[Milena Afeworki](https://milena-pa.medium.com/?source=post_page-----d6110e08bfea-----------------------------------)

https://milena-pa.medium.com/a-comparison-of-machine-learning-algorithms-knn-vs-decision-trees-d6110e08bfea

There are plenty of machine learning algorithms out there. There is no single machine algorithm that works best for all types of scenarios. Some of the factors that affect our choice of picking up a machine learning algorithm include:

1. Size of the training data
2. Accuracy and/or interpretability
3. Training time
4. Linearity
5. Number of features
6. Supervised or unsupervised

Hence, we should always be cautious in picking up an algorithm. In this post I am going to discuss the difference between two commonly used machine learning algorithms, namely; Decision Trees and K-nearest Neighbor and will also discuss the above factors. First, let’s briefly introduce how these algorithms work, and then compare them to list out their pros and cons.

**Decision Tree:**

Decision trees are non-parametric supervised ML methods used for classification and regression. It is a structure similar to a flowchart in which decisions and decision-making processes are visually and explicitly represented. In the figure below, depicting passenger survival on the Titanic, you can see how the decision tree algorithm made a decision at each node. That is the beauty of decision trees, you can easily understand and visualize the decision-making process.

Diagram

Description automatically generated

***K-Nearest Neighbor (KNN)***

K-nearest neighbor is a simple non-parametric, supervised machine learning algorithm. In KNN algorithm, the k is a user-defined constant. The following example will shed light on how KNNworks.

A picture containing diagram

Description automatically generated

Let’s say we are solving a classification problem and we want to classify which class the green dot belongs to. In this case, we see two concentric circles in the figure with blue squares and red triangles. Our goal here is to classify the green dot into either blue squares or red triangles. If we take k = 3, then we will consider the inner circle with a solid line and calculate the probability of each class. The probability of red is ⅔ therefore our green dot will belong to the red triangles. Similarly, if we consider the outer dotted circle, the probability of blue square is ⅗, hence, we would classify the green dot as a blue square. During classification, the model will look through all the training data and give out the result. In the above example, all the green squares and red triangles are training sets and the green dot is our test sample. During the prediction, the model goes through all the data and finds out the closest class to the test sample.

**Comparison of Decision Trees and KNN**

In the following list below, we compare Decision Trees to KNNs.

\* Both are non-parametric. This means that the data distribution cannot be defined in a few parameters. In other words, Decision trees and KNN’s don’t have an assumption on the distribution of the data.

\* Both can be used for regression and classification problems.

\* Decision tree supports automatic feature interaction, whereas KNN doesn’t.

\* Decision trees can be faster, however, KNN tends to be slower with large datasets because it scans the whole dataset to predict as it doesn’t generalize the data in advance.

We compared Decision Trees to KNNs and we can see that they have some advantages and limitations. Now, let’s explore the pros and cons of these two algorithms. We will start off with decision trees and then we will see KNN.

**Decision Trees:**

**Advantages**:

\* Decision trees are effective in capturing *non-linear relationships* which can be difficult to achieve with other algorithms like Support Vector Machine and Linear Regression.

\* *Easy to explain to people:* This is a great aspect of decision trees. The outputs are easy to read without requiring statistical knowledge or complex concepts.

\* Some people believe decision trees more closely mirror human decision-making than others like regression and classification approaches.

\* Trees can be displayed graphically and can be easily interpreted by non-experts.

\* Decision trees can easily handle qualitative (categorical) features without the need to create dummy variables.

**Disadvantages:**

\* don’t have the same level of predictive accuracy as some of other regression and classification approaches

\* trees can be non-robust. Eg. small change in the data can cause a large change in the final estimated tree

\* As the tree grows in size, it becomes prone to overfitting and requires pruning

**KNN:**

**Advantages:**

**\****Simple and intuitive:* Similar to decision trees it is simple and easy to explain to laypeople.

\* *Non-parametric*, therefore, it doesn’t have any assumptions on the data distribution

\* *No training step*: KNN is more of an exception to the general machine learning workflow. It doesn’t have a training/validation/test set. The model created with KNN is available in a labeled dataset, placed in metric space. Say, if you want to classify any object, the model has to read through all the data and compare the distances of the closest objects.

\**Easy to implement for multi-class problems:*Compared to other algorithms, it is very easy to predict multiclass problems. Just supply the ‘k’ a value that is equivalent to the number of classes and you are good to go.

\* *Few hyperparameters:* When working with K-NN, you just need to provide two parameters, k (the numbers of neighbors to consider) and the choice of Distance Function (e.g. Euclidean, Manhattan distance).

\* *Used for classification and regression:* It can be used for classification and regression

\* *Instance-based learning (lazy learning):* You don’t need to fit a model in advance, just provide the data point and it will give you the prediction.

**Disadvantages:**

\* Slow with a larger dataset. If it is going to classify a new sample, it will have to read the whole dataset, hence, it becomes very slow as the dataset increases.

*\* Curse of dimensionality:* KNN is more appropriate to use when you have a small number of inputs. If the number of variables grows, the KNN algorithm will have a hard time predicting the output of a new data point.

*\* Feature inputs need to be scaled:*It is a *must*that the features should be scaled. KNN uses distance criteria, like Euclidean or Manhattan distances, therefore, it is very important that all the features have the same scale.

*\* Outlier sensitivity:* KNN is very sensitive to outliers. Since it is an instance-based algorithm based on the distance criteria, if we have some outliers in the data, it is liable to create a biased outcome.

*\* Missing Value not treated:* It is not capable of treating or dealing with missing values

*\* Class imbalance can be an issue:*If we have an imbalanced class data, the algorithm might wrongly pick the majority class.