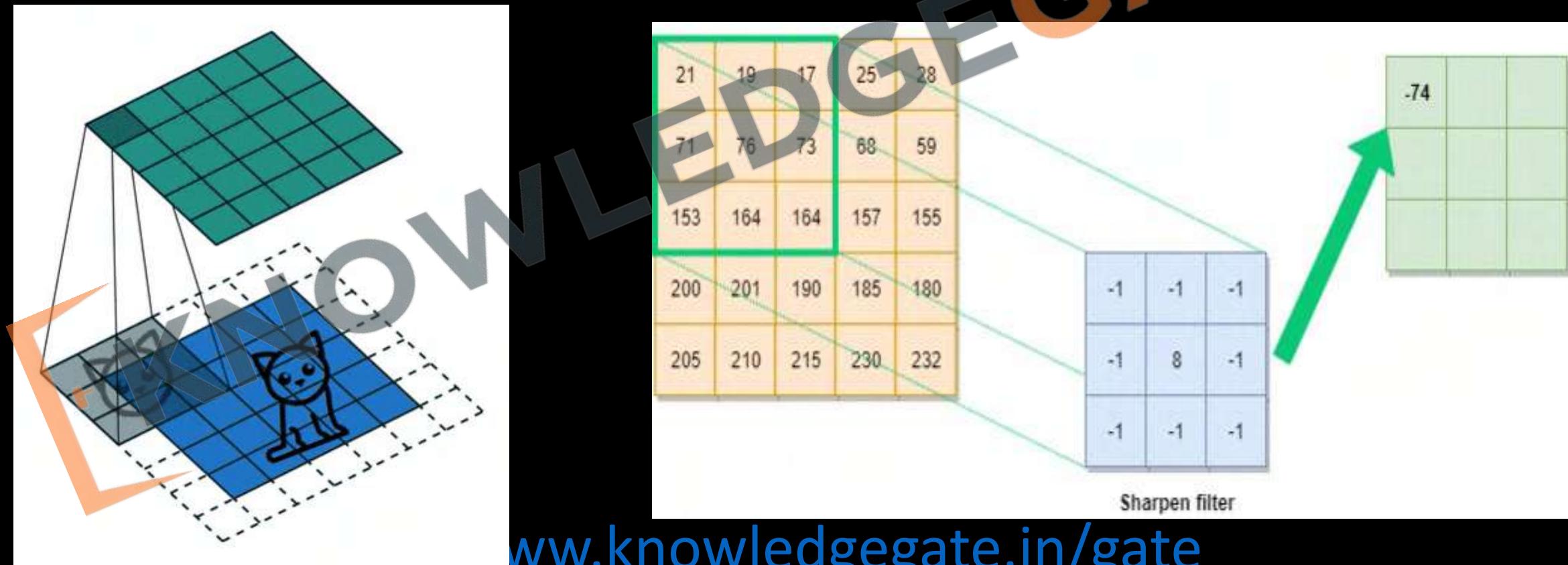
Convolutional Neural Networks (CNNs)

- Convolutional Neural Networks (CNNs) are a class of deep learning models designed specifically for processing data with a grid-like topology, such as images. They are composed of multiple layers that automatically and adaptively learn spatial hierarchies of features from input images.

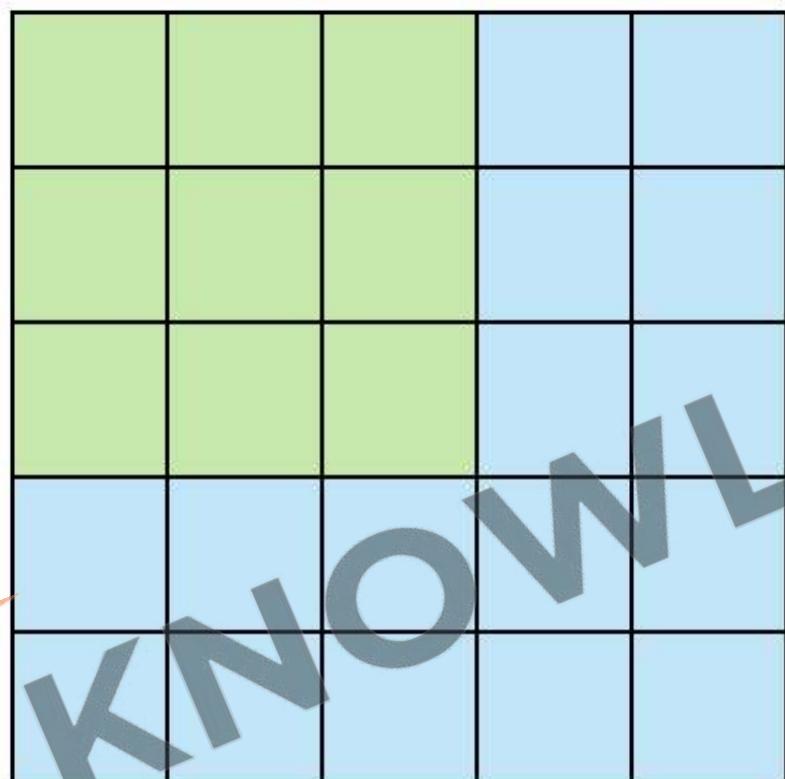


Convolutional Layers

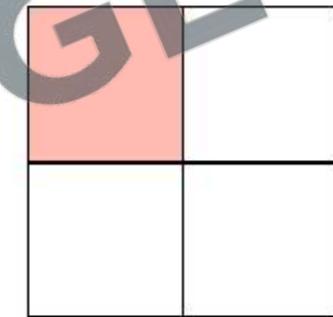
- **Function:** Extract features from the input image by applying convolution operations using filters (kernels).
- **Mechanism:**
 - **Filters/Kernels:** Small matrices (e.g., 3x3, 5x5) that slide over the input image.
 - **Convolution Operation:** Each filter convolves (slides) over the input image, computing the dot product between the filter and the input patch. This generates a feature map.



- **Stride**: The step size by which the filter moves. Larger strides result in smaller feature maps.



Stride 2



Feature Map

- Padding: Adding zeros around the input to preserve the spatial dimensions. 'Same' padding keeps the output size the same as the input, while 'valid' padding reduces it.

0	0	0	0	0	0	0	0
0	60	113	56	139	85	0	0
0	73	121	54	84	128	0	0
0	131	99	70	129	127	0	0
0	80	57	115	69	134	0	0
0	104	126	123	95	130	0	0
0	0	0	0	0	0	0	0

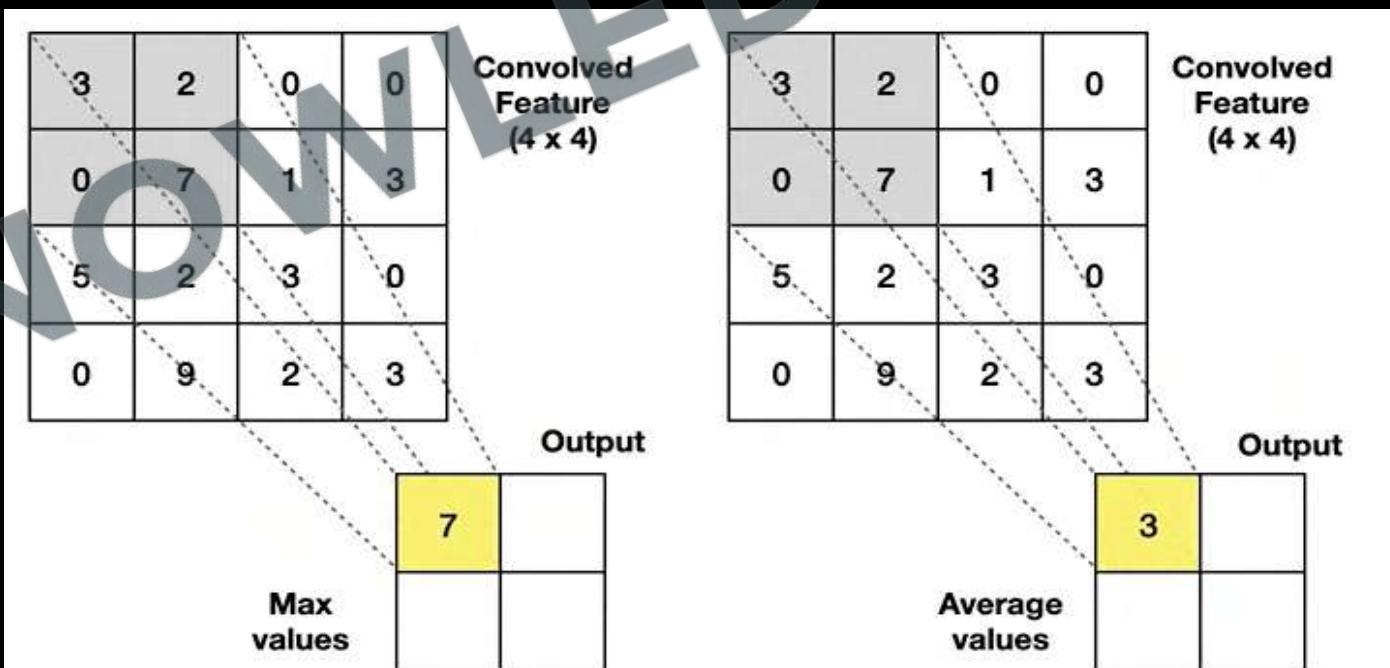
Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

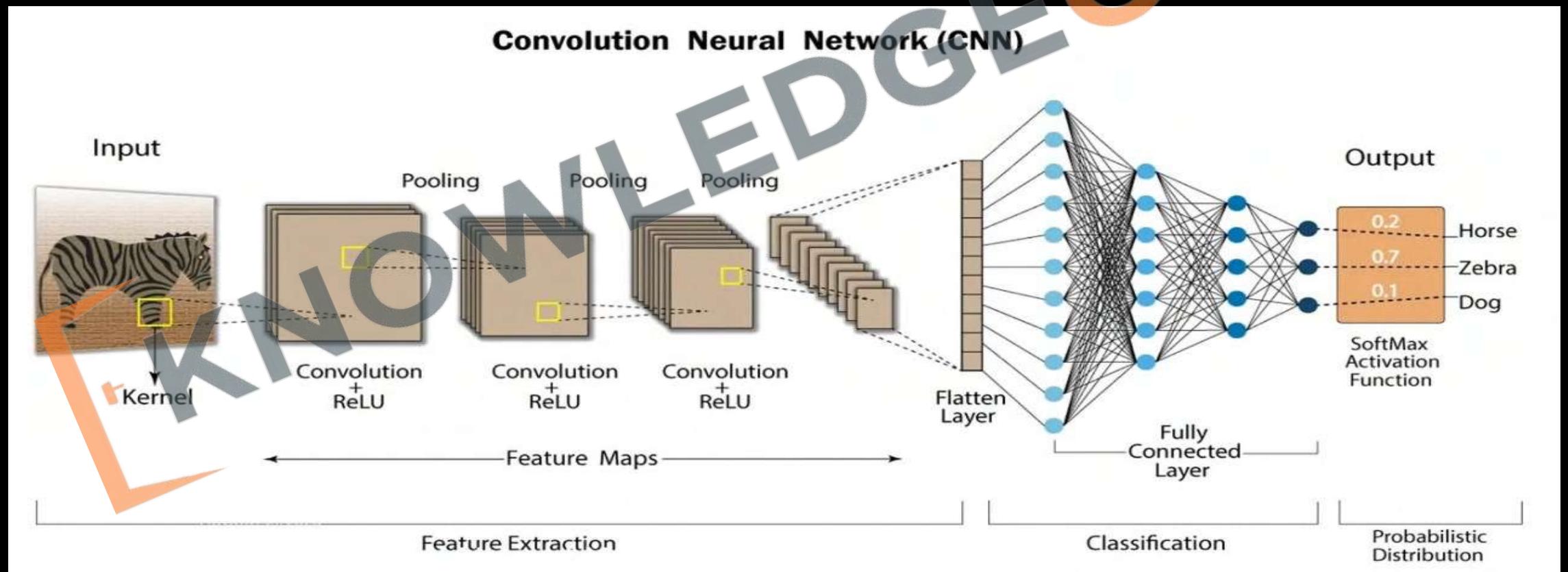
Pooling Layers

- Function: Reduce the spatial dimensions of the feature maps, keeping the most important information.
- Types:
 - Max Pooling: Takes the maximum value from each patch of the feature map.
 - Average Pooling: Takes the average value from each patch of the feature map.
- Mechanism:
 - A pooling layer slides a window (e.g., 2x2) over the input feature map and downsamples it by taking the maximum or average value within the window.



Fully Connected Layers

- **Function:** Perform classification based on the features extracted by the convolutional and pooling layers.
- **Mechanism:**
 - Flatten the output from the last convolutional or pooling layer into a one-dimensional vector.
 - Connect each neuron in the fully connected layer to every neuron in the previous layer.

