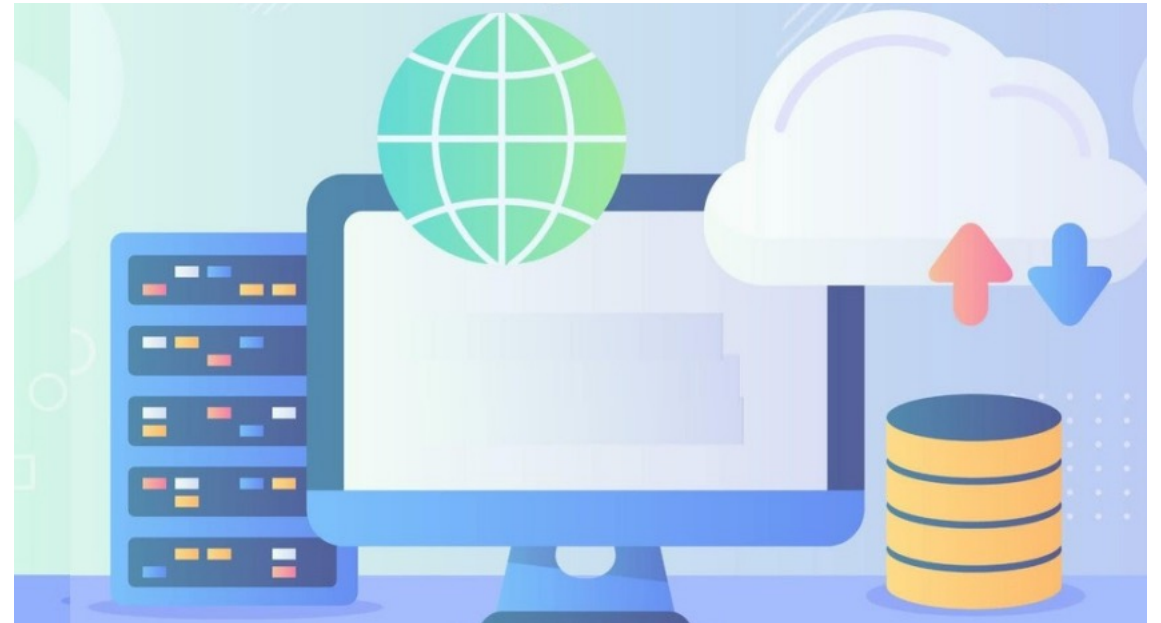
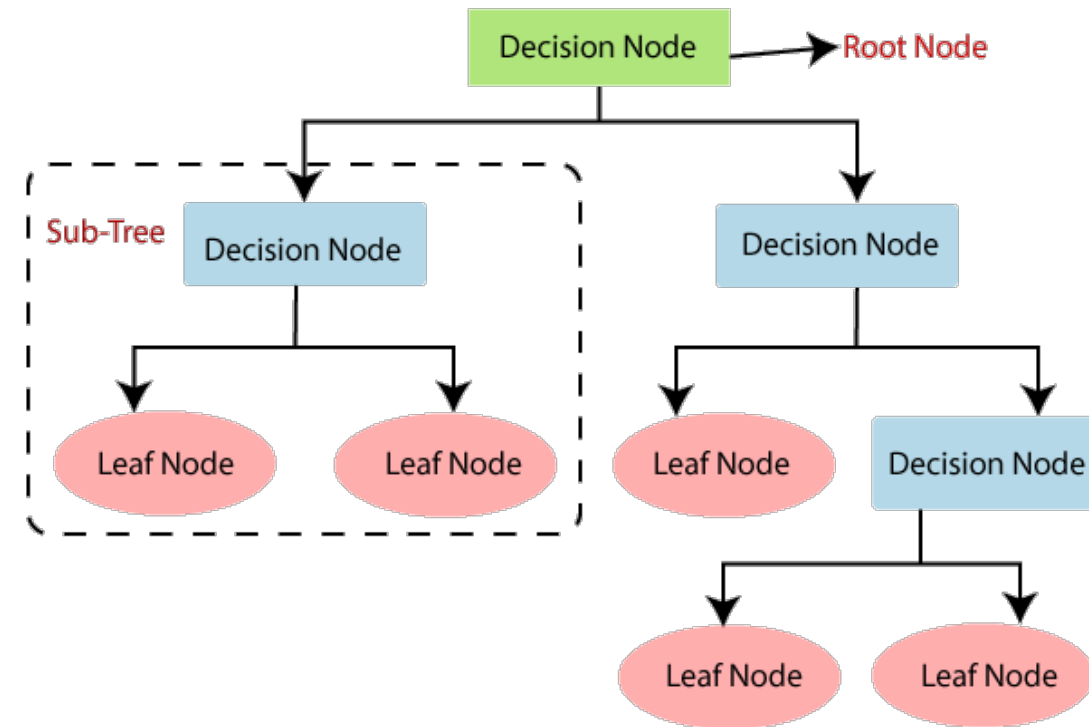


