# **Decision Tree Using Sci-Kit Learn**

Decision Tree is a type of supervised learning algorithms family which can perform both regression and classification problems. The purpose of employing Decision Trees is to build a training model that can be used to forecast the target variable's class or value by learning basic decision rules from prior data (training data).

When utilizing this algorithm to predict a record's class label, we start at the top of the tree. The root & record's attributes are compared. Based on the this, we follow the branch that corresponds to that value and then go to the next node.

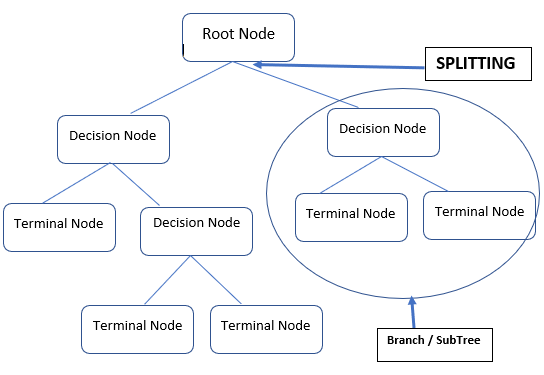
# **Types of Decision Trees**

There are 2 types of Decision Trees based on the target variable :

* **Categorical Variable:** Where target(y) variable is categorical.
* **Continuous Variable:** Where target(y) variable is continuous.

# **Components of Decision Trees**

* **Root node:** Symbolizes the total sample, which is then separated into two or more homogeneous groups.
* **Parent & Child Nodes:** A parent node of sub-nodes is a node that is divided into sub-nodes, whilst sub-nodes are the children of a parent node.
* **Decision Node:** Formed when a subnode splits into more subnodes.
* **Splitting:** Process of dividing a node into two/more subnodes.
* **Pruning:** It is the process of eliminating subnodes from a decision node.
* **Terminal / Leaf Nodes:** These are the nodes that do not split.
* **Sub-Tree / Branch:** A sub-tree/branch is a portion of the tree.



# **Assumptions of Decision Tree**

There are a few assumptions of Decision Trees as follows:

* At first, complete training dataset is regarded as the root.
* Basis attribute values, records are dispersed recursively.
* Using some statistical approaches, such as those listed below, it is possible to place attributes as the tree's root or internal node.

# **Let’s Address which Attribute(Feature) to be select as Root Node**

Choosing which attribute to insert at the root / at different levels of decision tree as internal nodes is a complex step since the dataset contains multiple features (variables). The problem cannot be solved by selecting any node at random as the root, it may end up with low accuracy & poor results.

This was solved by utilizing an algorithm such as Gini index, Information Gain, etc. Every attribute's value will be calculated using these algorithms. The values are sorted, and characteristics are ordered in tree, with the attribute having the highest value at the top (in the case of information gain).

# Approach to make decision tree

While making decision tree, at each node of tree we ask different type of questions. Based on the asked question we will calculate the information gain corresponding to it.

## Information Gain

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value **measures how much information a feature gives us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.**

## Hyper parameters

**max\_depth: int or None, optional (default=None)**

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

**min\_samples\_split: int, float, optional (default=2)**

The minimum number of samples required to split an internal node

**min\_samples\_leaf: int, float, optional (default=1)**

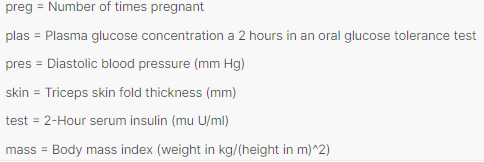
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

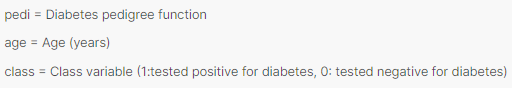
# **Let’s Build a simple Decision Tree (Classification) Model using Sci-kit Learn**

We’ll using a dataset from Kaggle – [Diabetes.](https://www.kaggle.com/uciml/pima-indians-diabetes-database)

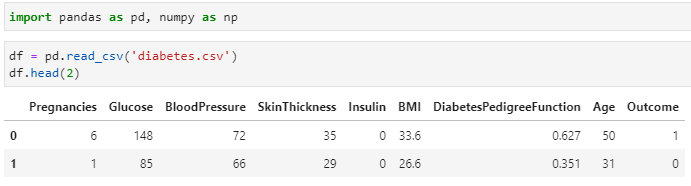
Download the csv files and load it into the jupyter environment.