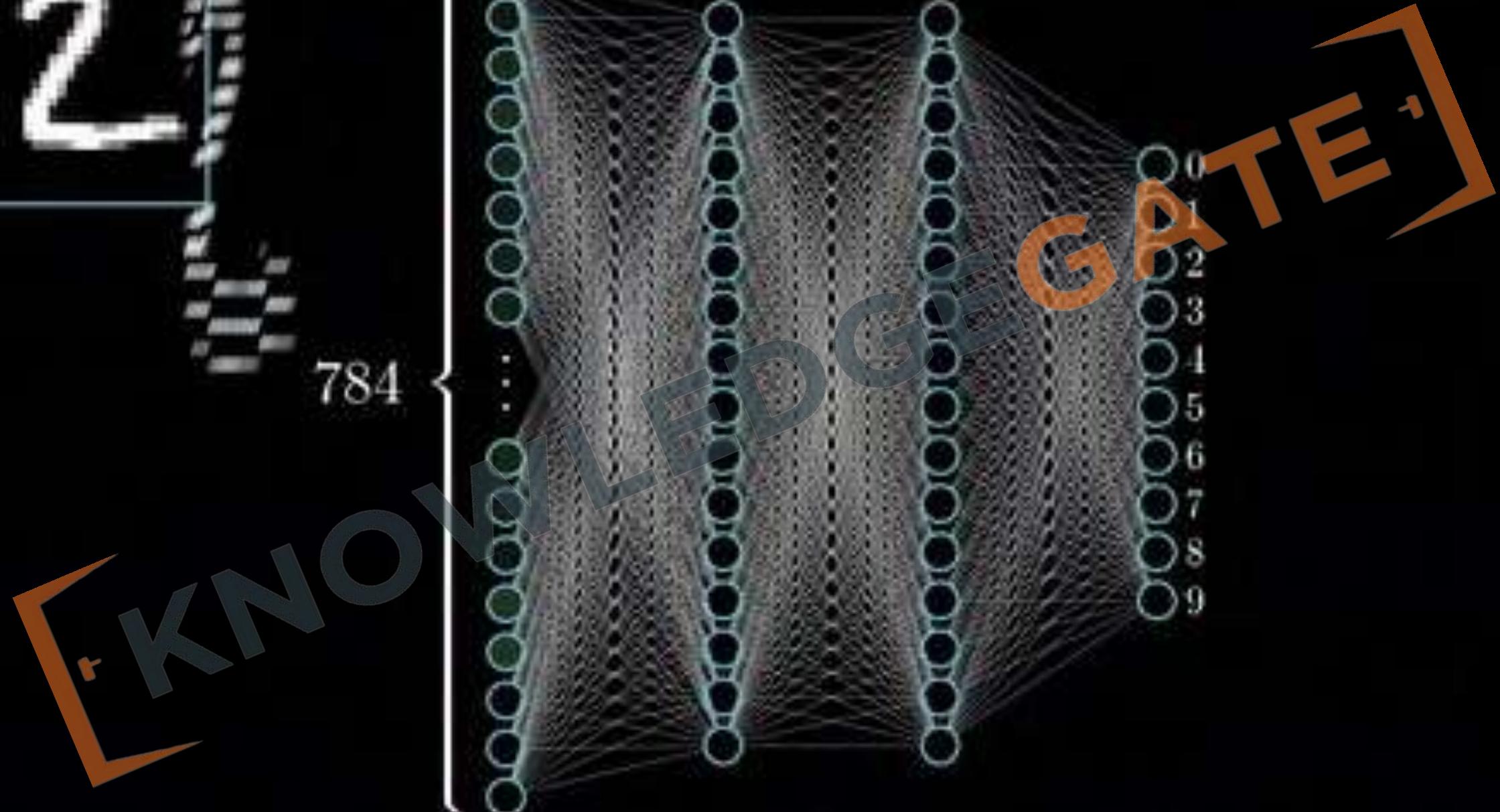


Activation Functions

- **Function:** Introduce non-linearity into the network, allowing it to learn complex patterns.
- **Common Activation Functions:**
 - **ReLU (Rectified Linear Unit):** $\text{ReLU}(x) = \max(0, x)$
 - **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$
 - **Tanh:** $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

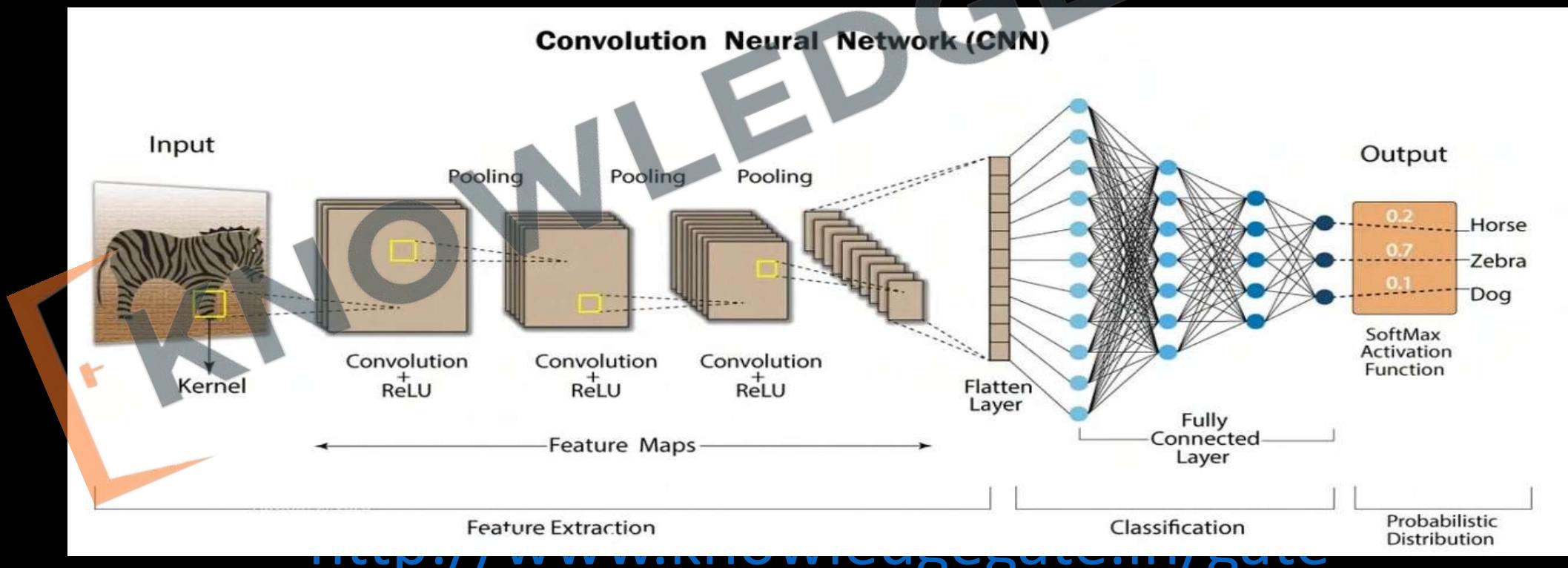
2

784



Dropout Layers

- **Function:** Prevent overfitting by randomly setting a fraction of input units to zero at each update during training.
- **Mechanism:**
 - During training, randomly drop neurons along with their connections.
 - During testing, use all neurons but scale the output to account for dropped units during training.

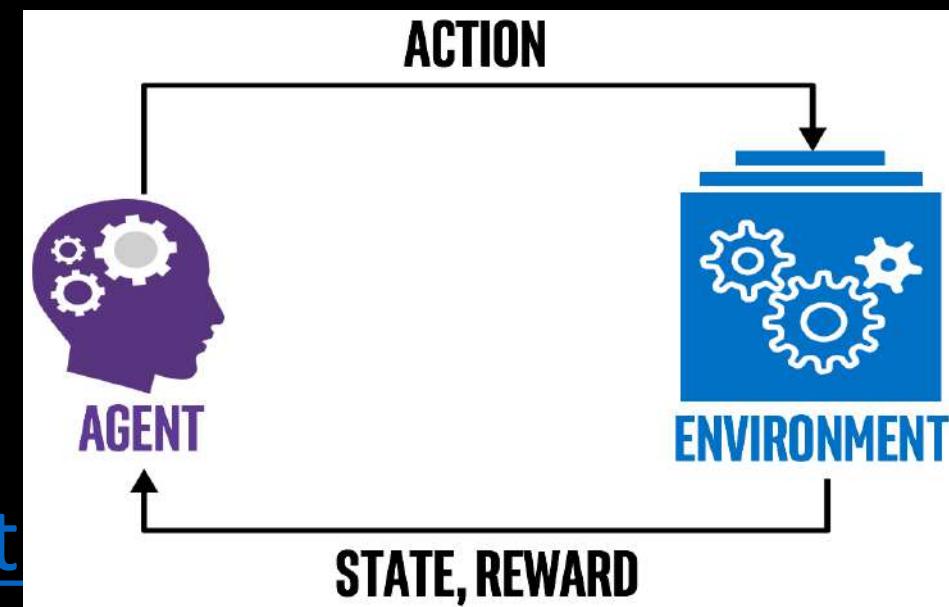


(UNIT-5: REINFORCEMENT LEARNING)

- -Introduction to Reinforcement Learning, Learning Task, Example of Reinforcement Learning in Practice, Learning Models for Reinforcement - (Markov Decision process, Q Learning - Q Learning function, @ Learning Algorithm), Application of Reinforcement Learning, Introduction to Deep Q Learning. GENETIC ALGORITHMS: Introduction, Components, GA cycle of reproduction, Crossover, Mutation, Genetic Programming, Models of Evolution and Learning, Applications.

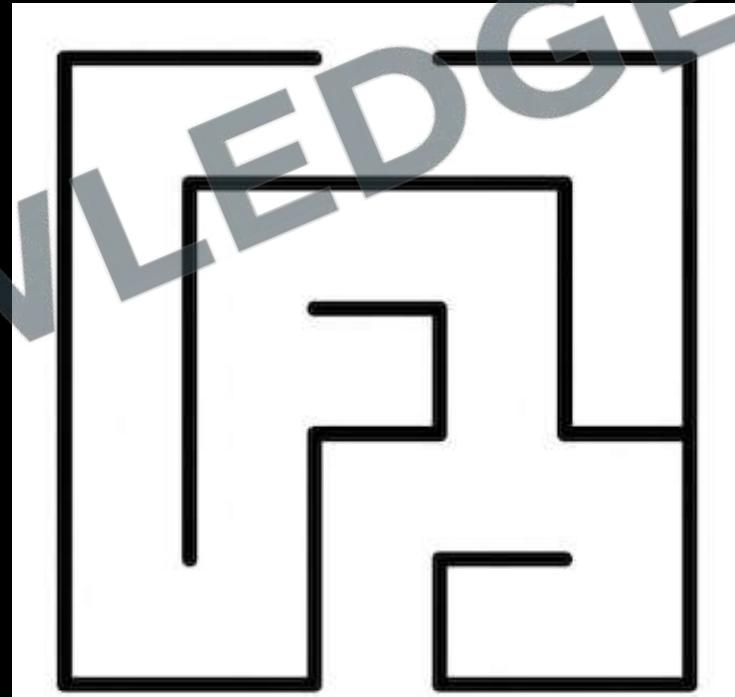
Reinforcement Learning

- Reinforcement Learning (RL) is a feedback-based machine learning approach. An agent learns which actions to perform by observing the environment and the results of its actions.
- **Feedback:**
 - Correct actions receive positive feedback.
 - Incorrect actions receive negative feedback or penalties.
- **Key Components:**
 - **Agent:** Learns and makes decisions.
 - **Environment:** The external system the agent interacts with.
 - **Action:** What the agent does.
 - **State:** The current situation of the agent.
 - **Reward:** Feedback from the environment.



- Process:

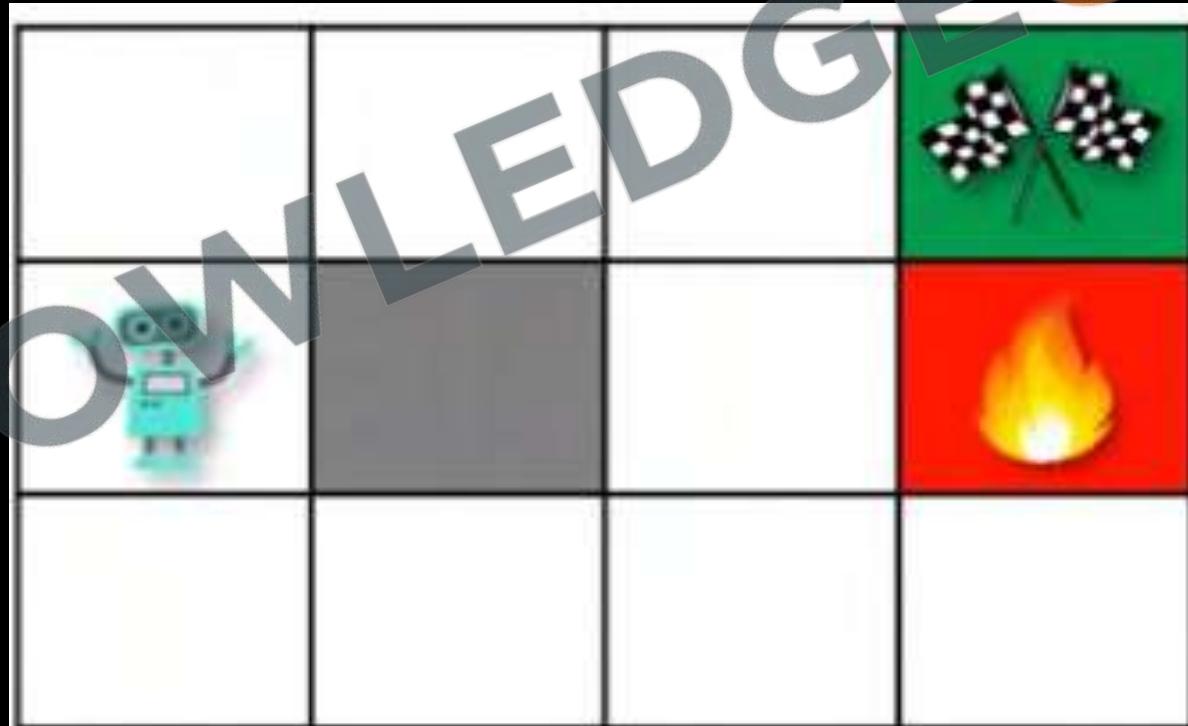
- The agent interacts with the environment.
- It identifies possible actions.
- Performs actions based on observations.
- Receives rewards or penalties.



<http://www.knowledgegate.in/gate>

- **Goals and Learning:**
 - **Primary Goal:** Perform actions that maximize positive rewards.
- **Learning Method:**
 - The agent learns automatically from feedback without labelled data.
 - Unlike supervised learning, RL relies on experience.
- **Application:**
 - **Problem Solving:** Suitable for problems requiring sequential decision-making like game-playing, robotics, etc.
- **Learning Types:**
 - **Positive Reinforcement:** Encourages behaviour by providing positive rewards.
 - **Negative Reinforcement:** Discourages behaviour by providing negative rewards.

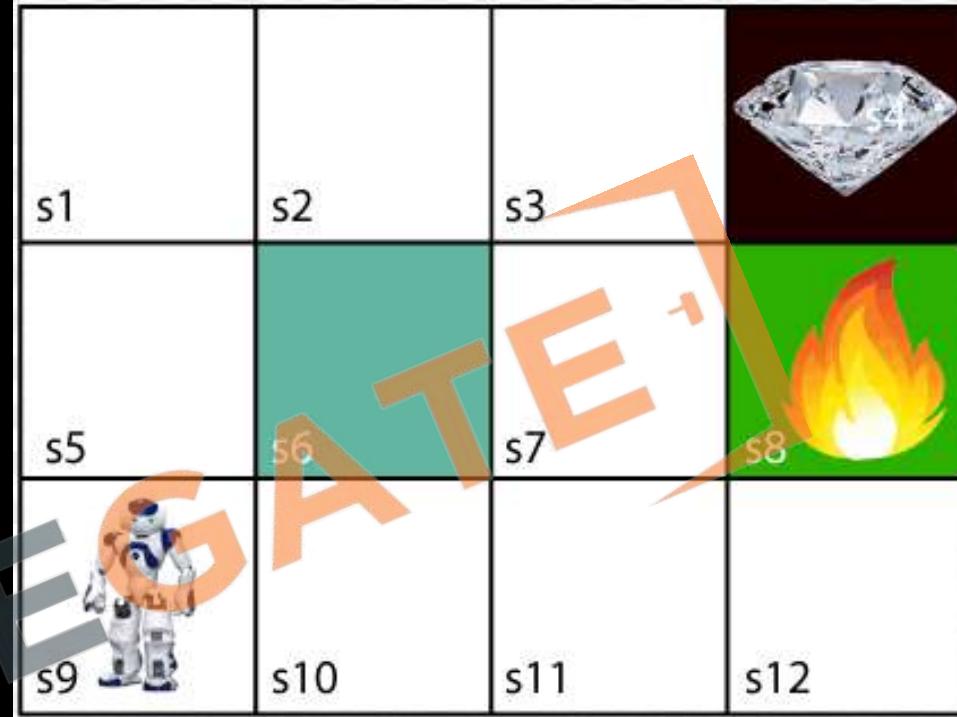
- Examples:
 - Maze Game: Avoiding danger spots to minimize loss.
 - Grid Games: Finding the shortest path or avoiding blocks and danger.
- Limitations:
 - Inapplicability: Not suitable for environments with complete information. For example, object detection and face recognition are better solved using classifiers rather than RL.



Criteria	Supervised Learning	Reinforcement Learning	Unsupervised Learning
Mapping	Present	Present	Not present
Feedback	Instantaneous feedback once the model is created	Constant feedback from environment	No feedback from environment
Supervisor	Presence of supervisor and labeled data	No supervisor and labeled dataset is not available	No supervisor
Decisions	Independent and based on training input	Dependent and made sequentially	Independent
Examples	Classifiers (e.g., image classification)	Chess, Go Games, Robotics	Clustering (e.g., customer segmentation)

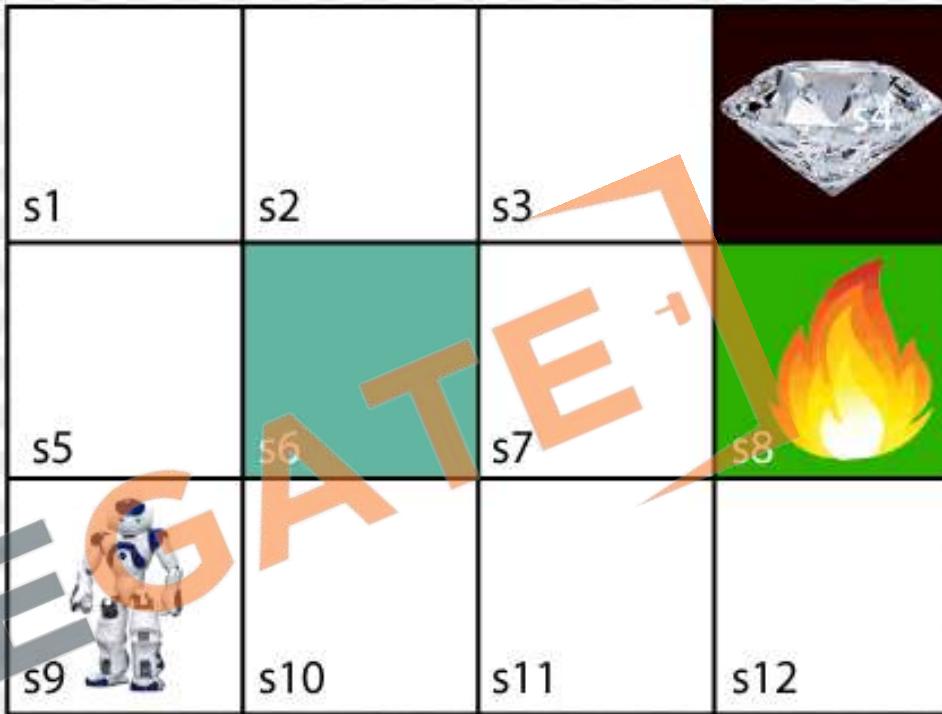
Characteristics of Reinforcement Learning

- **Sequential Decision Making:**
 - Decisions are made in sequence, each action affecting the next. Example: In a maze game, the path from start to goal involves a series of decisions.
- **Actions are Interdependent:**
 - Each action's outcome affects subsequent actions. Example: The agent's next move depends on the reward received from the previous action.
- **Delayed Feedback:**
 - Rewards or penalties are not immediate and can take time to be realized. Example: A robot may need several steps to receive feedback on whether it reached a target or avoided an obstacle.
- **Time-Related:**
 - All actions are associated with time stamps and are ordered in sequence. Example: In a grid game, actions are performed in a specific order over time.
- **Time-Consuming Operations:**
 - The learning process can be time-consuming due to the complexity and number of possible actions and states. Example: Training a model to play a game requires extensive computation due to the large state space.



Challenges of Reinforcement Learning

- **Reward Design:**
 - Designing appropriate rewards and determining their value can be complex. Example: In many games, setting suitable rewards is a significant challenge.
- **Absence of a Model:**
 - Some environments lack a fixed structure or rules, requiring simulations to gather experience. Example: Unlike chess, many games need simulations because they have no underlying model.
- **Partial Observability of States:**
 - Not all states are fully observable, leading to uncertainty. Example: In weather forecasting, complete information about the state is often unavailable.
- **Complexity:**
 - Large board configurations and numerous possible actions increase complexity. Example: Games like Go have vast possibilities, making labeled data unavailable and increasing algorithmic complexity.
- **Time-Consuming Operations:**
 - The need to explore extensive state spaces and possible actions increases the time required for learning. Example: Complex scenarios result in longer computational times, making the training process slow.



Applications of Reinforcement Learning

- **Industrial Automation:**
 - Used to automate and optimize manufacturing processes.
- **Resource Management Applications:**
 - Allocates resources efficiently in various domains.
- **Traffic Light Control:**
 - Reduces congestion by optimizing traffic light timings.
- **Personalized Recommendation Systems:**
 - Provides personalized content recommendations, such as news articles.
- **Bidding for Advertisement:**
 - Optimizes bidding strategies in digital advertising.
- **Driverless Cars:**
 - Used in autonomous vehicle navigation and decision-making.

Markov Decision Process (MDP)

- A Markov Decision Process (MDP) is a mathematical framework used to describe an environment in reinforcement learning. It provides a formal model for decision-making in situations where outcomes are partly random and partly under the control of a decision-maker (agent).
- Characteristics of MDP
- Markov Property:
 - The future state depends only on the current state and action, not on the sequence of events that preceded it.
 - This is known as the "memoryless" property.
- Policy (π):
 - A strategy or rule that defines the action to be taken in each state.
 - Example: A policy that always chooses the action that leads to the highest reward.

1. States (S):

- The set of all possible states in which an agent can exist.
- Example: Different rooms in a building.

2. Actions (A):

- The set of all possible actions an agent can take.
- Example: Moving from one room to another.

3. Transition Function (P):

- A probability function $P(s'|s, a)$ that defines the probability of transitioning to state s' from state s when action a is taken.
- Example: The probability of successfully moving from one room to another.

4. Reward Function (R):

- A function $R(s, a, s')$ that provides the immediate reward received after transitioning from state s to state s' due to action a .
- Example: Receiving a reward for successfully reaching a target room.

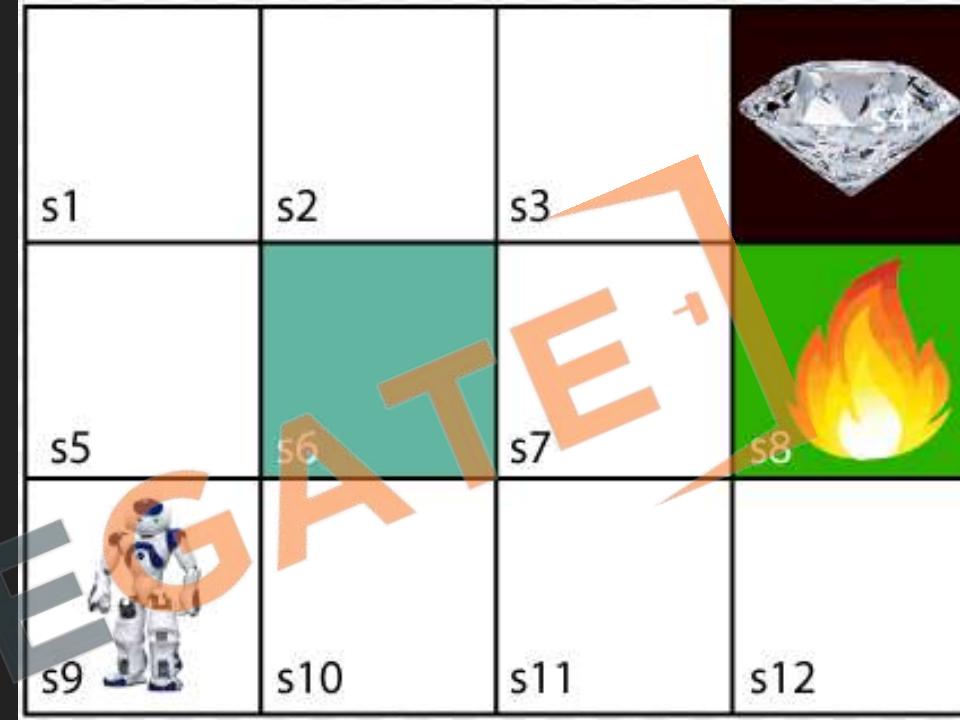
5. Discount Factor (γ):

- A factor $0 \leq \gamma \leq 1$ that represents the difference in importance between future rewards and present rewards.
- Example: Immediate rewards might be more valuable than future rewards.

Example of MDP

Consider a robot navigating a grid environment:

- **States (S):**
 - The cells in the grid (e.g., $S = \{s_1, s_2, s_3, \dots, s_n\}$).
- **Actions (A):**
 - The possible moves (e.g., $A = \{\text{up}, \text{down}, \text{left}, \text{right}\}$).
- **Transition Function (P):**
 - The probability of moving to a new cell based on the chosen action (e.g., $P(s'|s, a) = 0.8$ for successful moves and $P(s'|s, a) = 0.2$ for slipping to an adjacent cell).
- **Reward Function (R):**
 - The reward received for reaching a specific cell (e.g., $R = +10$ for reaching the goal cell, $R = -1$ for falling into a trap).



Q-Learning

- Q-learning is a model-free reinforcement learning algorithm used to find the optimal action-selection policy for any given finite Markov decision process (MDP). It is a value-based method, meaning it focuses on finding the optimal value of the action-state pairs.

Q-Learning Algorithm

1. Initialization:

- For each state s and action a , initialize the table entry $\hat{Q}(s, a)$ to zero.