KNN AND DECISION TREE CLASSIFIER



Decision Tree Classifier – Introduction

- A decision tree is a fundamental tool in supervised learning, serving for both classification and regression purposes.
- It creates a tree-like structure where each internal node represents a decision based on an attribute, each branch signifies the outcome of that decision, and each leaf node holds a class label or regression value.
- The tree is formed by iteratively dividing the training data into subsets according to attribute values until certain stopping conditions, like reaching a maximum depth or requiring a minimum number of samples per node, are met.

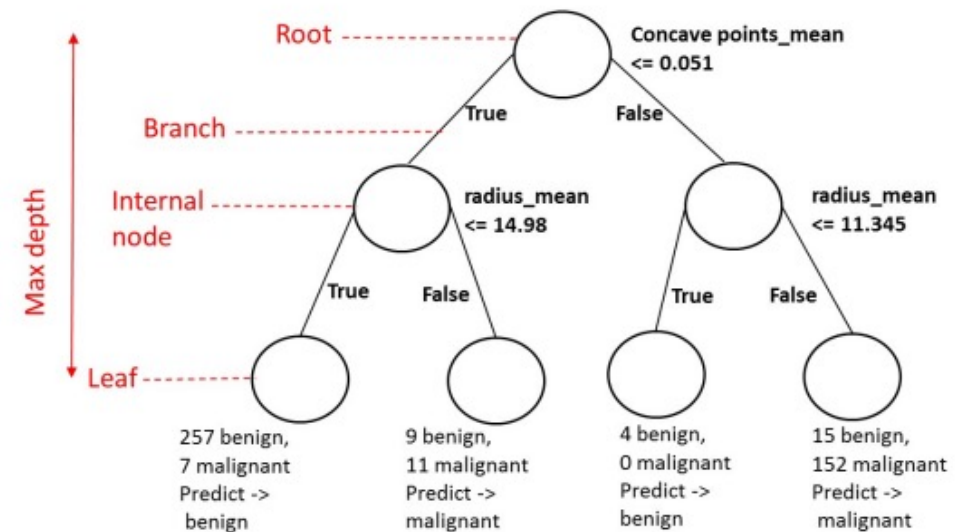


Decision Tree Classifier – Terminologies

- **Root Node:** It is the topmost node in the tree, which represents the complete dataset. It is the starting point of the decision-making process.
- **Decision/Internal Node:** A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
- **Leaf/Terminal Node:** A node without any child nodes that indicates a class label or a numerical value.
- **Splitting:** The process of splitting a node into two or more sub-nodes using a split criterion and a selected feature.
- **Branch/Sub-Tree:** A subsection of the decision tree starts at an internal node and ends at the leaf nodes.
- **Parent Node:** The node that divides into one or more child nodes.
- **Child Node:** The nodes that emerge when a parent node is split.
- **Impurity:** A measurement of the target variable's homogeneity in a subset of data. It refers to the degree of randomness or uncertainty in a set of examples. The **Gini index** and **entropy** are two commonly used impurity measurements in decision trees for classifications task

