LAB 4

Task 2 : Apply algorithm on wine dataset - LabelEncoding of features: and Train test Division 66%-34%

In [1]:
```
from sklearn import datasets
from sklearn import preprocessing

wine_data = datasets.load_wine()
```

In [2]:
```
print("Features: ", wine_data.feature_names)
print("\nLabels: ", wine_data.target_names)

print('\nData : \n',wine_data.data[0:3,:])
print('\nTarget : \n',wine_data.target)

wine_data.data.shape
```

```
Features:  ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesiu
m', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanin
s', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']

Labels:  ['class_0' 'class_1' 'class_2']

Data :
 [[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]]

Target :
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

Out[2]: (178, 13)

In [3]:
```
from sklearn.model_selection import train_test_split

#split data set into train and test sets

x_train, x_test, y_train, y_test = train_test_split(wine_data.data,
                        wine_data.target, test_size = 0.34, random_state = 1
29)
```

In [4]:
```
from sklearn.tree import DecisionTreeClassifier

#Create a Decision Tree Classifier (using Entropy)
clf = DecisionTreeClassifier(criterion='entropy')

# Train the model using the training sets
clf.fit(x_train,y_train)
```

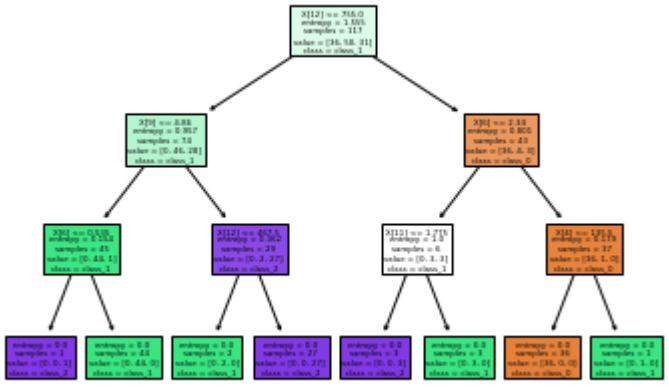
Out[4]: DecisionTreeClassifier(criterion='entropy')

In [5]:
```
#Predict the response for test dataset
y_pred = clf.predict(x_test)
```

In [7]:
```
from sklearn import tree

tree.plot_tree(clf,filled=True,class_names=['class_0','class_1','class_2'])
```

Out[7]:
```
[Text(167.4, 190.26, 'X[12] <= 755.0\nentropy = 1.555\nsamples = 117\nvalue
= [36, 50, 31]\nclass = class_1'),
 Text(83.7, 135.9, 'X[9] <= 4.86\nentropy = 0.957\nsamples = 74\nvalue = [0,
46, 28]\nclass = class_1'),
 Text(41.85, 81.53999999999999, 'X[6] <= 0.535\nentropy = 0.154\nsamples = 4
5\nvalue = [0, 44, 1]\nclass = class_1'),
 Text(20.925, 27.180000000000007, 'entropy = 0.0\nsamples = 1\nvalue = [0,
0, 1]\nclass = class_2'),
 Text(62.775000000000006, 27.180000000000007, 'entropy = 0.0\nsamples = 44\n
value = [0, 44, 0]\nclass = class_1'),
 Text(125.55000000000001, 81.53999999999999, 'X[12] <= 467.5\nentropy = 0.36
2\nsamples = 29\nvalue = [0, 2, 27]\nclass = class_2'),
 Text(104.625, 27.180000000000007, 'entropy = 0.0\nsamples = 2\nvalue = [0,
2, 0]\nclass = class_1'),
 Text(146.475, 27.180000000000007, 'entropy = 0.0\nsamples = 27\nvalue = [0,
0, 27]\nclass = class_2'),
 Text(251.10000000000002, 135.9, 'X[6] <= 2.34\nentropy = 0.801\nsamples = 4
3\nvalue = [36, 4, 3]\nclass = class_0'),
 Text(209.25, 81.53999999999999, 'X[11] <= 1.775\nentropy = 1.0\nsamples = 6
\nvalue = [0, 3, 3]\nclass = class_1'),
 Text(188.32500000000002, 27.180000000000007, 'entropy = 0.0\nsamples = 3\nv
alue = [0, 0, 3]\nclass = class_2'),
 Text(230.175, 27.180000000000007, 'entropy = 0.0\nsamples = 3\nvalue = [0,
3, 0]\nclass = class_1'),
 Text(292.95, 81.53999999999999, 'X[4] <= 135.5\nentropy = 0.179\nsamples =
37\nvalue = [36, 1, 0]\nclass = class_0'),
 Text(272.02500000000003, 27.180000000000007, 'entropy = 0.0\nsamples = 36\n
value = [36, 0, 0]\nclass = class_0'),
 Text(313.875, 27.180000000000007, 'entropy = 0.0\nsamples = 1\nvalue = [0,
1, 0]\nclass = class_1')]
```



In [8]:
```
from sklearn.metrics import confusion_matrix,precision_score,recall_score,accuracy_score

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",accuracy_score(y_test, y_pred))

print("\nConfusion Matrix :")
confusion_matrix(y_test, y_pred)
```

```
Accuracy: 0.819672131147541

Confusion Matrix :
```

Out[8]:
```
array([[19,  4,  0],
       [ 0, 20,  1],
       [ 0,  6, 11]])
```

In [10]:
```
precision = precision_score(y_test, y_pred,average='micro')
print('\nprecision: {}'.format(precision))

recall = recall_score(y_test, y_pred,average='micro')
print('\nrecall: {}'.format(recall))
```

```
precision: 0.819672131147541

recall: 0.819672131147541
```