2. Observation:

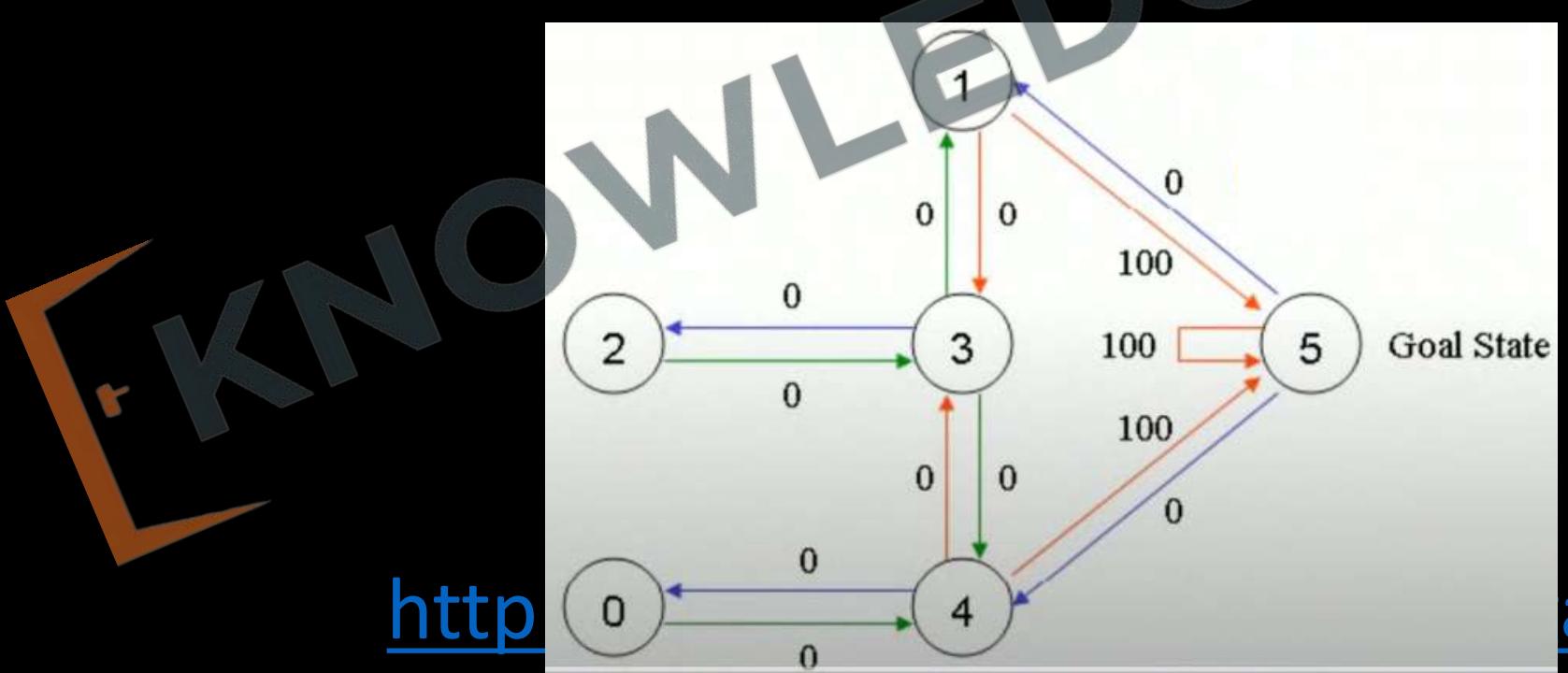
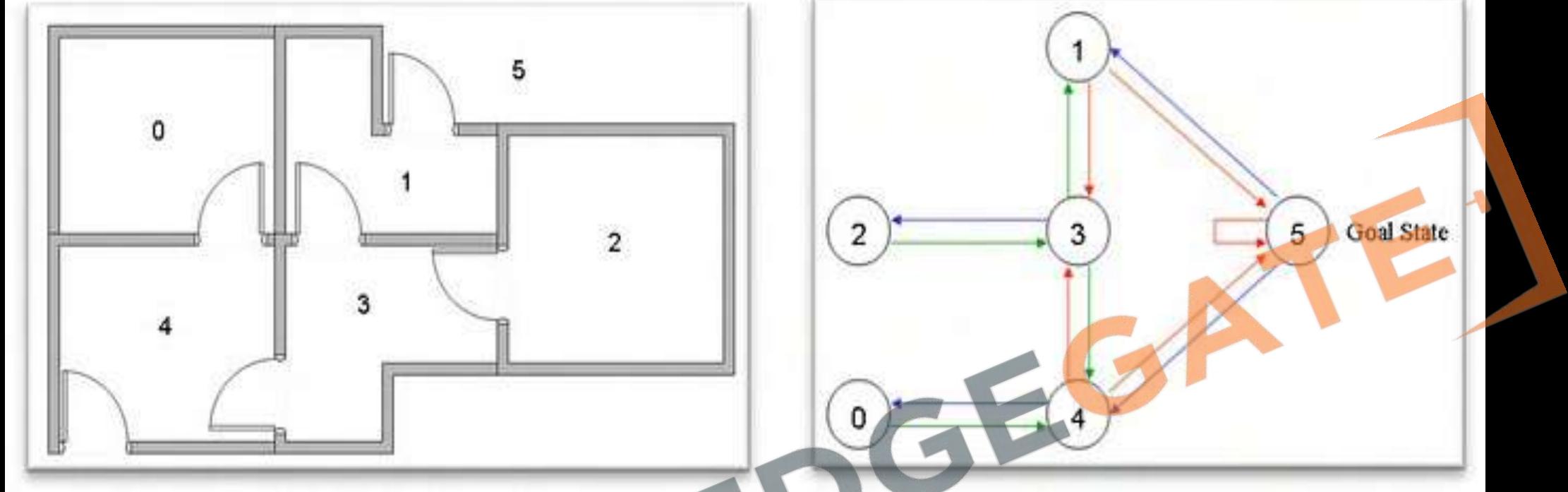
- Observe the current state s .

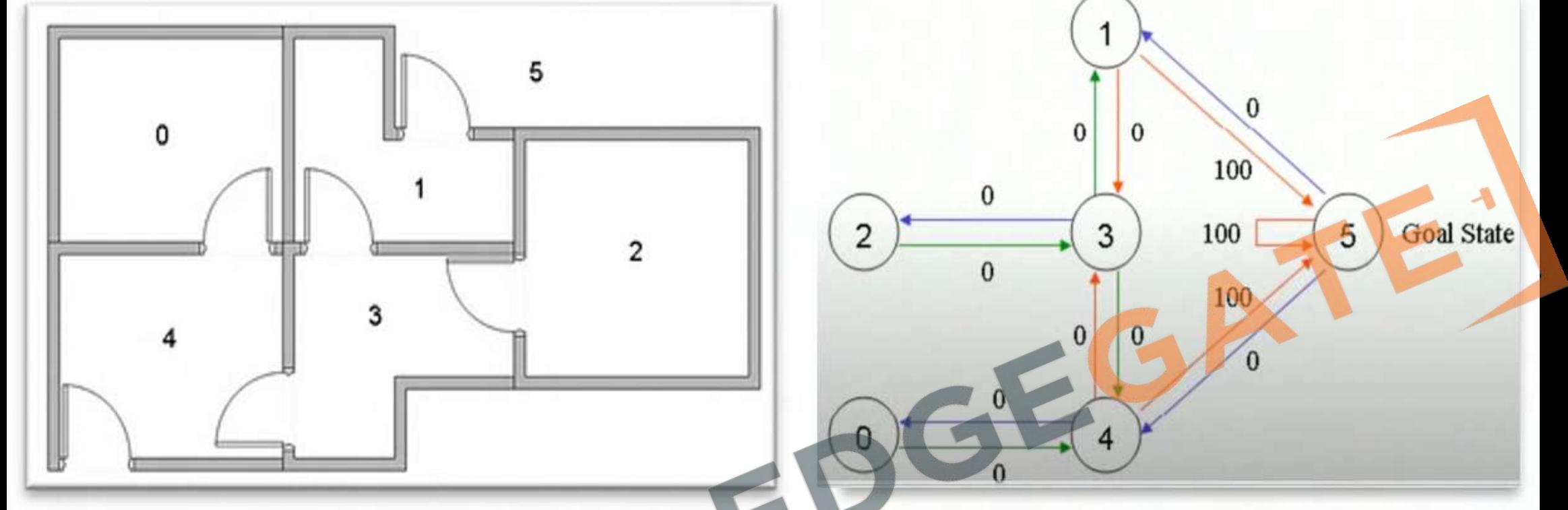
3. Repeat Forever:

- **Select Action:** Choose an action a and execute it.
- **Receive Reward:** Get the immediate reward r .
- **Observe New State:** Note the new state s' .
- **Update Q-Value:** Update the table entry for $\hat{Q}(s, a)$ using the formula:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

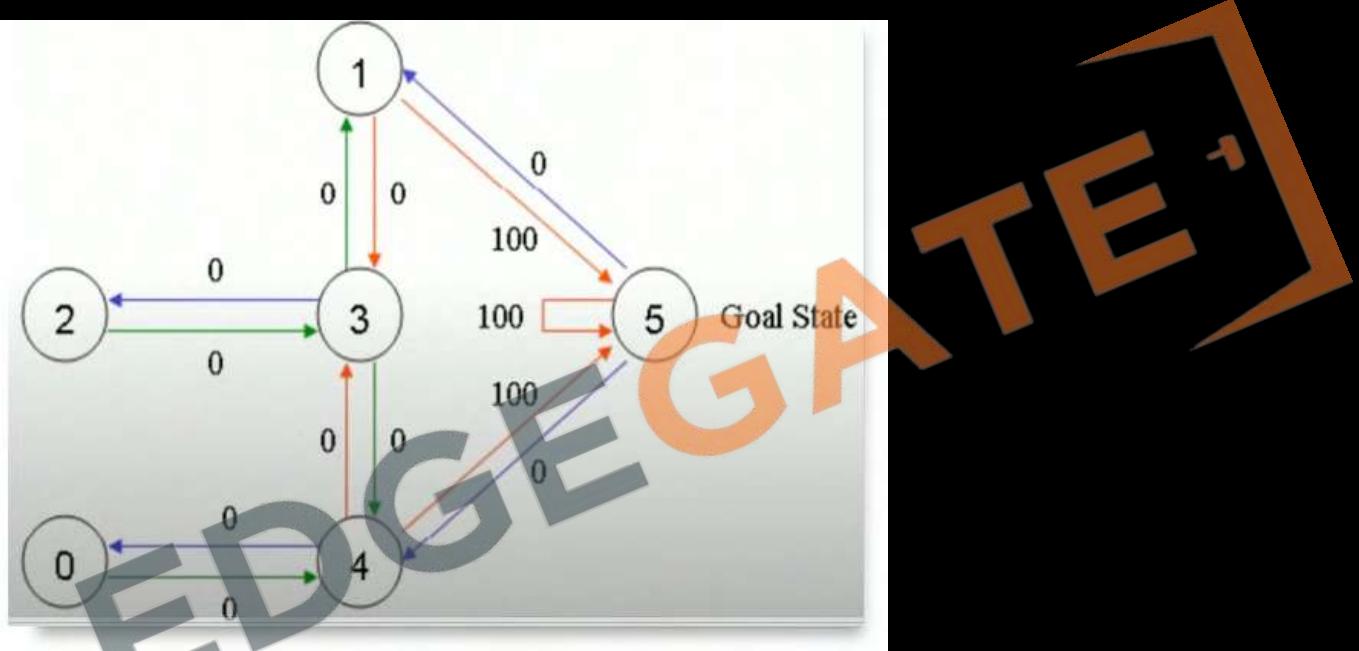
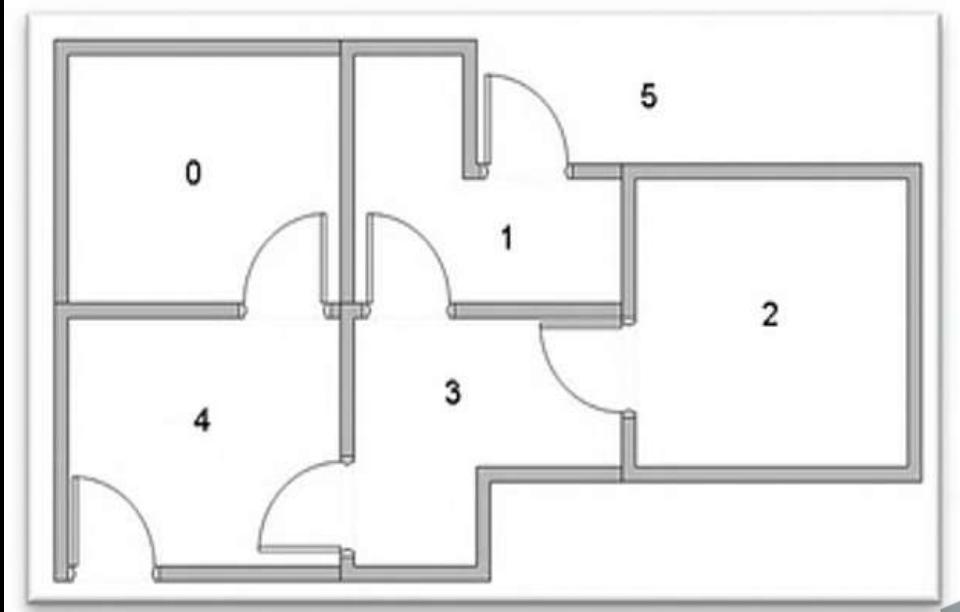
- **Transition:** Set $s \leftarrow s'$.





R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

- Let learning rate =0.8 and the initial state as room 1



R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

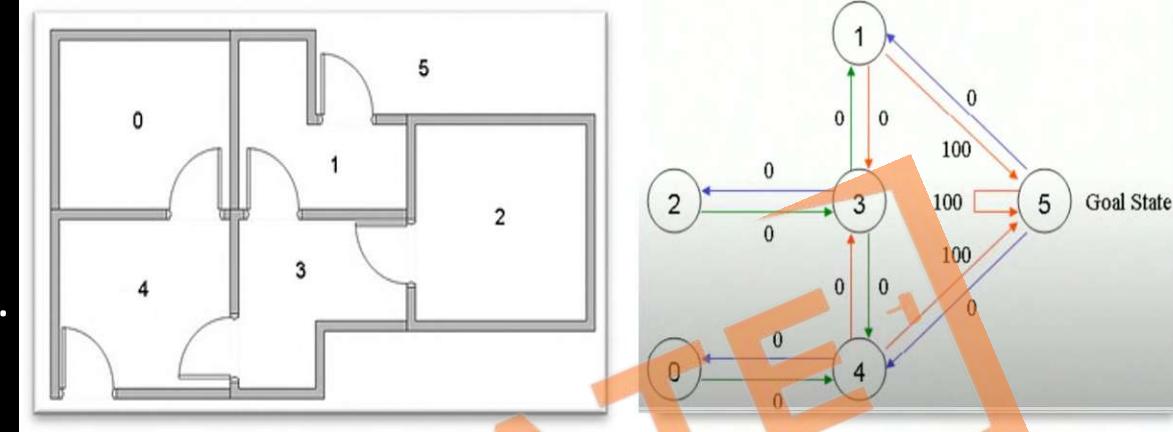
Q	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Consider the second row (state 1) of matrix R :

- There are two possible actions for the current state 1:

- Go to state 3 (Action 3)
- Go to state 5 (Action 5)

- By random selection, we choose to go to state 5 as our action.