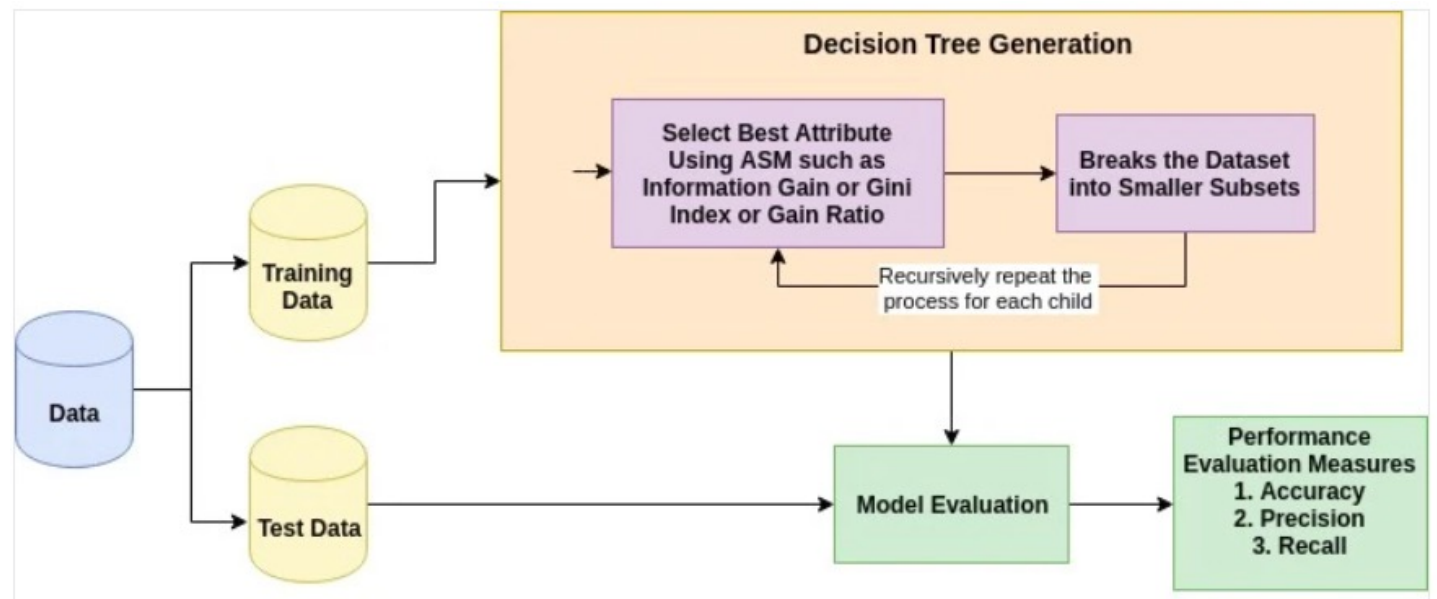
Decision Tree Classifier – Terminologies

- **Variance:** Variance measures how much the predicted and the target variables vary in different samples of a dataset. It is used for regression problems in decision trees. **Mean squared error, Mean Absolute Error, friedman_mse, or Half Poisson deviance** are used to measure the variance for the regression tasks in the decision tree.
- **Information Gain:** Information gain is a measure of the reduction in impurity achieved by splitting a dataset on a particular feature in a decision tree. The splitting criterion is determined by the feature that offers the greatest information gain, It is used to determine the most informative feature to split on at each node of the tree, with the goal of creating pure subsets
- **Pruning:** The process of removing branches from the tree that do not provide any additional information or lead to overfitting.



Decision Tree Classifier – Algorithm

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



Decision Tree Classifier – Attribute Selection Measure


- In decision trees, the attribute selection measure is a criterion used to evaluate the effectiveness of different attributes for splitting the data at each node of the tree.
- The primary goal is to select the attribute that best separates the data into homogeneous subsets with respect to the target variable (class label for classification tasks or target value for regression tasks).

1. Entropy:

- Entropy measures the uncertainty or randomness in a set of data. In decision trees, it is used to quantify the impurity of a node.
- The entropy of a node is calculated based on the proportion of instances belonging to each class.
- A node with low entropy indicates high purity, meaning it predominantly contains instances of a single class.

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5


$$\begin{aligned}\text{Entropy}(\text{PlayGolf}) &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

Decision Tree Classifier – Attribute Selection Measure

- **Important points related to Entropy:**
 - The entropy is 0 when the dataset is completely homogeneous, meaning that each instance belongs to the same class. It is the lowest entropy indicating no uncertainty in the dataset sample.
 - when the dataset is equally divided between multiple classes, the entropy is at its maximum value. Therefore, entropy is highest when the distribution of class labels is even, indicating maximum uncertainty in the dataset sample.
 - Entropy is used to evaluate the quality of a split. The goal of entropy is to select the attribute that minimizes the entropy of the resulting subsets, by splitting the dataset into more homogeneous subsets with respect to the class labels.
 - The highest information gain attribute is chosen as the splitting criterion (i.e., the reduction in entropy after splitting on that attribute), and the process is repeated recursively to build the decision tree.

