**DECISION TREES.**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems.

It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

**TERMS.**

**Root Node:** The initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.

**Decision Nodes:** Nodes resulting from the splitting of root nodes are known as decision nodes. These nodes represent intermediate decisions or conditions within the tree.

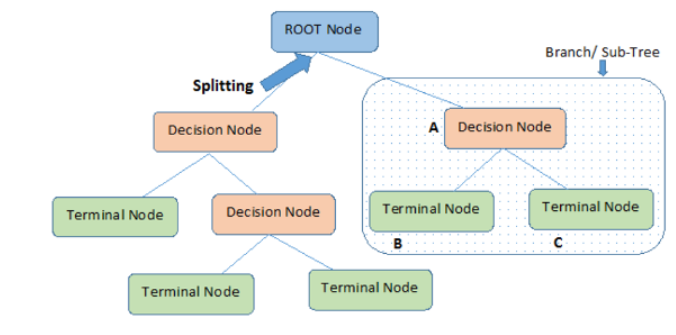
**Leaf Nodes:** Nodes where further splitting is not possible, often indicating the final classification or outcome. Leaf nodes are also referred to as terminal nodes.

**Sub-Tree:** Similar to a subsection of a graph being called a sub-graph, a sub-section of a decision tree is referred to as a sub-tree. It represents a specific portion of the decision tree.

**Pruning:** The process of removing or cutting down specific nodes in a decision tree to prevent overfitting and simplify the model.

**Branch / Sub-Tree:** A subsection of the entire decision tree is referred to as a branch or sub-tree. It represents a specific path of decisions and outcomes within the tree.

**Parent and Child Node:** In a decision tree, a node that is divided into sub-nodes is known as a parent node, and the sub-nodes emerging from it are referred to as child nodes. The parent node represents a decision or condition, while the child nodes represent the potential outcomes or further decisions based on that condition.

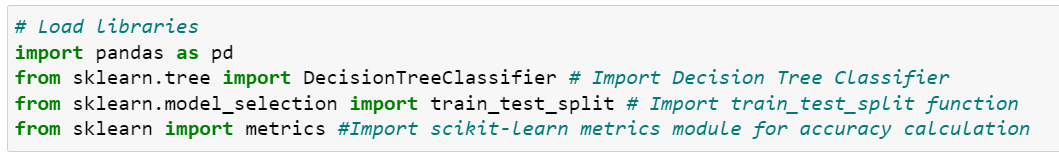


EXAMPLE USING PIMA INDIAN DIABTES DATASET.

This code is an example of building a machine learning model using a Decision Tree Classifier to predict whether a person has diabetes or not. It uses the scikit-learn library in Python.

1. **Import necessary libraries:**

Here, the code imports the required Python libraries for data manipulation, decision tree classification, and evaluation metrics.

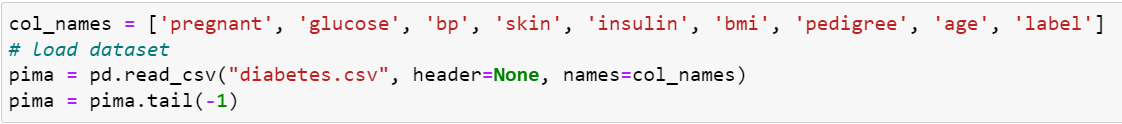


1. **Define column names:**

This line defines the names of columns in the dataset.

1. **Load the dataset:**

The code loads a dataset from a CSV file called "diabetes.csv" into a Pandas DataFrame and assigns the column names defined in col\_names. The second line removes the first row from the dataset (assuming it's a header).



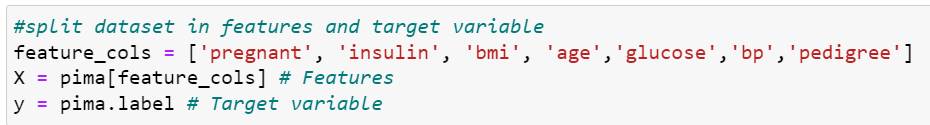
1. **Display the first few rows of the dataset:**

This line is used to display the first few rows of the loaded dataset for inspection.



