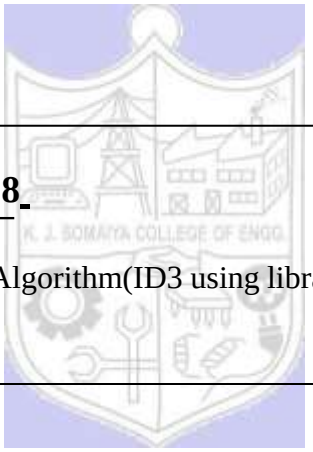


Experiment No. 8

Title: Decision Tree Algorithm(ID3 using library functions)

Batch: A1 (honours) **Roll No.: 16010421059** **Experiment No.: 8**

Aim: To implement Decision Tree Algorithm (ID3 using library functions)

Resources needed: Any Programming Language

Theory

Decision Tree is a supervised learning algorithm which falls under classification algorithm category. In machine learning, Classification involves two-steps, learning step and prediction step. In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data.

Decision Tree algorithm can be used for solving **regression and classification problems** too. Decision Tree is used to create a training model that can be further used to predict the class or value of the target variable by **learning simple decision rules** inferred from training data. In Decision Trees, for predicting a class label for a record start from the **root** of the tree and then compares the values of the root attribute with the record's attribute. On the basis of comparison, follow the branch corresponding to that value and jump to the next node.

Types of Decision Trees

Based on the type of target variable Decision Tree can be of two types:

Categorical Variable Decision Tree: Decision Tree which has a categorical target variable then it called a Categorical variable decision tree. Example: To predict whether a customer will pay his renewal premium with an insurance company (yes/ no).

Continuous Variable Decision Tree: Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree. **Example:** To predict customer income based on occupation, product, and various other variables.

Decision Tree have following:

- Root Node: It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- Splitting: It is a process of dividing a node into two or more sub-nodes.
- Decision Node: When a sub-node splits into further sub-nodes, then it is called the decision node.
- Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.
- Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- Branch / Sub-Tree: A subsection of the entire tree is called branch or sub-tree.
- Parent and Child Node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.
- In the beginning, the whole training set is considered as the root. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are distributed recursively on the basis of attribute values. Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees. Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is to know as the attributes selection. The decision tree splits the nodes on

all available variables and then selects the split which results in most homogeneous sub-nodes. The algorithm selection is also based on the type of target variables. Algorithms used in Decision Trees are:

- **ID3** → (extension of D3)
- **C4.5** → (successor of ID3)
- **CART** → (Classification And Regression Tree)
- **CHAID** → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
- **MARS** → (multivariate adaptive regression splines)

The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.

Steps in ID3 algorithm:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates **Entropy(H)** and **Information gain(IG)** of this attribute.
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.

Attribute Selection Measures

If the dataset consists of **N** attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy. For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criteria* like : **Entropy**, **Information gain, Gini index, Gain Ratio, Reduction in Variance Chi-Square**. These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root. While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

Information Gain:

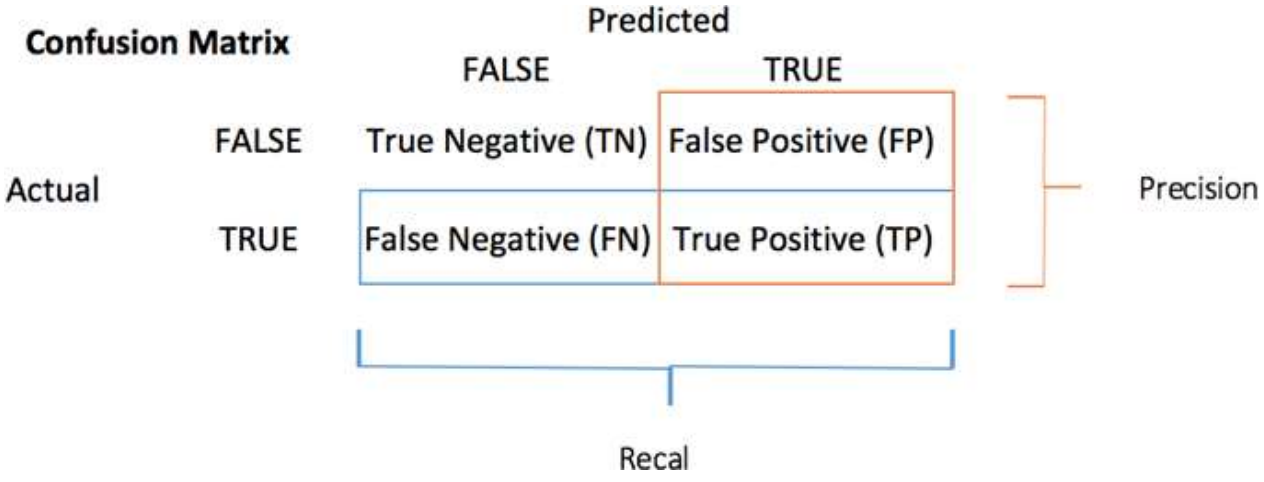
Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula: Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random. ID3 follows the rule — A branch with an entropy of zero is a leaf node and A brach with entropy more than zero needs further splitting. Mathematically Entropy for 1 attribute is represented as: Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no) **Where, S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no**

Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits. Gini index can be calculated using the below formula: Gini Index= $1- \sum_j P_j^2$

The **confusion matrix** is a better choice to evaluate the classification performance. It is based on the counts of test records correctly and incorrectly predicted by the model. Compute the accuracy test from the confusion matrix:



$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Activity:

Set up and train a decision tree classifier on the Titanic dataset and see how well the classifier performs on a validation set (80-20 train-test dataset). Find out accuracy and confusion matrix and plot created decision tree with following variations

1. Target Variable: Survived , remaining all input features
2. Target Variable: Survived , selecting subset of features as input
3. Target Variable: Survived , using transformed input feature (e.g. create new feature family =sibsp + parch, weighted_class = pclass*2 if pclass =1 ; pclass*3 if pclass =2 ; pclass*4 if pclass
=3 etc)

Results: (Softcopy submission)

+ Code + Text

[1]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
from mlxtend.plotting import plot_decision_regions
```

[2]
```
data = load_iris()
```

```
data.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

[4]
```
data.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

+ Code + Text

```
data_frame = {
    'sepal length (cm)': data.data[50:,0],
    # 'sepal width (cm)': data.data[50:,1],
    'petal length (cm)': data.data[50:,2],
    # 'petal width (cm)': data.data[50:,3],
    # 'Species': np.array([i for i in data.target if i>0])
    'Species': data.target[50:]
}
```

```
[6] df =pd.DataFrame(data_frame)
    df
```

	sepal length (cm)	petal length (cm)	Species
0	7.0	4.7	1
1	6.4	4.5	1
2	6.9	4.9	1
3	5.5	4.0	1
4	6.5	4.6	1
...	...	...	...
95	6.7	5.2	2

+ Code + Text

[6]

95	6.7	5.2	2
96	6.3	5.0	2
97	6.5	5.2	2
98	6.2	5.4	2
99	5.9	5.1	2

100 rows × 3 columns

[7]
```
df.nunique()
```

```
sepal length (cm)    28
petal length (cm)    34
Species               2
dtype: int64
```

[8]
```
df = df.sample(df.shape[0])
df
```

+ Code + Text

[8]

	sepal length (cm)	petal length (cm)	Species
73	6.3	4.9	2
98	6.2	5.4	2
64	5.8	5.1	2
43	5.0	3.3	1
78	6.4	5.6	2
...	...	...	...
1	6.4	4.5	1
76	6.2	4.8	2
21	6.1	4.0	1
28	6.0	4.5	1
81	7.9	6.4	2

100 rows × 3 columns

[9]

```
df_train = df.iloc[:60,:].sample(10)
dt_test = df.iloc[61:,:].sample(10)
```

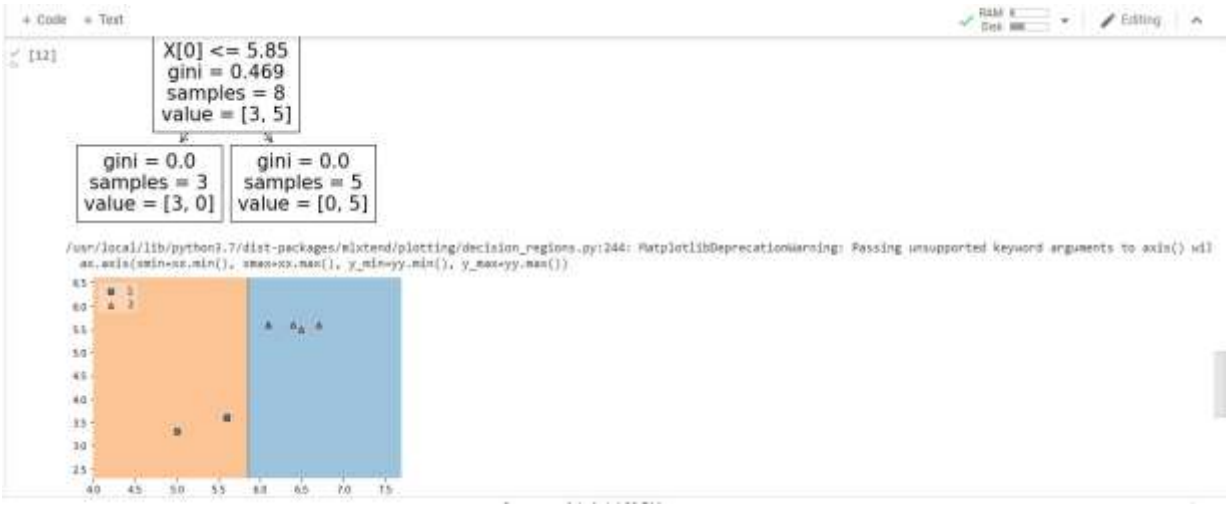
```
[10] dftest = dt_test.sample(40,replace=True)
     x_test = dftest.iloc[:,:-1].values
     y_test = dftest.iloc[:,-1].values
```

```
[11] # model
     def evaluate(model, x,y):

       model.fit(x,y)
       plot_tree(model)
       plt.show()
       plot_decision_regions(x,y,model,legend=2)
       plt.show()

       y_pred = model.predict(x_test)
       print("y_test:" ,y_test)
       print("y_pred:" ,y_pred)
       print("accuracy score",accuracy_score(y_test,y_pred)*100)
       return model
```

```
[12] df_t = df_train.sample(8,replace = True)
     x_train = df_t.iloc[:,:-1].values
     y_train = df_t.iloc[:,-1].values
     dt1 = DecisionTreeClassifier()
     bag1 = evaluate(dt1,x_train,y_train)
```



+ Code + Text

```
[12]  40   45   50   55   60   65   70   75
y_test: [1 2 2 2 1 1 1 2 2 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1 2 2 2 2 2 1 1
 1 1 2]
y_pred: [1 2 2 2 1 1 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 2 1 1 1 2 2 1 2 1 2 2 2 2 2 2 1
 2 1 2]
accuracy score 70.0
```

+ Code + Text

```
[13] data_frame_2 = {
        'sepal length (cm)': data.data[50:,0],
        'sepal width (cm)': data.data[50:,1],
        'petal length (cm)': data.data[50:,2],
        'petal width (cm)': data.data[50:,3],
        # 'Species': np.array([i for i in data.target if i>0])
        'Species': data.target[50:]
    }
```