Decision Tree Classifier – Attribute Selection Measure

2. Gini Impurity:

- Gini impurity, similar to entropy, measures the impurity of a node by calculating the probability of misclassifying an instance if it were randomly assigned a class label according to the class distribution in the node.
- A node with a lower Gini impurity value is considered more homogeneous.
- Gini Impurity is a score that evaluates how accurate a split is among the classified groups.
- The Gini Impurity evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes.

$$\begin{aligned} \text{Gini Impurity} &= 1 - \sum_{i=1}^K p_i^2 \\ &= 1 - \text{Gini Index} \end{aligned}$$

where K is the number of class labels,

p_i is the proportion of i^{th} class label

Decision Tree Classifier – Attribute Selection Measure

3. Information Gain:

- Information gain is the measure of the reduction in entropy or Gini impurity achieved by splitting the data on a particular attribute. It quantifies the amount of uncertainty removed about the class labels after the split.
- Attributes with higher information gain are preferred for splitting as they lead to more homogeneous child nodes.
- It is used in decision tree algorithms to determine the usefulness of a feature by partitioning the dataset into more homogeneous subsets with respect to the class labels or target variable.
- Information gain computes the difference between entropy before the split and average entropy after the split of the dataset based on given attribute values.
- $\text{Information Gain}(A) = \text{Entropy}(S) - \sum(|S_v| / |S|) * \text{Entropy}(S_v)$ Where:
 - A is the attribute
 - S is the current dataset
 - S_v is the subset of S for which attribute A has a value v
 - $|S_v|$ is the number of instances in S_v
 - $|S|$ is the number of instances in S

Decision Tree Classifier – ID3 Algorithm

- The ID3 (Iterative Dichotomiser 3) algorithm is a popular decision tree learning algorithm used for building decision trees from a given dataset.
- It was developed by J. Ross Quinlan and is based on the concept of information entropy and information gain.
- The ID3 algorithm follows a greedy approach and uses the top-down, recursive approach to construct a decision tree for the given dataset.
- The algorithm is called greedy because the highest values are always picked first and there is no backtracking.
- **Steps:**
 - Calculate the Information Gain of each feature.
 - Considering that all rows don't belong to the same class, split the dataset S into subsets using the feature for which the Information Gain is maximum.
 - Make a decision tree node using the feature with the maximum Information gain.
 - If all rows belong to the same class, make the current node as a leaf node with the class as its label.
 - Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Decision Tree Classifier – ID3 Algorithm

- Use ID3 Algorithm to construct a decision tree for the following dataset.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision Tree Classifier – Advantages

- 1. Interpretability:** Decision trees are easy to interpret and understand, as they provide a visual representation of the decision-making process, making them suitable for applications where interpretability is crucial.
- 2. Non-parametric:** Decision trees are non-parametric models, meaning they do not make any assumptions about the underlying data distribution, making them flexible and capable of capturing complex non-linear relationships.
- 3. Handling of various data types:** Decision trees can handle both numerical and categorical data, making them versatile for a wide range of problems.
- 4. Feature selection:** During the tree construction process, decision trees automatically perform feature selection, identifying the most relevant features for the task at hand.
- 5. Parallel processing:** Decision tree algorithms can be parallelized, allowing for efficient training on large datasets.