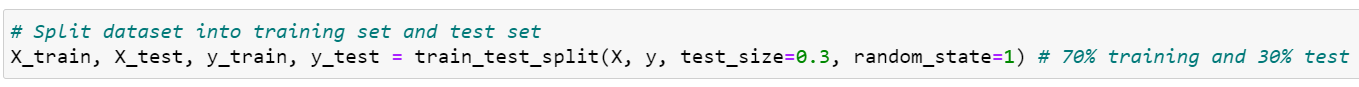
1. **Define features and target variable:**

Here, feature\_cols is a list of column names that are considered as input features (independent variables), and X is a DataFrame containing these features. y is a Series containing the target variable, which is labeled as 'label' in the dataset.



1. **Split the dataset into training and testing sets:**

This line uses the train\_test\_split function to split the dataset into a training set (X\_train and y\_train) and a testing set (X\_test and y\_test). The data is divided in a 70% training and 30% testing ratio, and random\_state is set to ensure reproducibility.



1. **Create a Decision Tree Classifier:**

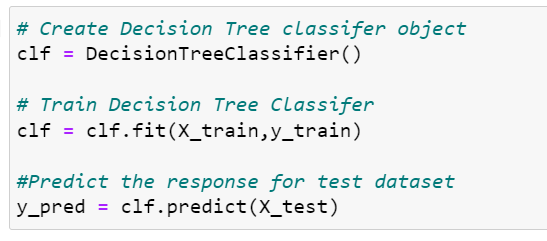
An instance of the Decision Tree Classifier is created.

1. **Train the Decision Tree Classifier:**

The Decision Tree Classifier is trained on the training data (X\_train and y\_train).

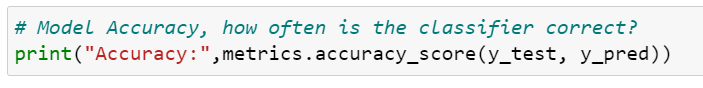
1. **Make predictions on the test data:**

The trained classifier is used to make predictions on the test data (X\_test), and the predictions are stored in y\_pred.



1. **Calculate and print the model accuracy:**

The accuracy of the model is calculated by comparing the predicted values (y\_pred) to the actual values in the test set (y\_test). The accuracy is then printed.

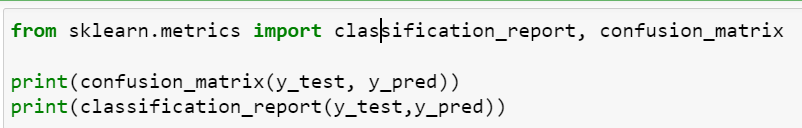


1. **Import additional metrics:**

This line imports additional evaluation metrics, including the classification report and confusion matrix.

1. **Print the confusion matrix and classification report:**

These lines print the confusion matrix and classification report, providing more detailed information about the model's performance, including precision, recall, F1-score, and support for each class.



**Accuracy:**

Purpose: Accuracy is one of the most straightforward metrics. It measures the ratio of correctly predicted instances to the total number of instances in the dataset. It provides a general idea of how well the model is performing. However, it may not be suitable for imbalanced datasets.

**Precision:**

Purpose: Precision is the ratio of correctly predicted positive instances to the total instances predicted as positive. It is useful when the cost of false positives is high. For example, in medical diagnoses, precision is crucial because you want to minimize false positives to avoid unnecessary treatments.

**Recall (Sensitivity or True Positive Rate):**

Purpose: Recall is the ratio of correctly predicted positive instances to the total actual positive instances. It is useful when the cost of false negatives is high. For example, in disease detection, recall is essential because you want to minimize false negatives to ensure all actual cases are detected.

**F1-Score:**

Purpose: The F1-Score is the harmonic mean of precision and recall. It balances both metrics and provides a single value that considers both false positives and false negatives. It is useful when you want to find a balance between precision and recall.

# Load libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model\_selection import train\_test\_split # Import train\_test\_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

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

# load dataset

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

pima = pima.tail(-1)

pima.head()

#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

# 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

# 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)

# Model Accuracy, how often is the classifier correct?

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

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test,y\_pred))