```
df_2 =pd.DataFrame(data_frame_2)
df_2
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	7.0	3.2	4.7	1.4	1
1	6.4	3.2	4.5	1.5	1
2	6.9	3.1	4.9	1.5	1
3	5.5	2.3	4.0	1.3	1
4	6.5	2.8	4.6	1.5	1
...	...	...	...	...	...
95	6.7	3.0	5.2	2.3	2
96	6.3	2.5	5.0	1.9	2
97	6.5	3.0	5.2	2.0	2
98	6.2	3.4	5.4	2.3	2
99	5.9	3.0	5.1	1.8	2

100 rows × 5 columns

```
In [3]: df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis='columns',inplace=True)
```

```
In [4]: df.head()
```

Out[4]:

	Survived	Pclass	Sex	Age	Fare
0	0	3	male	22.0	7.2500
1	1	1	female	38.0	71.2833
2	1	3	female	26.0	7.9250
3	1	1	female	35.0	53.1000
4	0	3	male	35.0	8.0500

```
In [5]: dfnew = df.head(50)
df1 = dfnew.drop('Survived', axis='columns')
target  = dfnew.Survived
```

In [6]:
```
#Label Encoding
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df1['Sex'] = label_encoder.fit_transform(df1['Sex'])
```

In [7]:
```
#Removing Null Values
df1.Age = df1.Age.fillna(df1.Age.mean())
df1.head()
```

Out[7]:

	Pclass	Sex	Age	Fare
0	3	1	22.0	7.2500
1	1	0	38.0	71.2833
2	3	0	26.0	7.9250
3	1	0	35.0	53.1000
4	3	1	35.0	8.0500

In [8]:
```
from sklearn.tree import DecisionTreeClassifier

#Splitting up the dataset
from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(df1,target,test_size=0.2)

#Training the decision tree classifier
m2 = DecisionTreeClassifier(criterion='gini')
m2.fit(X_train2,y_train2)

#Predicting the response for test data
y_pred2 = m2.predict(X_test2)

#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test2,y_pred2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         6
           1       0.67      1.00      0.80         4

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
weighted avg       0.87      0.80      0.80        10
```

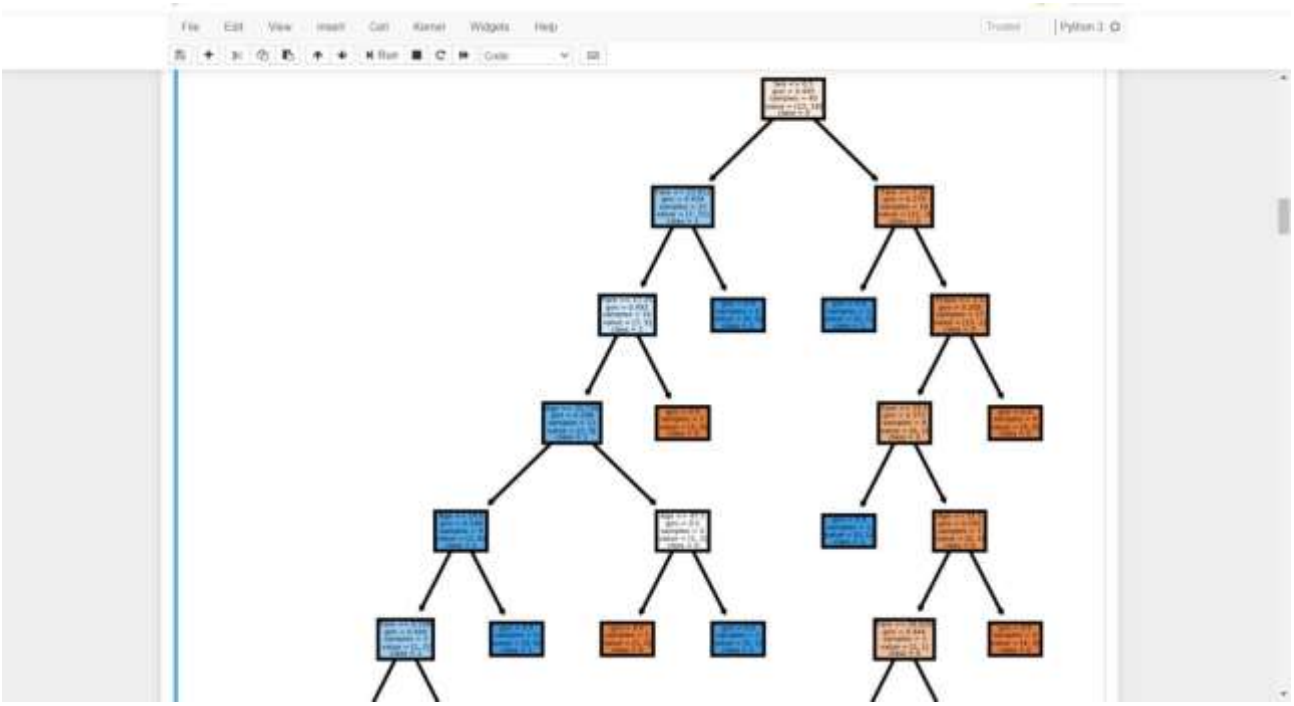
In [9]:
```
#Accuracy
print("accuracy: ", m2.score(X_test2,y_test2))
```

```
accuracy:  0.8
```