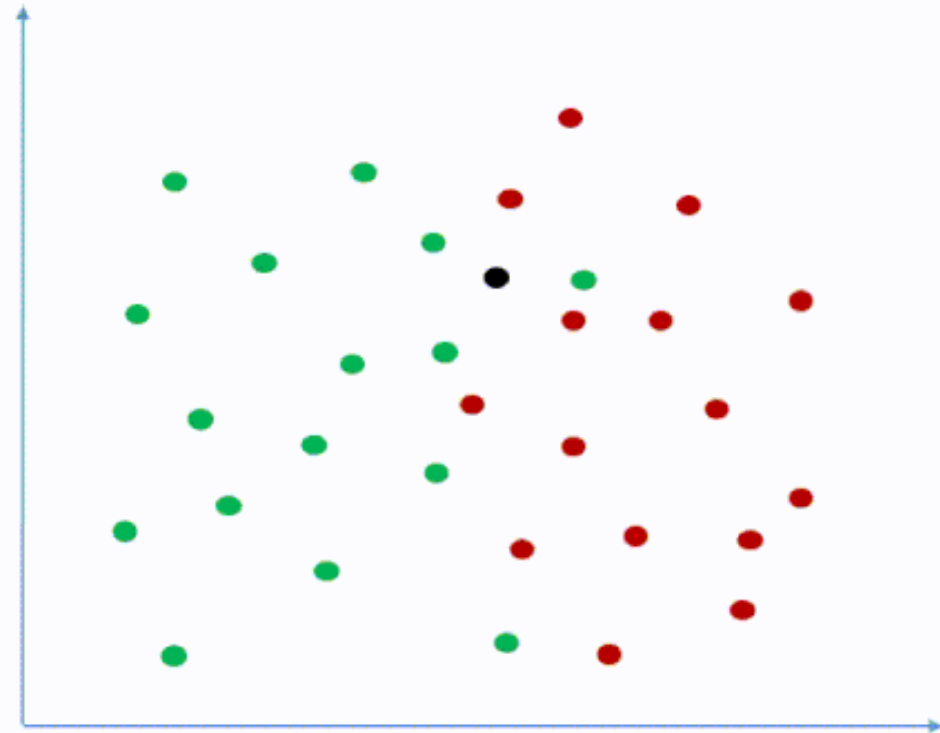
Decision Tree Classifier – Disadvantages

- 1. Overfitting:** Decision trees are prone to overfitting, especially when the tree grows too deep or when the dataset is noisy. Overfitting occurs when the model captures noise or outliers in the training data, leading to poor generalization performance on unseen data.
- 2. Instability:** Small variations in the training data can lead to significant changes in the structure of the decision tree, making them sensitive to noise.
- 3. High Variance:** Decision trees have high variance, meaning they can produce different trees for different training datasets, resulting in less stable predictions.
- 4. Bias Towards High Cardinality Attributes:** Decision trees tend to favor attributes with a large number of values (high cardinality) when selecting splitting criteria, potentially leading to biased trees.

Nearest Neighbor Classifier – Introduction

- The Nearest Neighbor algorithm, also known as the k-Nearest Neighbors (k-NN) algorithm, is indeed a supervised machine learning algorithm used for both classification and regression tasks.
- It was first introduced in 1951 by Evelyn Fix and Joseph Hodges in a technical report titled "Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties."
- In their original work, Fix and Hodges proposed the idea of classifying an unknown instance based on the most frequently occurring class among its nearest neighbors in the training data.
- Later, in 1967, Thomas Cover expanded on this concept and formalized the theoretical framework for the k-NN algorithm in his paper "Estimation by the Nearest Neighbor Rule."

K-Nearest Neighbors Classification



Nearest Neighbor Classifier – Properties

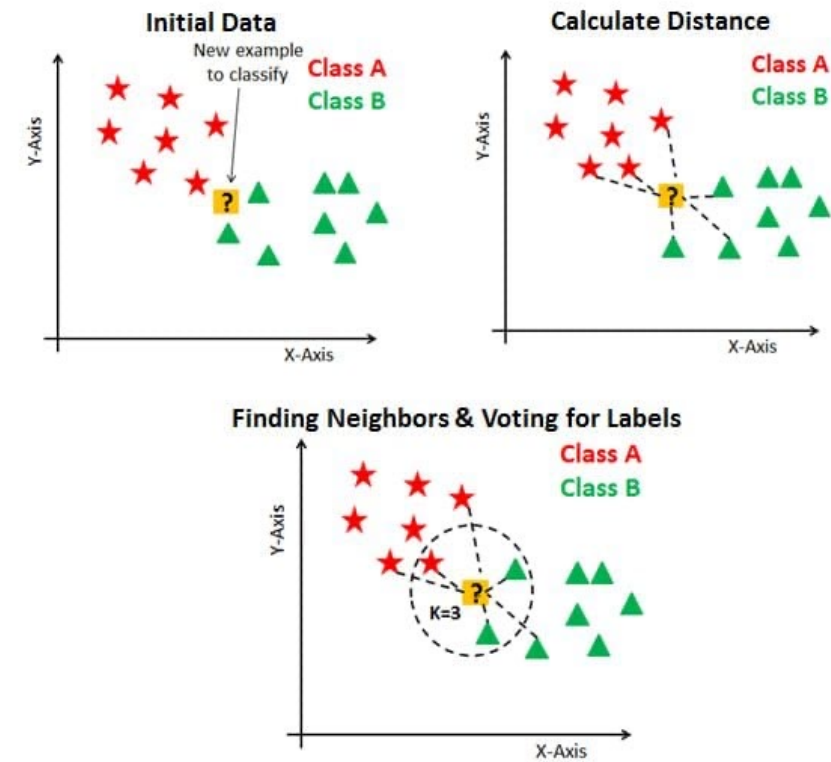
- 1. Supervised Learning Technique:** K-NN is a supervised learning algorithm, meaning it requires labeled training data to make predictions. It classifies new data points based on their similarity to existing labeled data points.
- 2. Similarity-Based Classification:** K-NN assumes that data points with similar features belong to the same class. It classifies a new data point by finding the k nearest neighbors in the feature space and assigning the majority class label among those neighbors to the new data point.
- 3. Usage for Classification and Regression:** While K-NN is commonly used for classification tasks, it can also be applied to regression problems by predicting the numerical value of a new data point based on the average or weighted average of the values of its nearest neighbors.
- 4. Non-Parametric Algorithm:** K-NN is a non-parametric algorithm because it does not make any assumptions about the underlying distribution of the data. Instead, it relies on the entire dataset during the prediction phase.
- 5. Lazy Learner Algorithm:** K-NN is sometimes referred to as a lazy learner algorithm because it postpones the learning process until the prediction phase. It stores the entire training dataset and performs computations only when making predictions for new data points.