Data Dictionary:





1. Let’s import the data :



import pandas as pd, numpy as np

df = pd.read\_csv('diabetes.csv')

df.head(2)

1. Feature Selection :



feature = ['Pregnancies', 'Insulin', 'BMI', 'Age','Glucose','BloodPressure','SkinThickness','Insulin']

X = df[feature] # ALl\_Features

y = df.Outcome # Target

1. Let’s split the data :

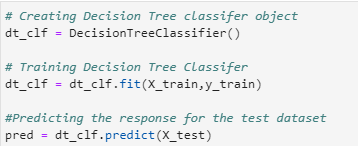


from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=1) # 75% training & 25% test

1. Let’s build a very simple intuitive Decision Tree Model :





from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

# Creating Decision Tree classifer object

dt\_clf = DecisionTreeClassifier()

# Training Decision Tree Classifer

dt\_clf = dt\_clf.fit(X\_train,y\_train)

#Predicting the response for the test dataset

pred = dt\_clf.predict(X\_test)

1. Let’s evaluate the decision tree classifier :

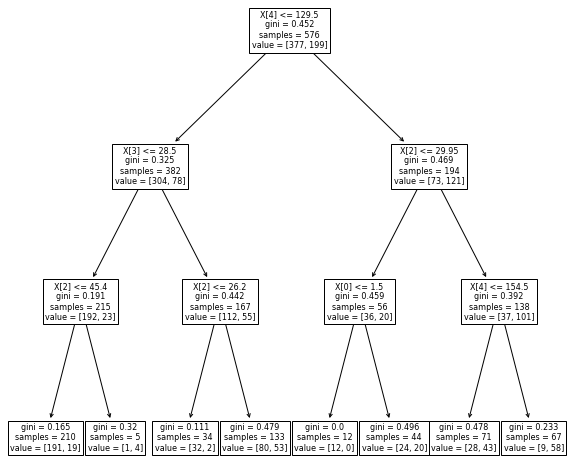


print("Accuracy:",metrics.accuracy\_score(y\_test, pred))

We have achieved the accuracy is 70% that can be improved by tuning some parameters.

1. Let’s visualize the decision tree:

First lets fix the depth of decision tree classifier.



# Creating Decision Tree classifer object

dt\_clf = DecisionTreeClassifier(max\_depth=3)

from sklearn import tree

import matplotlib.pyplot as plt

plt.figure(figsize=(10,10)) # set plot size (denoted in inches)

tree.plot\_tree(dt\_clf, fontsize=8)

plt.show()

# **Pros & Cons of Decision Trees :**

## Pros :

* Decision trees are simple to understand.
* It takes the same approach to decision-making that humans do in general.
* The visualizations of Decision Tree model can make it easier to understand.
* They are simple to understand and follow a pattern that is akin to human thought. In other words, it can be described as a set of questions / business rules.
* Prediction is a quick process. It's a series of operations that you perform until you reach a leaf node.
* Can be modified to deal with missing data without the need for data imputing

## Cons :

* In Decision Tree, there is a high risk of overfitting.
* In comparison to other machine learning techniques, it has a low prediction accuracy.
* In a decision tree with categorical variables, information gain leads to a biased response towards attributes with more categories.
* When there are a lot of class labels, calculations can get complicated.

# **Summary**

Decision trees are easy to comprehend and use, and they work well with large datasets. There are three main aspects to decision trees: decision nodes, chance nodes (which denotes probability), and end nodes (denoting conclusion). Decision trees algorithm can be used to with large datasets, and they can be pruned to avoid overfitting if needed.

Despite their many advantages, decision trees are not appropriate for all forms of data, such as datasets with imbalances or continuous variables.

## **References**

* [Jupyter Book Online](https://jupyter.org/try)
* [Kaggle - Diabetes dataset](https://www.kaggle.com/uciml/pima-indians-diabetes-database)
* Official documentation of [Decision Tree](https://scikit-learn.org/stable/modules/tree.html)