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

CSE-0408 Summer 2021

Mahmudul Islam [UG02-47-17-004]
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhanmondi, Dhaka, Bangladesh

In this report we discuss about Decision Trees Algorithm In Python

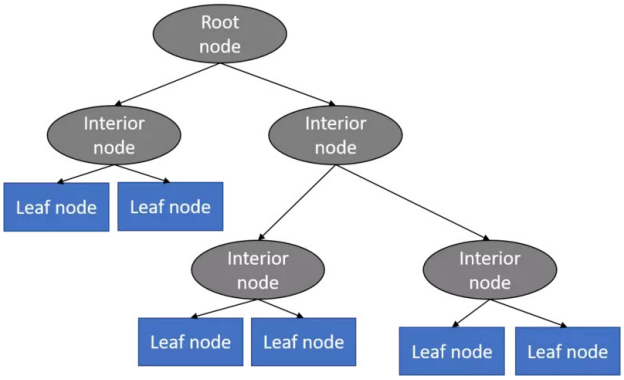
***Abstract*—A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis 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.**

I. Overview :

Decision trees are supervised learning algorithms used for both, classification and regression tasks where we will concentrate on classification in this first part of our decision tree tutorial. Decision trees are assigned to the information based learning algorithms which use different measures of information gain for learning. We can use decision trees for issues where we have continuous but also categorical input and target features. The main idea of decision trees is to find those descriptive features which contain the most "information" regarding the target feature and then split the dataset along the values of these features such that the target feature values for the resulting sub datasets are as pure as possible

The descriptive feature which leaves the target feature most purely is said to be the most informative one. This process of finding the "most informative" feature is done until we accomplish a stopping criteria where we then finally end up in so called leaf nodes. The leaf nodes contain the predictions we will make for new query instances presented to our trained model. This is possible since the model has kind of learned the underlying structure of the training data and hence can, given some assumptions, make predictions about the target feature value (class) of unseen query instances.

A decision tree mainly contains of a root node, interior nodes, and leaf nodes which are then connected by branches



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. LITERATURE REVIEW

A Decision Tree is a useful and popular classification technique that inductively learns a model from a given set of data. One reason for its popularity stems from the availability of existing algorithms that can be used to build decision trees, such as CART (Breiman et al., 1984), ID3 (Quinlan, 1986), and C4.5 (Quinlan, 1993). These algorithms all learn decision trees from a supplied set of training data, but do so in slightly different ways. As discussed in the introduction, a classifier is built by analyzing training data.

III. Advantages of Using a Decision Tree

1. Clear Visualization: The algorithm is simple to understand, interpret and visualize as the idea is mostly used in our daily lives. Output of a Decision Tree can be easily interpreted by humans.
2. Simple and easy to understand: Decision Tree looks like simple if-else statements which are very easy to understand.

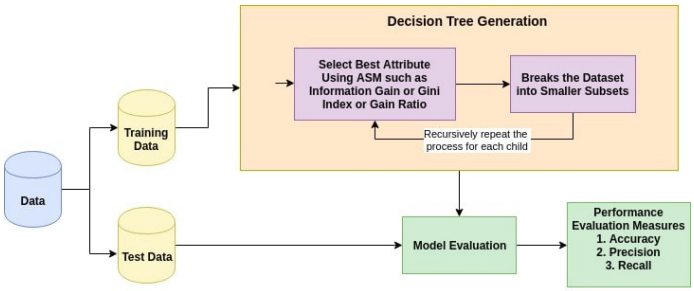
3. Decision Tree can be used for both classification and regression problems.
4. Decision Tree can handle both continuous and categorical variables.
5. No feature scaling required: No feature scaling (standardization and normalization) required in case of Decision Tree as it uses rule based approach instead of distance calculation.
6. Handles non-linear parameters efficiently: Non linear parameters don't affect the performance of a Decision Tree unlike curve based algorithms. So, if there is high non-linearity between the independent variables, Decision Trees may outperform as compared to other curve based algorithms.
7. Decision Tree can automatically handle missing values.
8. Decision Tree is usually robust to outliers and can handle them automatically.

IV. Disadvantages of Decision Tree

1. Overfitting: This is the main problem of the Decision Tree. It generally leads to overfitting of the data which ultimately leads to wrong predictions. In order to fit the data (even noisy data), it keeps generating new nodes and ultimately the tree becomes too complex to interpret. In this way, it loses its generalization capabilities. It performs very well on the trained data but starts making a lot of mistakes on the unseen data.
2. High variance: As mentioned in point 1, Decision Tree generally leads to the overfitting of data. Due to the overfitting, there are very high chances of high variance in the output which leads to many errors in the final estimation and shows high inaccuracy in the results. In order to achieve zero bias (overfitting), it leads to high variance.
3. Unstable: Adding a new data point can lead to re-generation of the overall tree and all nodes need to be recalculated and recreated.
4. Affected by noise: Little bit of noise can make it unstable which leads to wrong predictions.
5. Not suitable for large datasets: If data size is large, then one single tree may grow complex and lead to overfitting. So in this case, we should use Random Forest instead of a single Decision Tree.

V. 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.

VI. How the Decision Tree Algorithm Works

To understand these advantages more clearly, let us discuss what the basic idea behind the decision tree algorithm is.
1. Choose the best or most appropriate attribute using Attribute Selection Measures(ASM) in order to divide the records
2. Transform that attribute to become a decision node and divide the dataset to create smaller subsets.
3.Start to create trees by repeating the process recursively for each child. When it matches one of the conditions below, the process shall end: All the tuples are contained in the same attribute value.
* No more attributes remain.
* No more instances.

VII. 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.

In this report we discuss about K Nearest Neighbor Algorithm In Python:

***Abstract*—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.**

VIII. Overview :

K nearest neighbors or KNN Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction. kNN is often used in search applications where you are looking for similar items, like find items similar to this one.

IX. Where to use KNN

KNN can be used in both regression and classification predictive problems. However, when it comes to industrial problems, it's mostly used in classification since it fairs across all parameters evaluated when determining the usability of a technique

1.Prediction Power
2.Calculation Time
3.Ease to Interpret the Output

KNN algorithm fairs across all parameters of considerations. But mostly, it is used due to its ease of interpretation and low calculation time.

X. Summary of KNN Algorithm

1. K is a positive integer
2. With a new sample, you have to specify K
3. K is selected from database closest to the new sample
4. KNN doesn't learn any model
5. KNN makes predictions using the similarity between an input sample and each training instance.

XI. Advantages of KNN

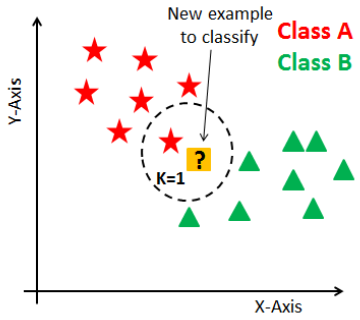
1. Quick calculation time
2. Simple algorithm – to interpret
3.Versatile – useful for regression and classification
4.High accuracy – you do not need to compare with better-supervised learning models

XII. Disadvantages of KNN

1.Accuracy depends on the quality of the data
2. With large data, the prediction stage might be slow
3.Sensitive to the scale of the data and irrelevant features

XIII. 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

XIV. Conclusion :

KNN is a simple yet powerful classification algorithm. It requires no training for making predictions, which is typically one of the most difficult parts of a machine learning algorithm. The KNN algorithm have been widely used to find document similarity and pattern recognition. It has also been employed for developing recommender systems and for dimensionality reduction and pre-processing steps for computer vision, particularly face recognition tasks.

Acknowledgment

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