Nearest Neighbor Classifier – Properties

6. **Storage of Training Data:** During the training phase, K-NN simply stores the entire training dataset without any further processing. This makes the training phase fast and efficient, as no model fitting or parameter estimation is required.
7. **Classification of New Data:** When presented with new data during the prediction phase, K-NN calculates the distance between the new data point and all existing data points in the training set. It then selects the k nearest neighbors and assigns the majority class label among those neighbors to the new data point.
 - The kNN algorithm can be considered a voting system, where the majority class label determines the class label of a new data point among its nearest ' k ' (where k is an integer) neighbors in the feature space.
 - Imagine a small village with a few hundred residents, and you must decide which political party you should vote for. To do this, you might go to your nearest neighbors and ask which political party they support.
 - If the majority of your ' k ' nearest neighbors support party A, then you would most likely also vote for party A. This is similar to how the kNN algorithm works, where the majority class label determines the class label of a new data point among its k nearest neighbors.

Nearest Neighbor Classifier – ‘K’ in KNN

- The ‘K’ in KNN is a parameter that refers to the number of nearest neighbors.
- K is a positive integer and is typically small in value and is recommended to be an odd number.
- The example below shows 3 graphs. The first, the ‘Initial Data’ is a graph where data points are plotted and clustered into classes, and a new example to classify is present.
- In the ‘Calculate Distance’ graph, the distance from the new example data point to the closest trained data points is calculated.
- However, this still does not categorize the new example data point. Therefore, using k-value, essentially created a neighborhood where we can classify the new example data point.
- We would say that if $k=3$ and the new data point will belong to Class B as there are more trained Class B data points with similar characteristics to the new data point in comparison to Class A.



K-Nearest Neighbor Classifier – Algorithm

1. Load the Data: Load the training data that consists of feature vectors and their corresponding labels (for classification) or target values (for regression).
2. Choose the Value of k: Select the value of 'k', which represents the number of nearest neighbors to consider for classification or regression.
3. Choose a Distance Metric: Select a distance metric to calculate the distance between data points. Common choices are Euclidean distance, Manhattan distance, or Minkowski distance.
4. For Each New Data Point:
 - a. Calculate the distance between the new data point and all the training data points using the chosen distance metric.
 - b. Sort the distances in ascending order.
 - c. Select the 'k' nearest neighbors based on the sorted distances.
5. For Classification:
 - a. Among the 'k' nearest neighbors, count the number of instances belonging to each class.
 - b. Assign the new data point to the class with the highest frequency among the 'k' nearest neighbors.

K-Nearest Neighbor Classifier – Algorithm

6. Repeat Steps 4-5 for each new data point that needs to be classified or have its value predicted.
7. Evaluate the Model: If the data is labeled, evaluate the performance of the KNN model using appropriate metrics such as accuracy (for classification) or mean squared error (for regression).
8. Optimize the Value of k (Optional): If the performance is not satisfactory, try different values of ' k ' and choose the one that yields the best performance on a validation set or using techniques like cross-validation.