In [11]:
```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test2,y_pred2))
```

```
[[4 2]
 [0 4]]
```

In [14]:
```
target = target.astype('str')
```



In [17]:
```
#Example 1
import pandas as pd
df = pd.read_csv("titanic_data.csv")
df2 = df.head(50).drop('Cabin', axis = 1)
df2
```

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	S

In [18]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

#Label Encoding
df2.loc[:,('Name')] = label_encoder.fit_transform(df2['Name'])
df2.loc[:,('Sex')] = label_encoder.fit_transform(df2['Sex'])
df2.loc[:,('Ticket')] = label_encoder.fit_transform(df2['Ticket'])

#Handling Null Values
df2.loc[:,('Age')] = df2.Age.fillna(df2.Age.mean())

em_dummies = pd.get_dummies(df2['Embarked'], drop_first=True)
df2[['Embarked_Q','Embarked_S']] = em_dummies
df2.head()
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Embarked_Q	Embarked_S
0	1	0	3	7	1	22.0	1	0	38	7.2500	S	0	1
1	2	1	1	9	0	38.0	1	0	42	71.2833	C	0	0
2	3	1	3	16	0	26.0	0	0	48	7.9250	S	0	1
3	4	1	1	13	0	35.0	1	0	3	53.1000	S	0	1
4	5	0	3	1	1	35.0	0	0	34	8.0500	S	0	1

In [19]:
```
df2.drop(['Embarked'],axis=1,inplace=True)
df2.head()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked_Q	Embarked_S
0	1	0	3	7	1	22.0	1	0	38	7.2500	0	1
1	2	1	1	9	0	38.0	1	0	42	71.2833	0	0
2	3	1	3	16	0	26.0	0	0	48	7.9250	0	1
3	4	1	1	13	0	35.0	1	0	3	53.1000	0	1
4	5	0	3	1	1	35.0	0	0	34	8.0500	0	1

In [20]:
```
#Getting the target column
target1  = df2.Survived
df2 = df2.drop('Survived', axis='columns')
```

In [21]:
```
target1 = target1.astype('float')
```

In [22]:
```
from sklearn.tree import DecisionTreeClassifier

#Splitting up the dataset
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(df2,target1,test_size=0.2)

#Training the decision tree classifier
m1 = DecisionTreeClassifier(criterion='entropy')
m1.fit(X_train1,y_train1)

#Predicting the response for test data
y_pred1 = m1.predict(X_test1)

