**K-Nearest Neighbors (KNN)**

 K-Nearest Neighbors is a simple yet effective machine learning algorithm used for both classification and regression tasks. In KNN, the input consists of the k closest training examples in the feature space. The output depends on whether KNN is used for classification or regression. KNN does not have a traditional objective function that is optimized during training, as it does not explicitly learn a model. Instead, it relies on instance-based learning, where it uses the training data itself during the testing phase to make predictions.

However, there are some considerations that can be made when using KNN:

***Distance Metrics:*** In the context of KNN, the choice of distance metric plays a crucial role. The most commonly used distance metrics are Euclidean distance and Manhattan distance. Other distance metrics such as Minkowski distance, Hamming distance, and cosine similarity can also be used depending on the nature of the data. ***Choosing the Value of K:*** The choice of the parameter K, the number of nearest neighbors, is crucial in KNN. A small value of K can lead to noisy classification, while a large value can smooth out the decision boundary.

***Weighting Schemes:*** In some variations of KNN, the contribution of the neighbors to the classification or regression can be weighted based on their distance from the query point. This helps give more importance to the closer neighbors.

Some commonly used distance metrics for image data in the context of KNN include: ***Euclidean Distance:*** This is a straightforward distance metric that calculates the straight-line distance between two points in the feature space. It is commonly used when dealing with numerical or continuous image features.



***Manhattan Distance:*** Also known as the city block distance, this metric calculates the sum of the absolute differences between the coordinates of two points. It is useful when dealing with high-dimensional data or data with sparse features. Cosine Similarity: Rather than measuring the distance between points, cosine similarity calculates the cosine of the angle between two non-zero vectors. It is particularly useful when dealing with high-dimensional data and is commonly used in tasks such as image retrieval and clustering.



**Decision Tree**

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.



* In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.
* It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

**How to choose the best attribute at each node**

While there are multiple ways to select the best attribute at each node, two methods, information gain and Gini impurity, act as popular splitting criterion for decision tree models. They help to evaluate the quality of each test condition and how well it will be able to classify samples into a class.

***Entropy and Information Gain***

It’s difficult to explain information gain without first discussing entropy. Entropy is a concept that stems from information theory, which measures the impurity of the sample values. It is defined with by the following formula, where:

 



Entropy values can fall between 0 and 1. If all samples in data set, S, belong to one class, then entropy will equal zero. If half of the samples are classified as one class and the other half are in another class, entropy will be at its highest at 1. In order to select the best feature to split on and find the optimal decision tree, the attribute with the smallest amount of entropy should be used. Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it’s doing the best job at classifying the training data according to its target classification. Information gain is usually represented with the following formula, where:



