01. Decision Tree Algorithm In Python
02. K Nearest Neighbor Algorithm In Python

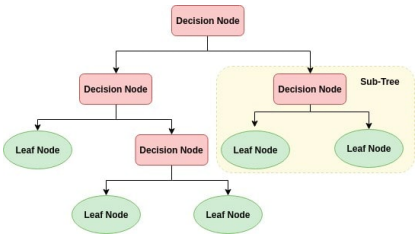
CSE-0408 Summer 2021

Ashraf uddin Mamun [UG02-44-17-044]
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
mamun336929@gmail.com

***Abstract*—Decision Tree Algorithm In Python:**

I. Overview :

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

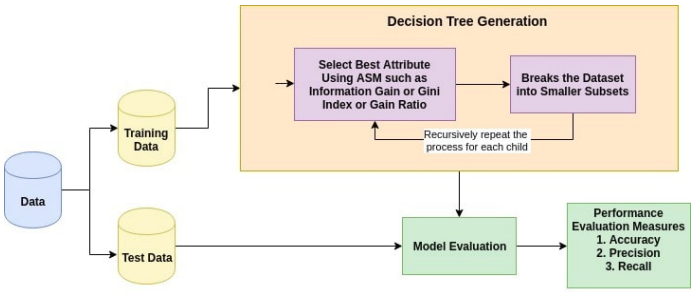
II. Pros and Cons of Decision Tree:

Pros: 01. Decision trees are easy to interpret and visualize. 02. It can easily capture Non-linear patterns. 03. It requires fewer data pre-processing from the user, for example, there is no need to normalize columns. 04. It can be used for feature engineering such as predicting missing values, suitable for variable selection.05. The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm.

Cons: 01. Sensitive to noisy data. It can overfit noisy data. 02. The small variation (or variance) in data can result it is different decision tree. This can be reduced by bagging and boosting algorithms. 03. Decision trees are biased with imbalance dataset, so it is recommended that balance out the dataset before creating the decision tree.

III. Decision Tree algorithm:

The basic idea behind any decision tree algorithm is as follows:



01. Select the best attribute using Attribute Selection Measures(ASM) to split the records. 02. Make that attribute a decision node and breaks the dataset into smaller subsets. 03. Starts tree building by repeating this process recursively for each child until one of the condition will match:

*All the tuples belong to the same attribute value.

*There are no more remaining attributes.

*There are no more instances.

IV. Conclusion :

Decision Trees are easy to interpret, don't require any normalization, and can be applied to both regression and classification problems. Unfortunately, Decision Trees are seldom used in practice because they don't generalize well. Stay tuned for the next article where we'll cover Random Forest, a method of combining multiple Decision Trees to achieve better accuracy.

***Abstract*—K Nearest Neighbor Algorithm In Python:**

V. Overview :

K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based off of the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little to no training phase. Therefore, we can immediately classify new data points as they present themselves.

VI. Some pros and cons of KNN :

Pros:

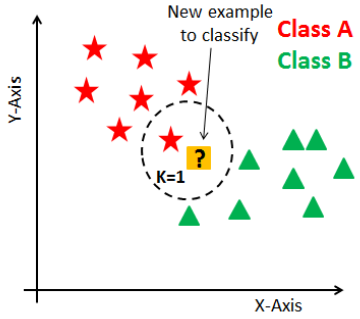
The training phase of K-nearest neighbor classification is much faster compared to other classification algorithms. There is no need to train a model for generalization, That is why KNN is known as the simple and instance-based learning algorithm. KNN can be useful in case of nonlinear data. It can be used with the regression problem. Output value for the object is computed by the average of k closest neighbors value.

Cons:

The testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory. It requires large memory for storing the entire training dataset for prediction. KNN requires scaling of data because KNN uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weight more than features with low magnitudes. KNN also not suitable for large dimensional data.

VII. How does the KNN algorithm work?

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.



Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

01. Calculate distance

02. Find closest neighbors

03. Vote for labels

VIII. Conclusion :

The K Nearest Neighbors algorithm doesn't require any additional training when new data becomes available. Rather it determines the K closest points according to some distance metric (the samples must reside in memory). Then, it looks at the target label for each of the neighbors and places the new found data point into the same category as the majority. Given that KNN computes distance, it's imperative that we scale our data. In addition, since KNN disregards the underlying features, it's our responsibility to filter out any features that are deemed irrelevant.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this course.