#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test1,y_pred1))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.88      0.88         8
         1.0       0.50      0.50      0.50         2

    accuracy                           0.80        10
   macro avg       0.69      0.69      0.69        10
weighted avg       0.80      0.80      0.80        10
```

In [23]:
```
#Accuracy
print("accuracy: ", m1.score(X_test1,y_test1))
```

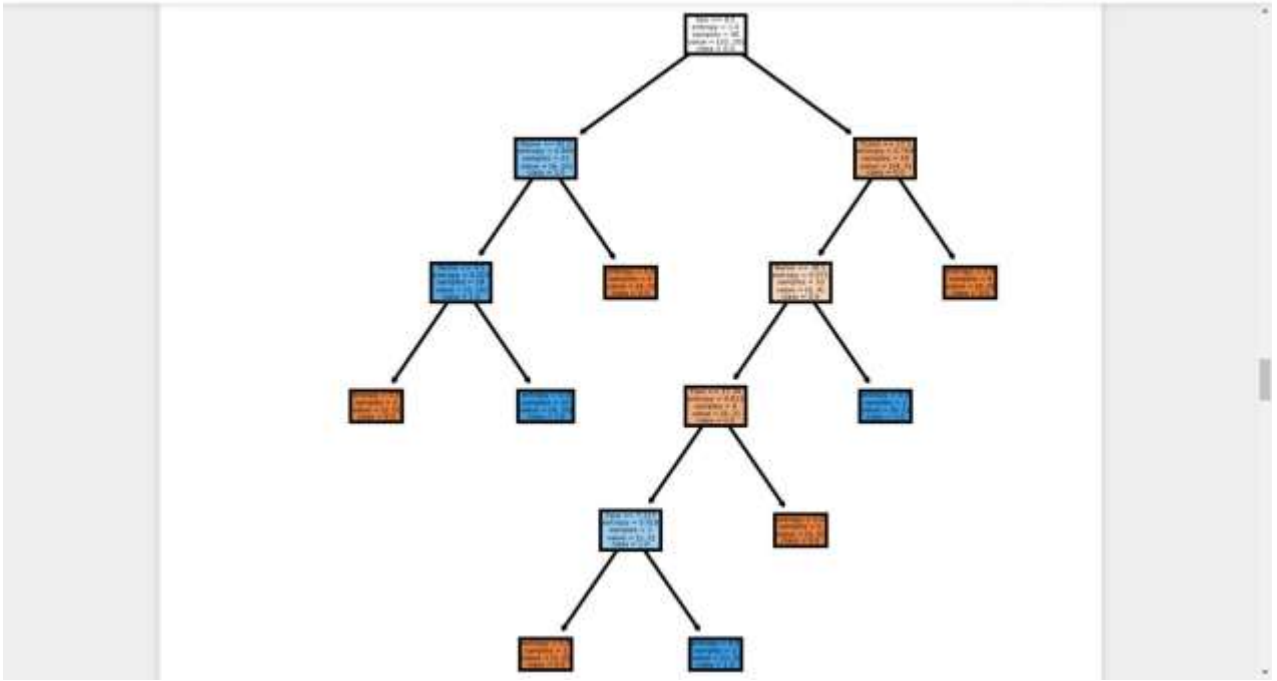
```
accuracy:  0.8
```

In [24]:
```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test1,y_pred1))
```

```
[[7 1]
 [1 1]]
```

```
target1 = target1.astype('str')

from sklearn import tree
%matplotlib inline
from matplotlib import pyplot as plt
fig1,axes1 = plt.subplots(nrows=1,ncols=1,figsize=(4,4),dpi=400)
tree.plot_tree(m1,feature_names=df2.columns,class_names=target1,filled=True, fontsize=2)
```



In [26]:
```
#Example3

df3 = pd.read_csv("titanic_data.csv")
df3.head()
X = df3.drop(['Fare','Age','Cabin'], axis=1)
X['Family'] = X['SibSp'] + X['Parch']
X = X.drop(['SibSp', 'Parch'], axis = 1)
X.head()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Ticket	Embarked	Family
0	1	0	3	Braund, Mr. Owen Harris	male	A/5 21171	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C	1
2	3	1	3	Heikkinen, Miss. Laina	female	STON/O2. 3101282	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	S	1
4	5	0	3	Allen, Mr. William Henry	male	373450	S	0