- if our agent were in state 5 (next state).
- Look at the reward matrix R (i.e. state 5).
- It has 3 possible actions: go to state 1, 4 or 5.
- $Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$
- $Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$

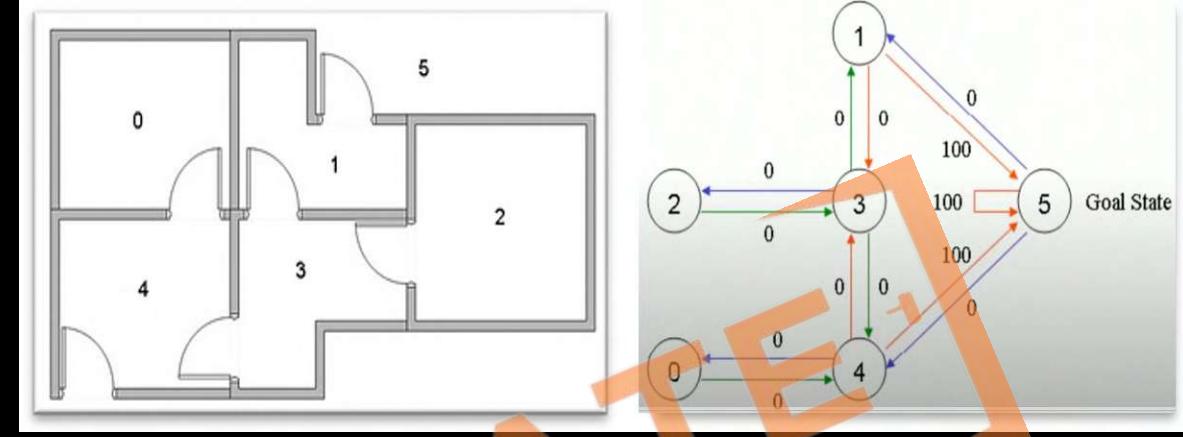
R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Q	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Now lets consider next episode, suppose initial state is 3

We can go to 1 or 2 or 4. and suppose we choose 1.

- if our agent were in state 1 (next state).
- Look at the reward matrix R (i.e. state 1).
- It has 3 possible actions: go to state 3 or 5.
- $Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[\text{Q(next state, all actions)}]$
- $Q(3,1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 3), Q(1, 5)] = 0 + 0.8 * \text{Max}(0, 100) = 80$



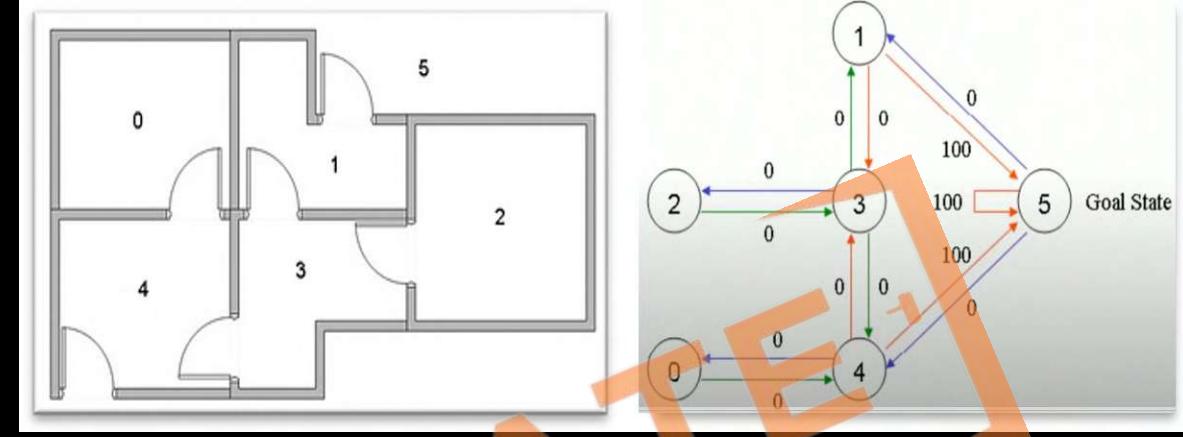
R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Q	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	100
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Now lets consider next episode, suppose initial state is 3

We can go to 1 or 2 or 4. and suppose we choose 1.

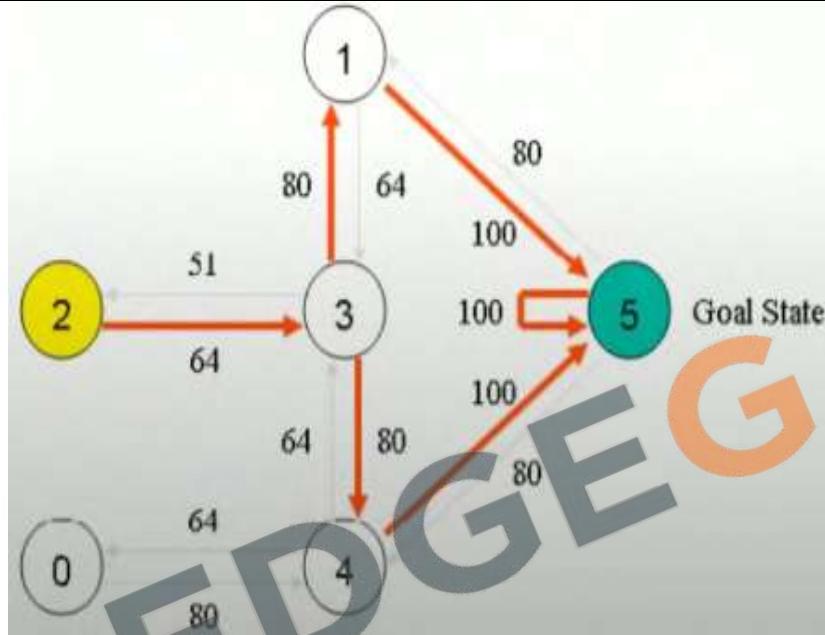
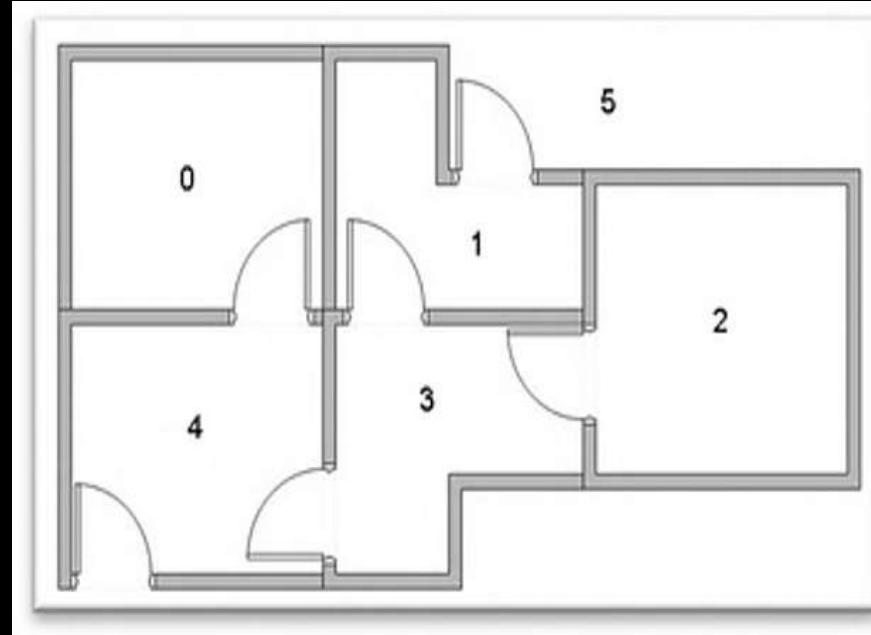
- if our agent were in state 1 (next state).
- Look at the reward matrix R (i.e. state 1).
- It has 3 possible actions: go to state 3 or 5.
- $Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[\text{Q(next state, all actions)}]$
- $Q(3,1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 3), Q(1, 5)] = 0 + 0.8 * \text{Max}(0, 100) = 80$



R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Q	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	100
2	0	0	0	0	0	0
3	0	80	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

After repeating these episode multiple times



R	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Q	0	1	2	3	4	5
0	0	0	0	0	80	0
1	0	0	0	64	0	100
2	0	0	0	64	0	0
3	0	80	51	0	80	0
4	64	0	0	64	0	100
5	0	80	0	0	80	100

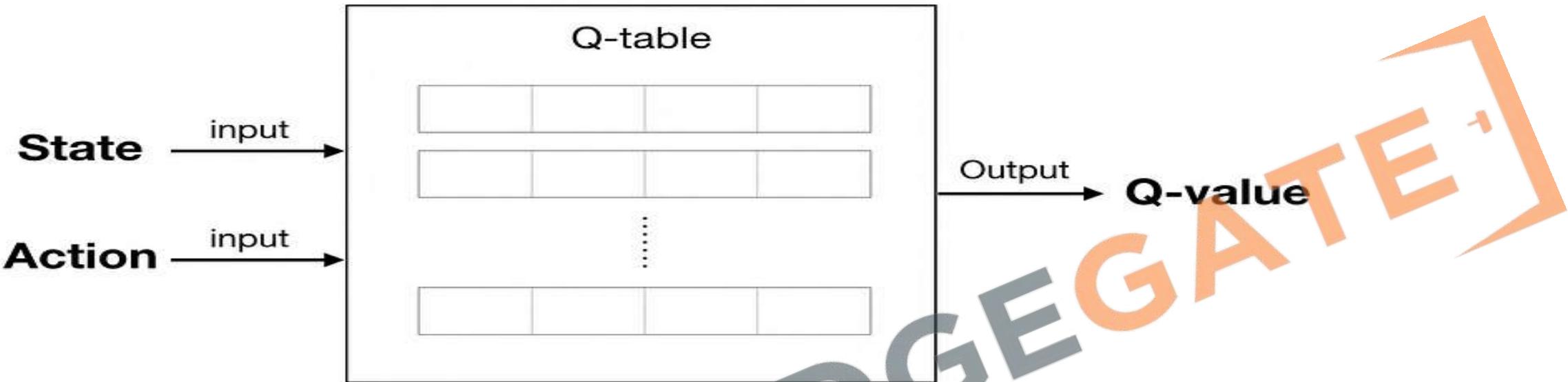
Steps of the Q-Learning Algorithm

1. **Initialize** the Q-values for all state-action pairs to zero (or random values).
2. **Observe** the current state s .
3. **Select** an action a based on the current policy (e.g., ϵ -greedy policy, where ϵ is a small probability of choosing a random action).
4. **Execute** the action a and observe the reward r and new state s' .
5. **Update** the Q-value using the Q-learning update rule.
6. **Set** the new state s' as the current state.
7. **Repeat** steps 3-6 until the convergence of Q-values or for a predetermined number of episodes.

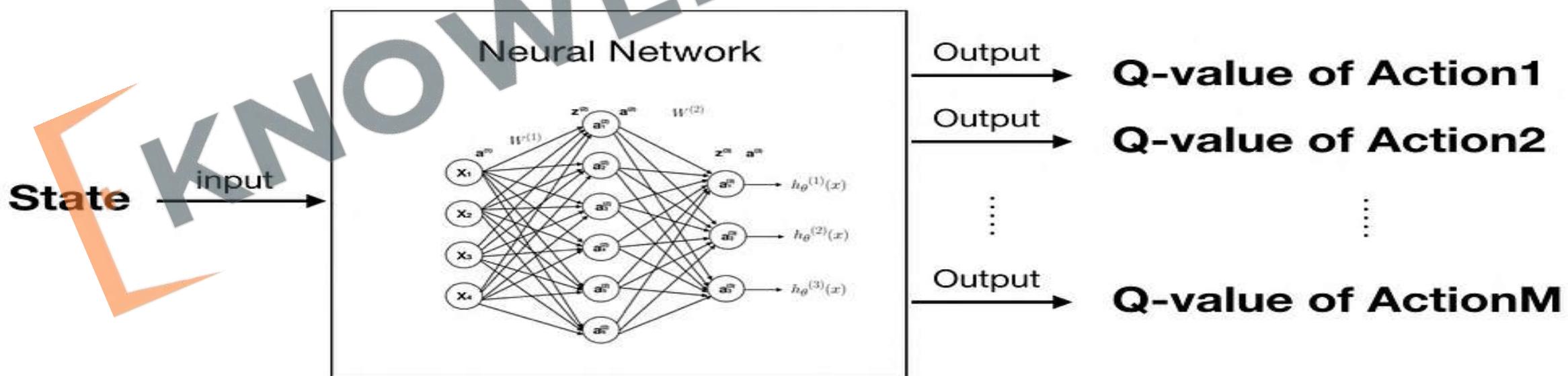
Deep Q-Learning

- Deep Q-Learning (DQL) is an extension of Q-Learning that uses deep neural networks to approximate the Q-values. This method is particularly useful for handling high-dimensional state spaces, where traditional Q-Learning becomes impractical due to the exponential growth in the number of state-action pairs.

Q-Learning



Deep Q-Learning



Benefits:

- Can handle high-dimensional state spaces.
- Generalizes across similar states, reducing the need for exhaustive state-action pair enumeration.
- Experience replay improves sample efficiency and stability.

Challenges:

- Training can be unstable due to the non-stationary target values.
- Requires significant computational resources.
- Hyperparameter tuning (e.g., learning rate, batch size, replay buffer size) can be complex.

