**DATE:**

**TASK5: DECISION TREE MODEL**

**AIM:** Program to implement classification using Decision Tree Model.

**Data set:** IrisData

**Source**:UCI Repository

**DECISION TREE:**

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. The core algorithm for building decision trees called ID3, which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree.

**Entropy:**

ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

**Formula for computing entropy**:

Entropy(p1,p2,……,pn)=-p1 logp1- p2 logp2-……. pn logpn

H(x) = -∑p(x)\*log(p(x))

**Information Gain:** The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

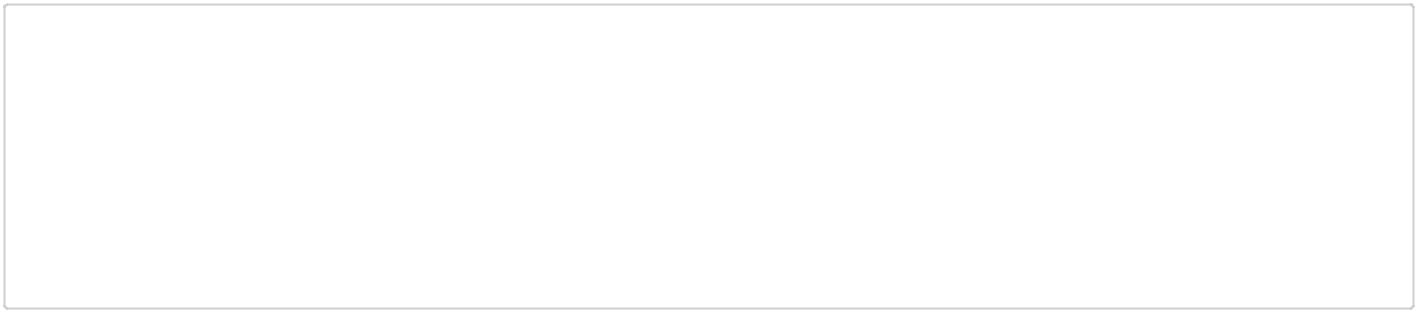
Information\_Gain = Entropy\_before - Entropy\_after

**Gain ratio:**

In decision tree gain ratio is a ratio of information gain to the intrinsic information. It is used to reduce a bias towards multi-valued attributes by taking the number and size of branches into account when choosing an attribute.

**PROGRAM:**

In [1]:



**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.datasets **import** load\_iris

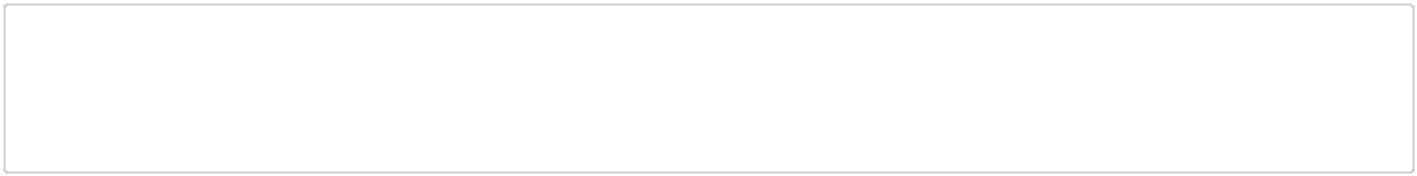
**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.model\_selection **import** train\_test\_splitdata**=**load\_iris()

print('classes to predict:',data.target\_names)

classes to predict: ['setosa' 'versicolor' 'virginica']

In [2]:



x**=**data.data

y**=**data.target

print('Number of examples in the data:',x.shape[0])

print(x[:4])

Number of examples in the data: 150

[[5.1 3.5 1.4 0.2]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]]

In [13]:



x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,random\_state**=**22,test\_size**=**0.35)

In [14]:



clf**=**DecisionTreeClassifier(criterion**=**'entropy')

In [15]:



clf.fit(x\_train,y\_train)

Out[15]:

DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=Non e,

max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

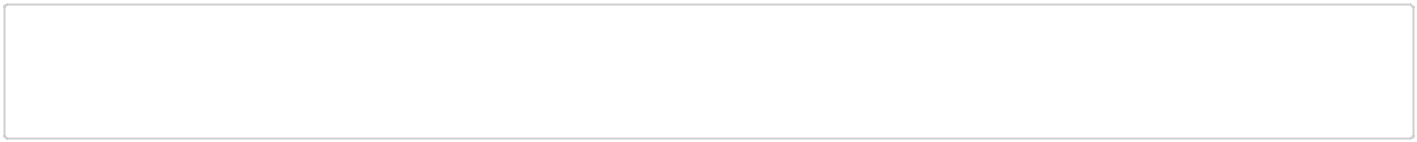
splitter='best')

In [16]:



y\_pred**=**clf.predict(x\_test)

In [17]:



**from** sklearn.metrics **import** accuracy\_score

print('Accuracy score on train data:',accuracy\_score(y\_true**=**y\_train,y\_pred**=**clf.predict(x\_t r print('Accuracy score on test data:',accuracy\_score(y\_true**=**y\_test,y\_pred**=**y\_pred))

Accuracy score on train data: 1.0

Accuracy score on test data: 0.9056603773584906

**RESULT:** The above program decision tree is executed successfully.