In [27]:
```
def weClass(x):
    if x==1:
        return x*2
    elif x==2:
        return x*3
    else:
        return x*4

X['Weighted_class'] = X['Pclass'].apply(weClass)
X.head()
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Ticket	Embarked	Family	Weighted_class
0	1	0	3	Braund, Mr. Owen Harris	male	A/5 21171	S	1	12
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C	1	2
2	3	1	3	Heikkinen, Miss. Laina	female	STON/O2. 3101282	S	0	12
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	S	1	2
4	5	0	3	Allen, Mr. William Henry	male	373450	S	0	12

In [28]:
```
newdf=X.drop(['Pclass'], axis = 1)
newdf.head()
```

Out[28]:

	PassengerId	Survived	Name	Sex	Ticket	Embarked	Family	Weighted_class
0	1	0	Braund, Mr. Owen Harris	male	A/5 21171	S	1	12
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C	1	2
2	3	1	Heikkinen, Miss. Laina	female	STON/O2. 3101282	S	0	12
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	S	1	2
4	5	0	Allen, Mr. William Henry	male	373450	S	0	12

In [29]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

#Label Encoding
newdf.loc[:,('Name')] = label_encoder.fit_transform(newdf['Name'])
newdf.loc[:,('Sex')] = label_encoder.fit_transform(newdf['Sex'])
newdf.loc[:,('Ticket')] = label_encoder.fit_transform(newdf['Ticket'])


em_dummies = pd.get_dummies(X['Embarked'], drop_first=True)
newdf[['Embarked_Q','Embarked_S']] = em_dummies
newdf =  newdf.drop(['Embarked'], axis =1 )
newdf.head()
```

Out[29]:

	PassengerId	Survived	Name	Sex	Ticket	Family	Weighted_class	Embarked_Q	Embarked_S
0	1	0	108	1	523	1	12	0	1
1	2	1	190	0	596	1	2	0	0
2	3	1	353	0	669	0	12	0	1
3	4	1	272	0	49	1	2	0	1
4	5	0	15	1	472	0	12	0	1

In [31]:
```
y = newdf.Survived
y = y.astype('float')
```

In [32]:
```
from sklearn.tree import DecisionTreeClassifier

#Splitting up the dataset
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(newdf,y,test_size=0.2)

#Training the decision tree classifier
m1 = DecisionTreeClassifier(criterion='entropy')
m1.fit(X_train1,y_train1)

#Predicting the response for test data
y_pred1 = m1.predict(X_test1)

#Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_test1,y_pred1))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       100
         1.0       1.00      1.00      1.00        79

    accuracy                           1.00       179
   macro avg       1.00      1.00      1.00       179
weighted avg       1.00      1.00      1.00       179
```