GENETIC ALGORITHMS: Introduction

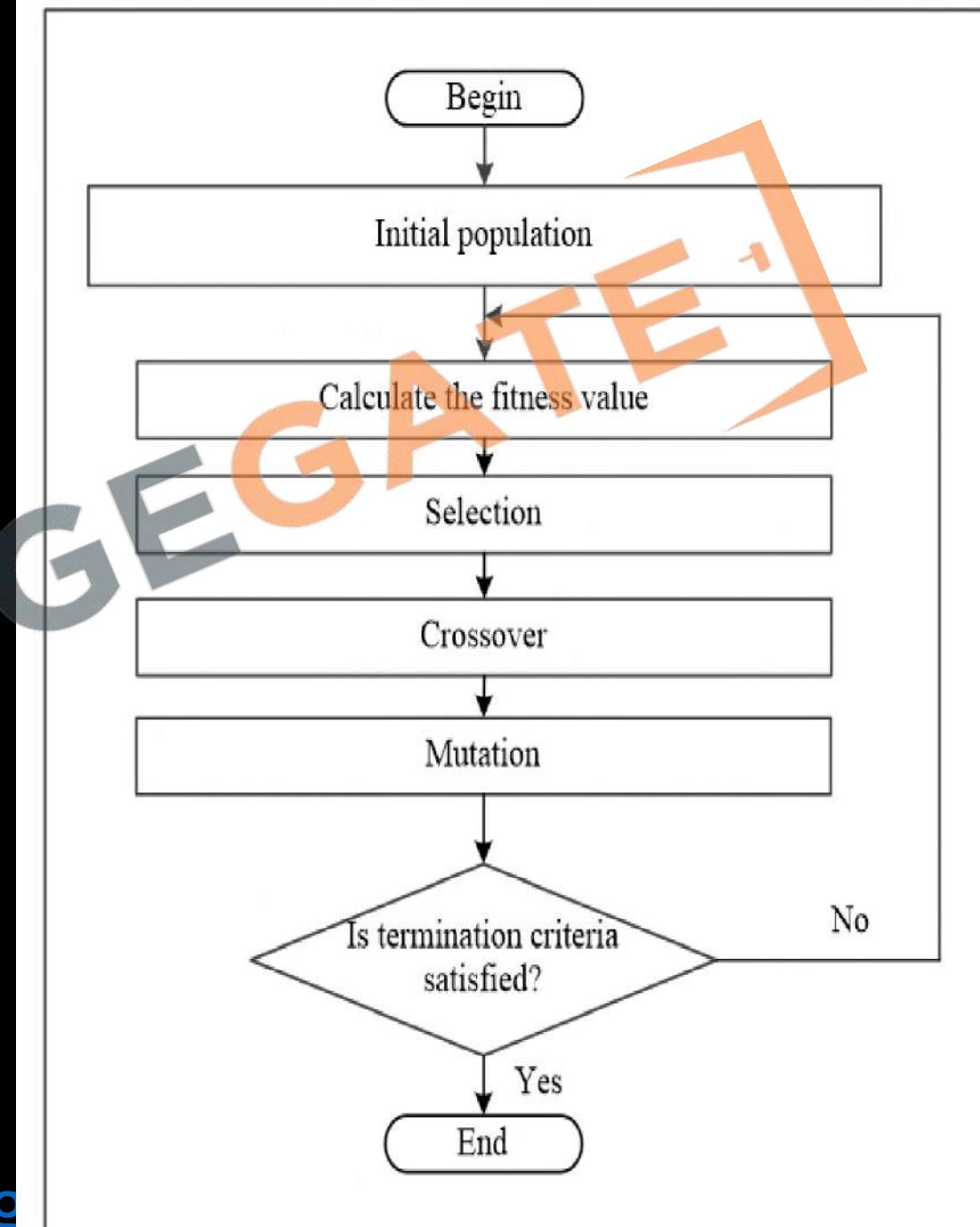
- Evolutionary Algorithms (EAs) are optimization algorithms inspired by the principles of natural selection and genetics. They are used to find approximate solutions to complex problems. Types of Evolutionary Algorithms:
- Genetic Algorithms (GAs): Use binary strings to represent solutions.
 - Solving the Traveling Salesman Problem (TSP)
- Genetic Programming (GP): Evolve computer programs.
 - Symbolic Regression

Genetic Algorithms

- Genetic Algorithms (GAs) are a subset of evolutionary algorithms that model biological processes such as genetics and evolution to optimize highly complex functions. They are particularly useful for solving problems that are difficult to model mathematically or are NP-hard, which are computationally very expensive to solve or involve a large number of parameters.

Steps of a Genetic Algorithm:

- **Initialization**: Generate an initial population of random chromosomes.
- **Evaluation**: Calculate the fitness of each chromosome using the fitness function.
- **Selection**: Select pairs of chromosomes to reproduce based on their fitness (e.g., roulette wheel selection).
- **Crossover**: Combine selected pairs of chromosomes to produce offspring (e.g., single-point crossover).
- **Mutation**: Apply random changes to the offspring to maintain diversity (e.g., bit-flip mutation).
- **Termination**: Repeat the evaluation, selection, crossover, mutation, and replacement steps until a stopping criterion is met (e.g., a maximum number of generations or satisfactory fitness level).



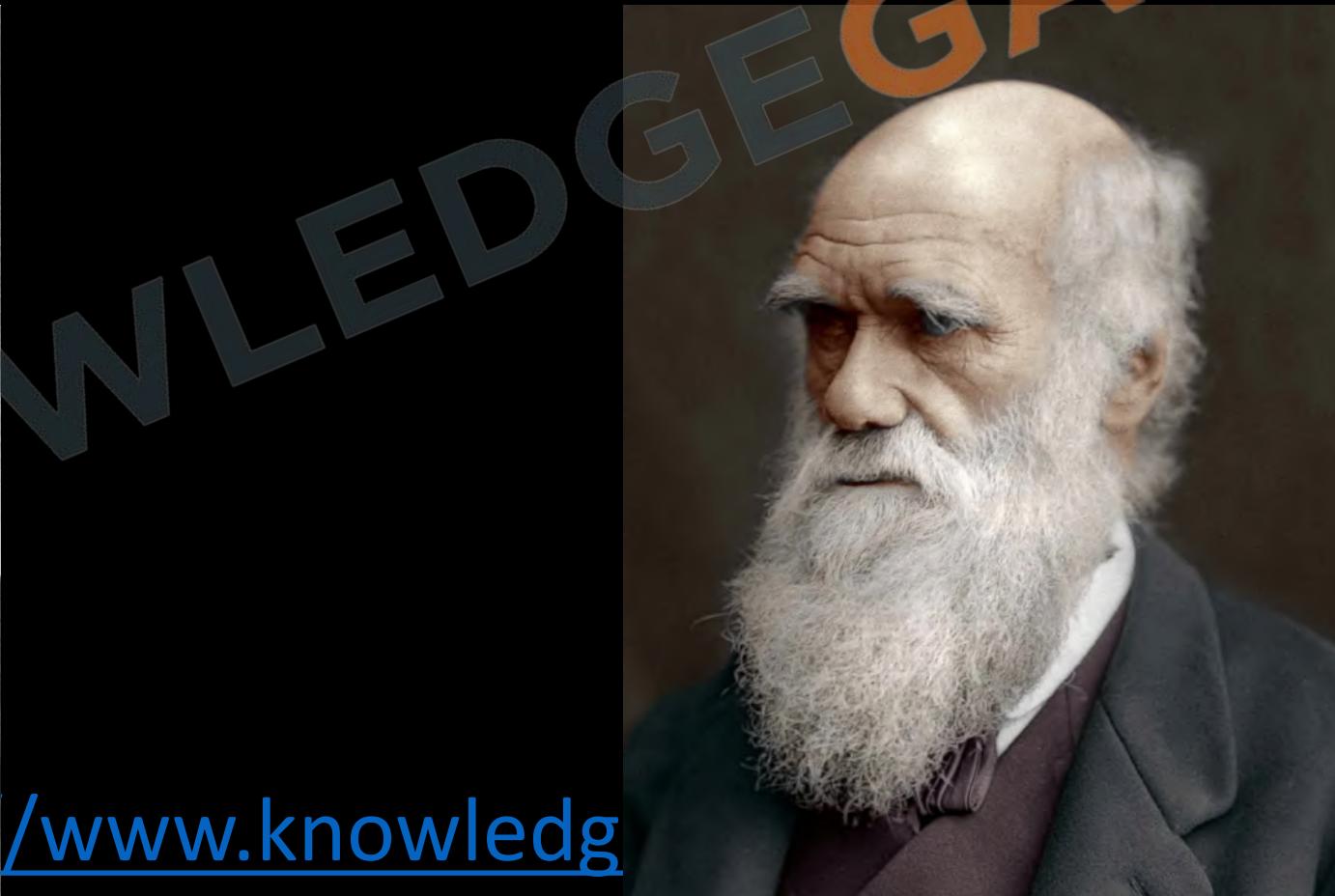
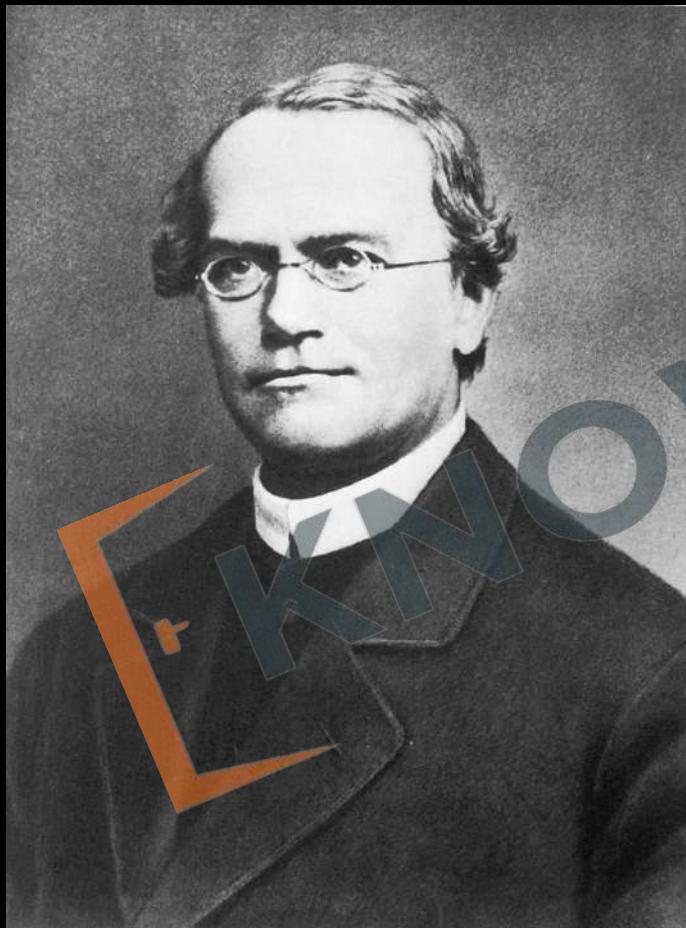
Background of Genetic Algorithm

- Introduction:
 - Introduced by Prof. John Holland at the University of Michigan, USA in 1965.
 - The first article on Genetic Algorithms (GAs) was published in 1975.



<http://www.knowledgegate.in/gate>

- Fundamental Biological Processes:
 - Genetics: Based on the principles of genetics as discovered by Gregor Johann Mendel in 1865.
 - Evolution: Inspired by the theory of evolution by Charles Darwin published in 1875.



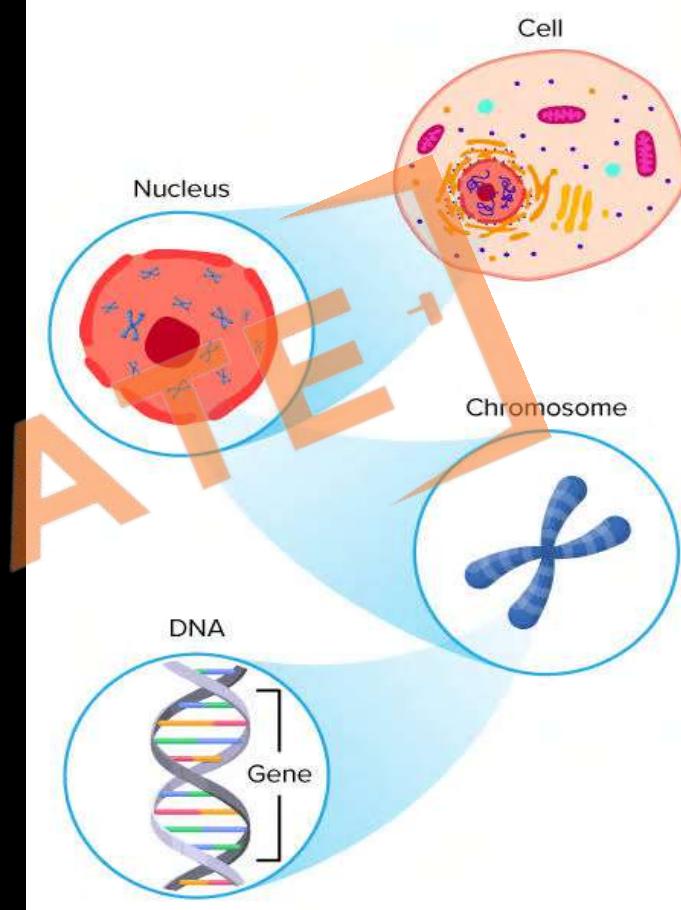
Genetic Algorithm

Based on Two Main Concepts:

- Genetics: Involves cells, chromosomes, genes, and DNA.
- Evolution: Involves heredity, diversity, selection, and ranking.

