**Decision Tree**

* Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.
* Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.
* We can represent any boolean function on discrete attributes using the decision tree.

**Implementation in Python**

Anaconda Python 3 Package

**Dataset**

<https://www.kaggle.com/uciml/pima-indians-diabetes-database>

**Importing Required Libraries**

Let's first load the required libraries.

# Load libraries

import pandas as pd

# Import Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

# Import train\_test\_split function

from sklearn.model\_selection import train\_test\_split from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

### Loading Data

Using sklearn package load the required Pima Indian Diabetes dataset using pandas' read CSV function.

col\_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

# load dataset

pima= pd.read\_csv("pima-indians-diabetes.csv",header=None, names=col\_names)

pima.head()

### Feature Selection

Here, we need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

#split dataset in features and target variable

feature\_cols=['pregnant','insulin', 'bmi', 'age','glucose','bp','pedigree']

X = pima[feature\_cols] # Features

y = pima.label # Target variable

### Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. Let's split the dataset by using function train\_test\_split(). You need to pass 3 parameters features, target, and test\_set size.

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1) # 70% training and 30% test

### Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer

clf = clf.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

### Evaluating Model

Let's estimate, how accurately the classifier or model can predict the type of cultivars. Accuracy can be computed by comparing actual test set values and predicted values.

Model Accuracy, how often is the classifier correct?

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