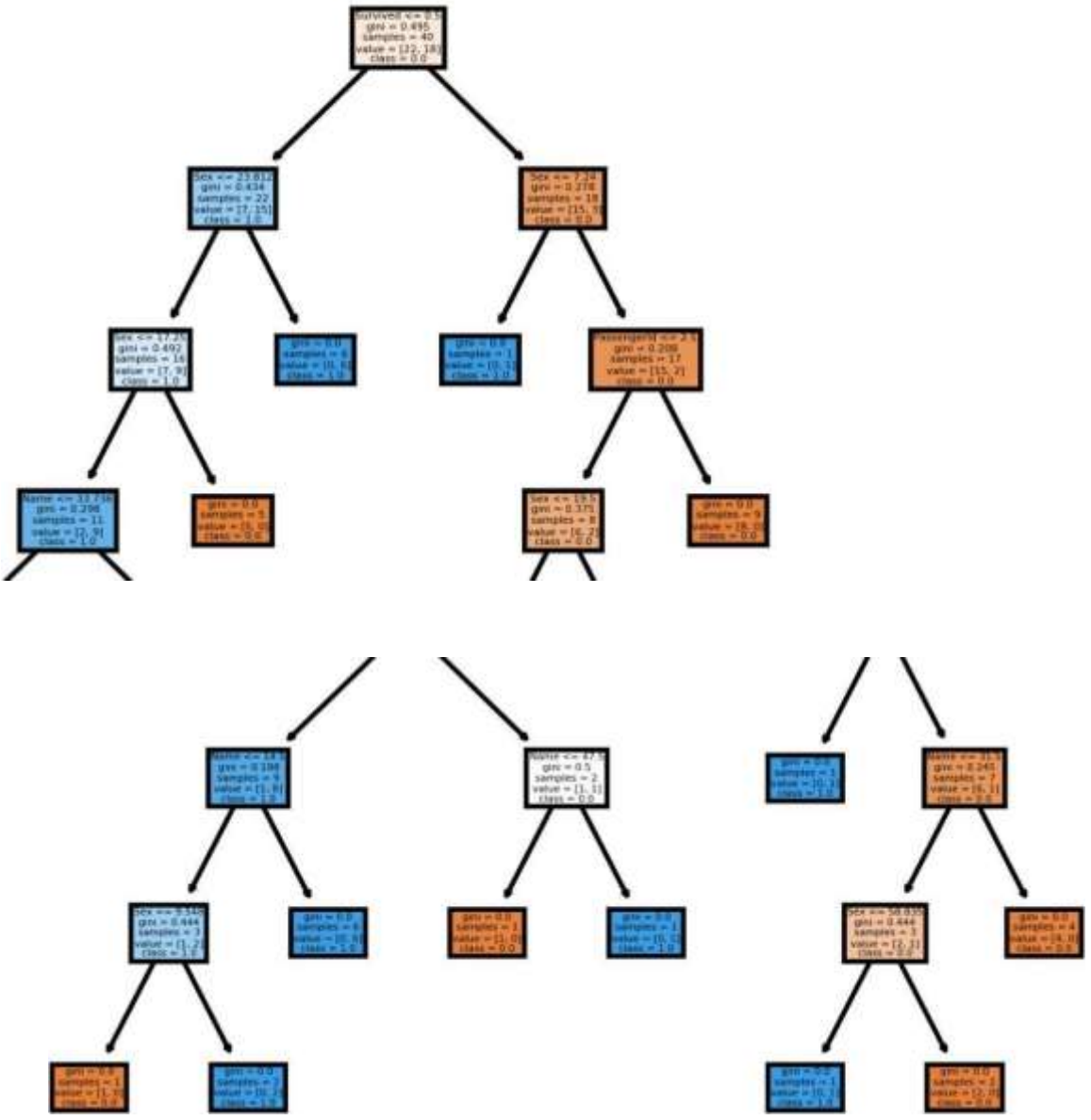
In [33]:
```
#Accuracy
print("accuracy: ", m1.score(X_test1,y_test1))
```

```
accuracy:  1.0
```

In [34]:
```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test1,y_pred1))
```

```
[[100   0]
 [  0  79]]
```

In [35]:
```
from sklearn import tree
y = y.astype('str')
%matplotlib inline
from matplotlib import pyplot as plt
fig2,axes2 = plt.subplots(nrows=1,ncols=1,figsize=(4,4),dpi=400)
tree.plot_tree(m2,feature_names=newdf.columns,class_names=y,filled=True, fontsize=2)
```



Outcomes: CO5 Understand fundamentals of learning in AI.

Conclusion: We tested and understood the implementation of a Decision Tree Algorithm (ID3 using library functions).

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication
2. Jason Brownlee , Master Machine Learning Algorithms, eBook, 2017 ,Edition v1.12.