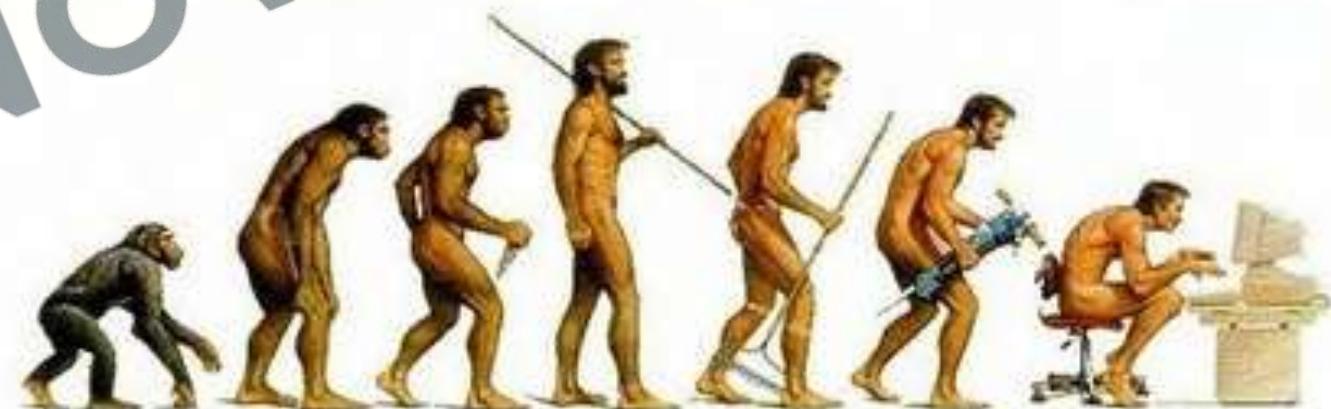
Genetics

- Cells:
 - Basic building blocks of living bodies.
 - Each cell has the same set of chromosomes.
 - Chromosomes:
 - Humans: 46 chromosomes.
 - Frogs: 26 chromosomes.
 - Maize: 6 chromosomes.
 - DNA:
 - Chromosomes are made up of DNA (sequences of 0's and 1's).
 - Genes are segments of DNA.
 - Alleles are different forms of a gene.
 - Example: A gene for eye color might have alleles for blue, brown, etc.
- <http://www.knowledgegate.in/gate>

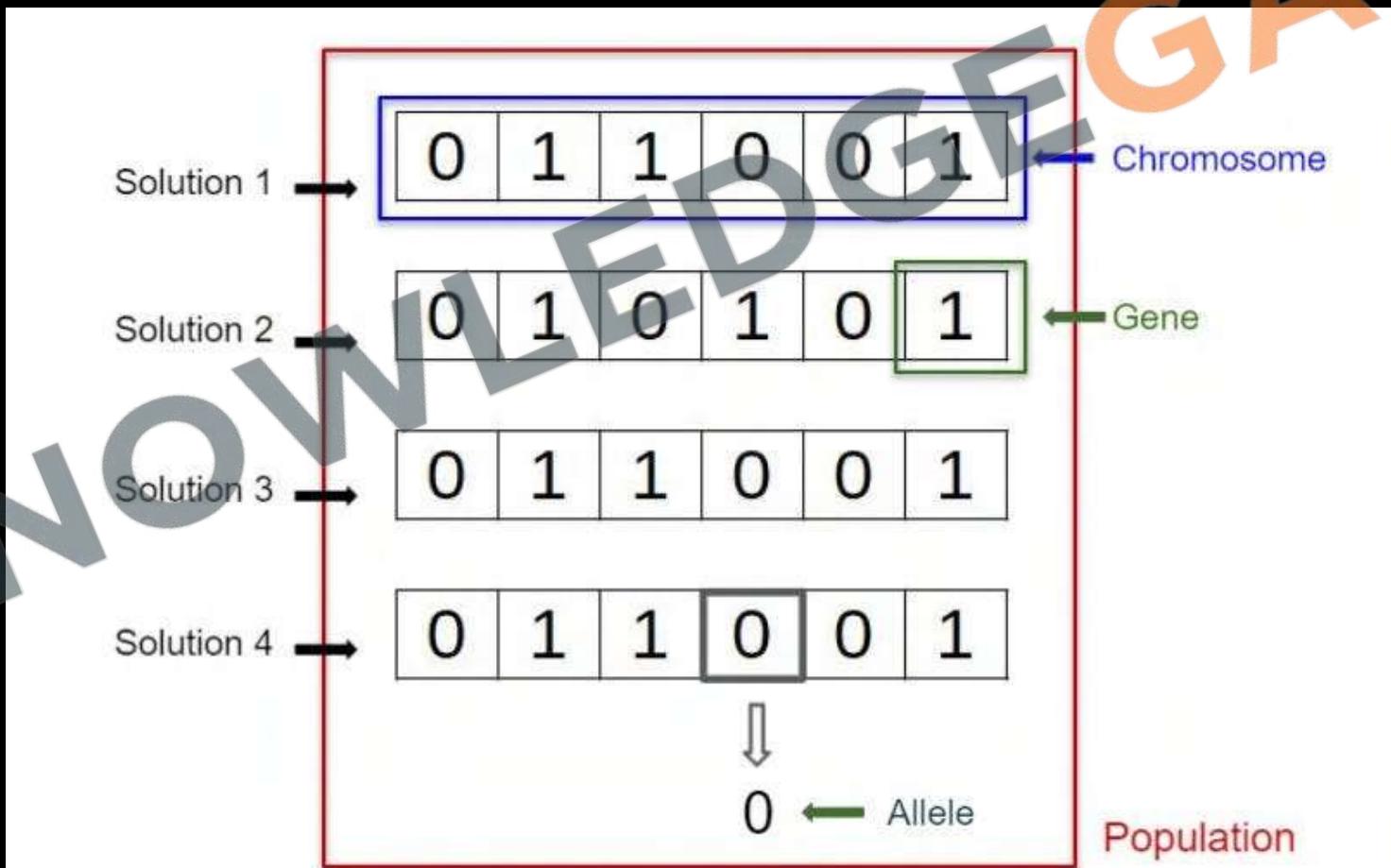


Evolution

- Heredity:
 - Offspring inherit many characteristics from their parents.
- Diversity:
 - Variation in characteristics within the next generation.
- Selection:
 - Only a small percentage of offspring survive to adulthood.
- Ranking:
 - Survival depends on inherited characteristics (natural selection).



- **Definition**: A subset of all possible solutions to a given problem.
- **Chromosome**: One possible solution to the problem.
- **Gene**: An element position of a chromosome.
- **Population**: A set of chromosomes.



Encoding

Encoding is a process of representing individual genes. The encoding depends mainly on solving the problem. Types of Encoding:

- **Binary Encoding:**
 - Description: The most common way of encoding is a binary string.
 - Example:
 - Chromosome 1: 10101011
 - Chromosome 2: 11011001
 - Details:
 - Each chromosome consists of a binary (bit) string.
 - Each bit in the string can represent some characteristic of the solution.
 - Every string is a solution, but not necessarily the best solution.
- **Octal Encoding:**
 - Description: In this encoding, each string uses a set of octal numbers (0 to 7).
 - Example:
 - Chromosome 1: 1534620
 - Chromosome 2: 0772014

- **Hexadecimal Encoding:**
 - Description: This encoding uses strings made up of hexadecimal numbers (0-9, A-F).
 - Example:
 - Chromosome 1: ZAD4F
 - Chromosome 2: A05CE
 - **Permutation Encoding:**
 - Description: In permutation encoding, every chromosome is a string of integers/real numbers.
 - Example:
 - Chromosome 1: 2199456476.1
 - Chromosome 2: 21752437976
 - **Value Encoding:**
 - Description: In value encoding, every chromosome is a string of some values.
 - Details:
 - Values can be anything connected to the problem (integers, real numbers, characters, combination of words, etc.).
 - Example:
 - Chromosome 1: 2156784
 - Chromosome 2: 356259327891
 - Chromosome 3: AB1DFG3
- <http://www.knowledgegate.in/gate>

Genetic Operators

- Genetic operators are used in genetic algorithms to guide the algorithm towards a solution for a given problem. These operators alter (change) the gene composition of the offspring (children). They work in conjunction with one another for the algorithm to be successful.
- Types of Genetic Operators:
 - Selection
 - Crossover
 - Mutation

Selection

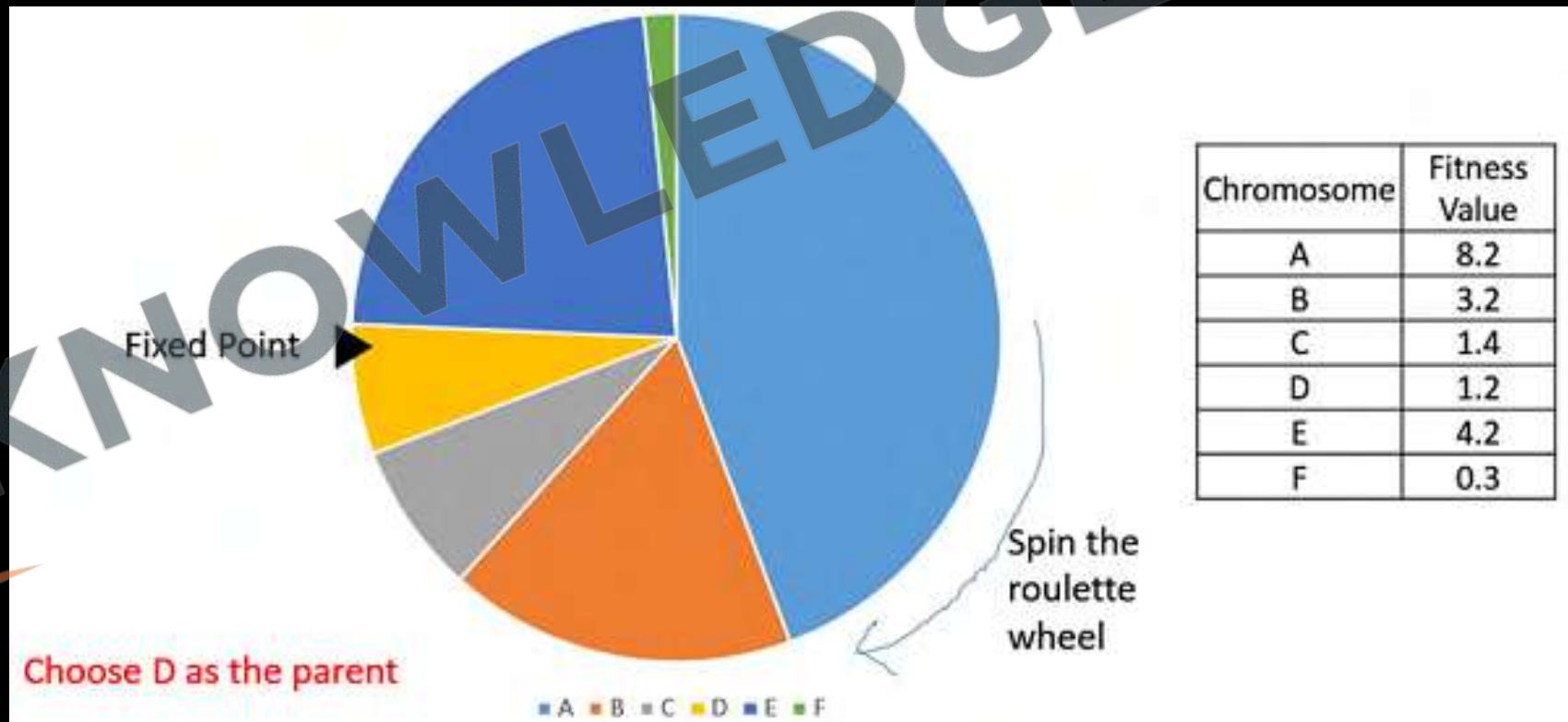
- **Objective**: Emphasize good solutions and eliminate bad solutions in a population while keeping the population size constant.
- **Methods**: Different solutions for choosing the best solution exist (e.g., fitness function, tournament selection).
- **Goal**: Ensure survival of the fittest, keeping the best-performing solutions for reproduction.

Different Techniques for Parent Selection:

- Fitness Proportionate Selection
 - Tournament Selection
 - Rank Selection
 - Random Selection
-
- Fitness Proportionate Selection:
 - Description: Every individual can become a parent with a probability proportional to its fitness.
 - Detail: Fitter individuals have a higher chance of mating and propagating their features to the next generation.
 - Example Method: Roulette Wheel Selection (Stochastic Universal Sampling).

Roulette Wheel Selection

- Process:
 - The circular wheel is divided into parts, where each part is proportional to the fitness value of an individual.
 - Each individual gets a portion of the circle corresponding to its fitness value.
 - A fixed point on the wheel is chosen and the wheel is spun. The region which comes in front of the fixed point is chosen as the parent.
 - The process is repeated to select the second parent.

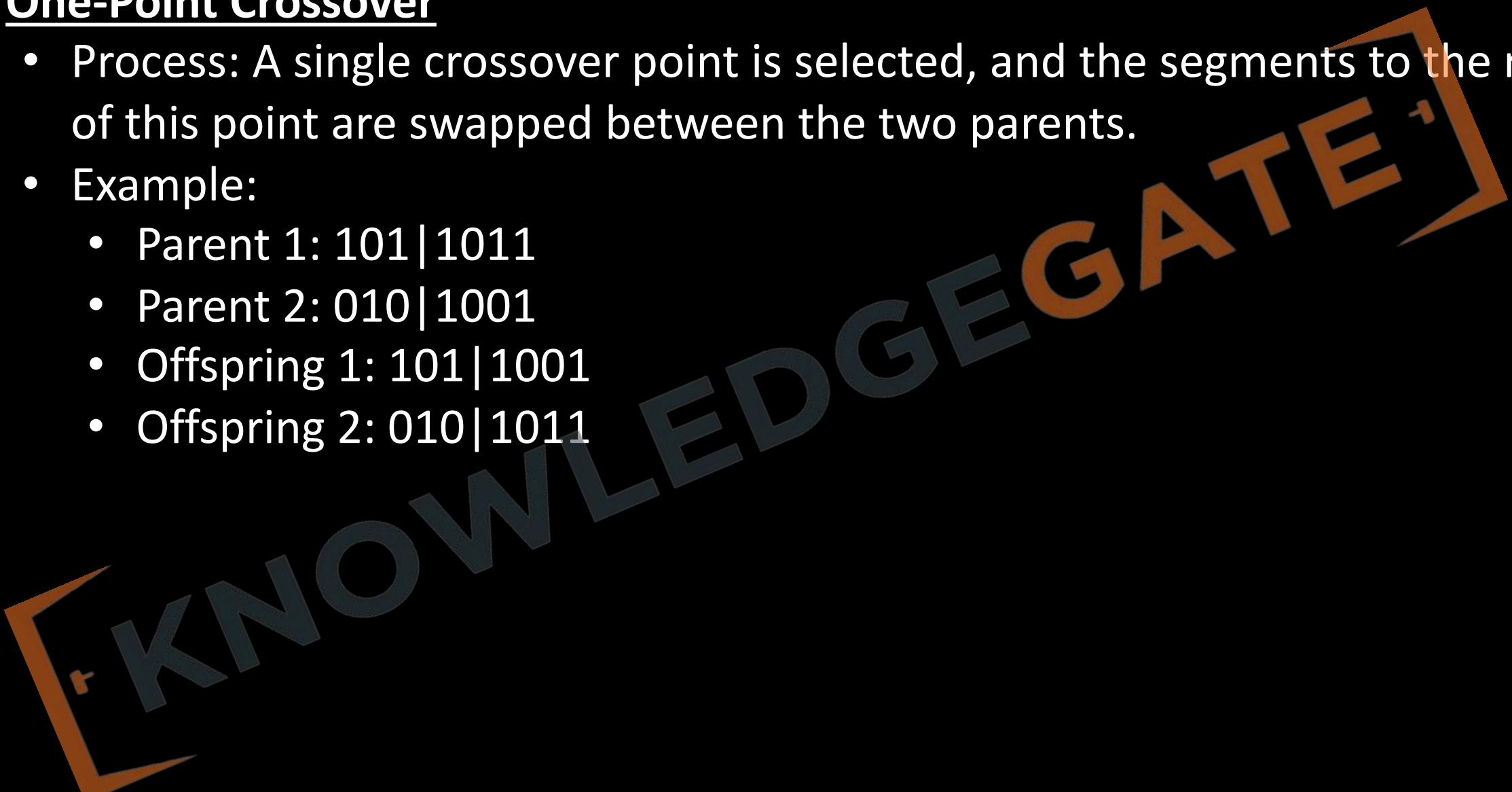


- Example:
 - Chromosomes: 1, 2, 3, 4, 5
 - Fitness values: 29, 18, 16, 6, 26
 - Sum of fitness: 95
 - Probabilities: 0.305, 0.189, 0.168, 0.063, 0.274
 - Cumulative Probabilities: 0.305, 0.494, 0.662, 0.726, 1.000
 - If $r=0.65$, select chromosome 3 since $0.494 < 0.65 \leq 0.662$.

Crossover

- **Process**: Taking more than one parent solution (chromosomes) and producing a child solution from them.
- **Method**: Re-combine portions of good solutions to create better solutions.
- Example:
 - Parent 1: ABCDE | FGH
 - Parent 2: EBCDA | HAF
 - Offspring: ABCDEHAF
- **Goal**: Reproduction using recombination to combine the best features of parent solutions.

- Types of Crossover:
 - **One-Point Crossover**
 - Process: A single crossover point is selected, and the segments to the right of this point are swapped between the two parents.
 - Example:
 - Parent 1: 101|1011
 - Parent 2: 010|1001
 - Offspring 1: 101|1001
 - Offspring 2: 010|1011



- Two-Point Crossover

- Process: Two crossover points are selected, and the segments between these points are swapped between the two parents.
- Example:
 - Parent 1: 10|110|11
 - Parent 2: 01|011|01
 - Offspring 1: 10|011|11
 - Offspring 2: 01|110|01

- Uniform Crossover

- Process: Each gene is treated separately, and we decide for each gene whether to swap it based on a predefined probability or using a binary mask.
- Example:
 - Parent 1: 00101100
 - Parent 2: 11010011
 - Binary Mask: 11010110
 - Child 1: 01101100 (biased to Parent 1)
 - Child 2: 10010011 (biased to Parent 2)

- **Half-Uniform Crossover**
 - Method:
 - Calculate the Hamming distance (number of different bits) between Parent 1 and Parent 2.
 - Swap the bits in Offspring 1 and 2 if the bits are different; otherwise, they remain the same.
 - Example:
 - Parent 1: 11100010
 - Parent 2: 01011011
 - Hamming Distance = 4
 - Offspring 1: 01101010
 - Offspring 2: 11010001
 - **Three-Parent Crossover**
 - Method:
 - Three parents are taken.
 - Each bit of Parent 1 is compared with each bit of Parent 2.
 - If the bit is the same in both parents, it is taken in the offspring. If different, the bit from Parent 3 is taken for the offspring.
 - Example:
 - Parent 1: A B C A D B E A
 - Parent 2: B D E A C D B E
 - Parent 3: D A B C D B E A
 - Offspring: D B C C D B B A
- <http://www.knowledgegate.in/gate>

- **Shuffle Crossover**

- Method:
 - A single crossover point is selected, dividing the chromosome into two parts.
 - Shuffle bits (genes) in each parent using any logic.
 - After shuffling, mix modified parents as in the single crossover method.
- Example:
 - Parent 1: 1 2 3 4 | 5 6 7 8
 - Parent 2: 8 1 7 6 | 2 5 3 4
 - Shuffled Parent 1: 3 5 1 4 2 7 6 8
 - Shuffled Parent 2: 5 7 8 6 1 2 3 4
 - Offspring 1: 3 5 1 4 5 7 8 6
 - Offspring 2: 5 7 8 6 2 7 6 8

Mutation

- **Objective**: Encourage genetic diversity among solutions and attempt to provide genetic algorithms with solutions that are slightly different from the previous generation. Mutation alters one or more gene values in a chromosome from its initial state. With the new gene values, the genetic algorithm may be able to arrive at better solutions.
- **Process**: A small change in the DNA segment of a chromosome to produce an improved solution.
- **Example**:
 - Before Mutation: ABCDEHAF
 - After Mutation: ABCDFHAF
- **Goal**: Introduce small changes to maintain diversity and avoid local optima.

- Types of Mutation:
 - **Flip Bit Method (Flipping)**
 - **Description:**
 - For a binary encoded chromosome, the corresponding bit is flipped (0 to 1 or 1 to 0).
 - **Example:**
 - Offspring Chromosome: 00100100
 - Mutation: Flip the second bit.
 - Mutated Chromosome: 01100100
 - **Boundary Method**
 - **Description:**
 - The mutation operator replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly).
 - **Example:**
 - Offspring Chromosome: 12543173
 - Mutation: Replace the first gene with its upper bound.
 - Mutated Chromosome: 17543173

- **Interchange Method**
 - Description:
 - Two positions of the child's chromosome are chosen randomly, and the bits corresponding to these positions are interchanged.
 - Example:
 - Offspring Chromosome: 10100100
 - Mutation: Interchange the second and sixth bits.
 - Mutated Chromosome: 11100000
- **Reversing Method**
 - Description:
 - A position is chosen at random, and the bit next to that position is reversed to produce a mutated child.
 - Example:
 - Offspring Chromosome: 10101100
 - Mutation: Reverse the bit next to the third position.
 - Mutated Chromosome: 10001100

Convergence & Convergence Criteria in Genetic Algorithms

- Convergence in GA:
 - For any problem, if a Genetic Algorithm (GA) is correctly implemented, the population (set of solutions) evolves over successive generations with fitness values increasing toward the global optimum.
- Convergence:
 - The progression toward increasing uniformity. A population is said to have converged when 95% of the individuals (solutions) constituting the population have the same fitness value.
- Convergence (Termination) Criteria:
 - Each iteration should be tested with some convergence test.

- Various Convergence (Termination) Criteria:
 - **Objective Criteria**: A solution is found that satisfies the objective criteria.
 - **Fixed Number of Generations**: A predetermined number of generations is executed.
 - **Allocated Budget**: The allocated computational time or resources are exhausted.
 - **Fitness Threshold**: The highest-ranking solution's fitness is reaching or has reached a point such that successive iterations no longer produce better results.
 - **Manual Inspection**: Convergence is determined through manual inspection.
 - **Combination**: A combination of the above criteria is used.

Applications of Genetic Algorithms

- Optimization- GAs are most commonly used for optimization problems where we have to maximize or minimize a given objective function under a set of constraints.
- Neural Networks- GAs are used to train neural networks, especially recurrent neural networks.
- Image Processing- GAs are used for various digital image processing tasks.
- DNA Analysis- GAs have been used to determine the structure of DNA.
- Traveling Salesman Problem- GAs are used to solve the Traveling Salesman Problem (TSP).

- **Scheduling Applications**- GAs are used in solving various scheduling problems.
- **Machine Learning**- GAs are used in GBML (Genetic-Based Machine Learning).
- **Parametric Design of Aircraft**- GAs are used to design aircraft by varying parameters and evolving better solutions.
- **Robotics**- GAs are used to create learning robots which behave like humans.
- **Economics**- GAs are used to characterize various economic models like Cobweb models, game theory, equilibrium resolution, and asset pricing, etc.

GA Approach:

1. **Chromosome Representation:** A permutation of city indices.
 - Example: For 5 cities, a chromosome could be `[1, 3, 5, 2, 4]`.
2. **Fitness Function:** The total distance of the route represented by the chromosome.
 - Calculate the distance for each route permutation and use the inverse of this distance as the fitness value (shorter distances have higher fitness).
3. **Selection:** Roulette wheel selection or tournament selection to choose parent chromosomes based on fitness.
4. **Crossover:** Order crossover (OX) where segments of parents are exchanged while maintaining city sequence.
 - Example: Parent1 = `[1, 3, 5, 2, 4]`, Parent2 = `[2, 5, 1, 4, 3]`
 - Offspring could be `[1, 3, 5, 4, 2]` and `[2, 5, 1, 3, 4]`.
5. **Mutation:** Swap mutation where two cities in the chromosome are swapped to introduce variation.
 - Example: Mutate `[1, 3, 5, 2, 4]` to `[1, 4, 5, 2, 3]`.
6. **Replacement:** Replace the least fit chromosomes in the population with the new offspring.
7. **Termination:** Continue for a set number of generations or until no significant improvement is observed.

Step 1: Initialization

- Cities: Assume 5 cities with the following distances between them:

	1	2	3	4	5
1	0	10	8	9	7
2	10	0	6	4	5
3	8	6	0	7	3
4	9	4	7	0	6
5	7	5	3	6	0

- Initial Population: Let's assume an initial population of 4 chromosomes:

Chromosome 1 [1, 3, 5, 2, 4]
Chromosome 2 [2, 5, 1, 4, 3]
Chromosome 3 [3, 4, 2, 1, 5]
Chromosome 4 [5, 2, 4, 3, 1]

Step 2: Calculate Fitness

- **Chromosome 1:** [1, 3, 5, 2, 4]
 - Distance: $1 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 1 = 8 + 3 + 5 + 4 + 9 = 29$
 - Fitness: $\frac{1}{29} \approx 0.0345$
- **Chromosome 2:** [2, 5, 1, 4, 3]
 - Distance: $2 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 2 = 5 + 7 + 9 + 7 + 6 = 34$
 - Fitness: $\frac{1}{34} \approx 0.0294$
- **Chromosome 3:** [3, 4, 2, 1, 5]
 - Distance: $3 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 5 \rightarrow 3 = 7 + 4 + 10 + 7 + 3 = 31$
 - Fitness: $\frac{1}{31} \approx 0.0323$
- **Chromosome 4:** [5, 2, 4, 3, 1]
 - Distance: $5 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 \rightarrow 5 = 5 + 4 + 7 + 8 + 7 = 31$
 - Fitness: $\frac{1}{31} \approx 0.0323$

Step 3: Selection

- Using roulette wheel selection, we select parents based on their fitness.

Step 4: Crossover

- **Parents:** Let's select Chromosome 1 and Chromosome 2 for crossover.
- **Order Crossover (OX):**

- Parent 1: [1, 3, 5, 2, 4]
- Parent 2: [2, 5, 1, 4, 3]
- Offspring 1: [1, 3, 5, 4, 2]
- Offspring 2: [2, 5, 1, 3, 4]

Step 5: Mutation

- **Offspring 1 before mutation:** [1, 3, 5, 4, 2]
 - Swap mutation: Swap city 3 and city 5.
 - **Offspring 1 after mutation:** [1, 5, 3, 4, 2]
- **Offspring 2 before mutation:** [2, 5, 1, 3, 4]
 - Swap mutation: Swap city 1 and city 4.
 - **Offspring 2 after mutation:** [2, 5, 4, 3, 1]

Step 6: Replacement

- Replace the least fit chromosomes in the population with the new offspring.
- **New Population:**

Chromosome 1	[1, 3, 5, 2, 4]
Chromosome 2	[3, 4, 2, 1, 5]
Offspring 1	[1, 5, 3, 4, 2]
Offspring 2	[2, 5, 4, 3, 1]

Step 7: Termination

- Continue the process for a set number of generations or until no significant improvement is observed.

Genetic Algorithm: Advantages

- **Global Search Capability:**
 - **Example:** In optimizing the design of an aircraft wing, genetic algorithms can explore a wide range of shapes and configurations to find the most aerodynamic design.
- **No Need for Derivative Information:**
 - **Example:** For complex financial modeling where the relationship between variables is not clearly defined, genetic algorithms can optimize investment strategies without needing gradient information.
- **Adaptability:**
 - **Example:** In dynamic routing for delivery trucks, genetic algorithms can adapt to changes in traffic conditions or delivery requirements, continually finding optimal routes.
- **Versatility:**
 - **Example:** Genetic algorithms can be used for a variety of problems, such as scheduling tasks in a manufacturing plant, optimizing investment portfolios, or designing efficient network topologies.

Genetic Algorithm: Disadvantages

- **High Computational Cost:**
 - **Example:** In a complex scheduling problem for a large manufacturing plant, the computational resources required to evaluate many potential schedules can be substantial.
 - **Parameter Sensitivity:**
 - **Example:** When optimizing a neural network's architecture, the performance of the genetic algorithm can vary significantly based on the chosen mutation and crossover rates, requiring careful tuning.
 - **Premature Convergence:**
 - **Example:** In a genetic algorithm optimizing a chemical process, if the population quickly converges to a local optimum, it may miss better solutions, leading to suboptimal process parameters.
 - **Complexity in Implementation:**
 - **Example:** Implementing a genetic algorithm for optimizing the layout of electronic circuits can be complex, requiring detailed knowledge of genetic operations and careful parameter tuning.
- <http://www.knowledgegate.in/gate>

Genetic Programming (GP)

- Genetic Programming (GP) is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It is a specialized form of genetic algorithms (GAs) where the individuals in the population are computer programs.
- Example Workflow:
 - **Initialization**: Generate an initial population of random programs.
 - **Evaluation**: Assess the fitness of each program using the fitness function.
 - **Selection**: Select the fittest programs to act as parents for the next generation.
 - **Crossover**: Combine parts of two parent programs to create offspring programs.
 - **Mutation**: Apply random changes to some programs to introduce variability.
 - **Iteration**: Repeat the evaluation, selection, crossover, and mutation steps for many generations.

- Example Application:
 - Symbolic Regression: GP can be used to find a mathematical expression that best fits a set of data points. For instance, given a dataset of input-output pairs, GP can evolve an equation that models the relationship between inputs and outputs.