Import the Required Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Import the Data Set fruits.xlsx

In [2]:
```
fruits = pd.read_excel('c:/data/fruits.xlsx',sheet_name='fruits')
fruits.shape
```

Out[2]:
```
(60, 4)
```

Perform the Basic EDA

In [3]:
```
fruits.head()
```

Out[3]:

	Color	Sphericity	Weight	labels
0	Orange	0.947	120	Orange
1	Orange	0.917	125	Orange
2	Orange	0.923	110	Orange
3	Orange	0.916	123	Orange
4	Orange	0.943	124	Orange

In [4]:
```
fruits.labels.value_counts()
```

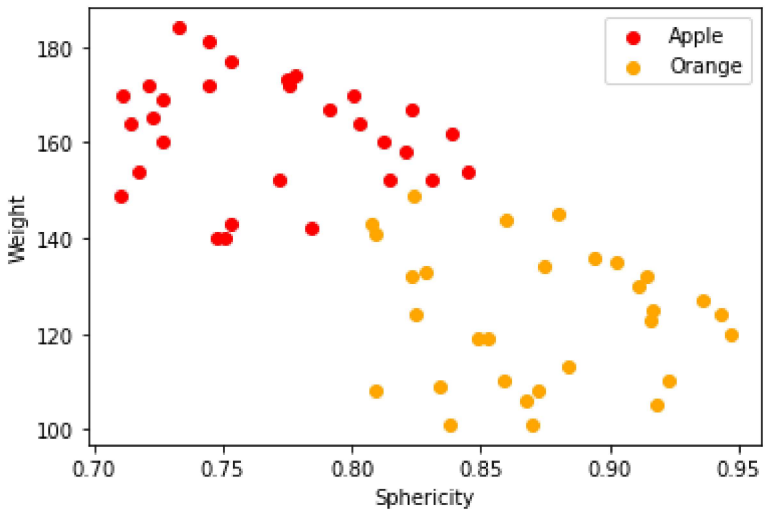
Out[4]:
```
Apple     30
Orange    30
Name: labels, dtype: int64
```

In [5]:
```
fruits[['Sphericity','Weight','labels']].groupby('labels').describe()
```

Out[5]:

	Sphericity								Weight	
	count	mean	std	min	25%	50%	75%	max	count	mean
labels										
Apple	30.0	0.768133	0.042097	0.710	0.7285	0.7625	0.80250	0.845	30.0	161.966667	12.07
Orange	30.0	0.873033	0.042740	0.808	0.8350	0.8710	0.91325	0.947	30.0	123.533333	14.12

In [7]:
```
plt.scatter(x='Sphericity',y='Weight',data=fruits[fruits.labels=='Apple'],c='red'
plt.scatter(x='Sphericity',y='Weight',data=fruits[fruits.labels=='Orange'],c='ora
plt.legend()
plt.xlabel("Sphericity")
plt.ylabel('Weight')
plt.show()
```



In [8]:
```
X = fruits[['Sphericity','Weight']]
y = fruits['labels']
```

Prepare the Train and Test Samples

In [9]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[9]:
```
((45, 2), (15, 2), (45,), (15,))
```

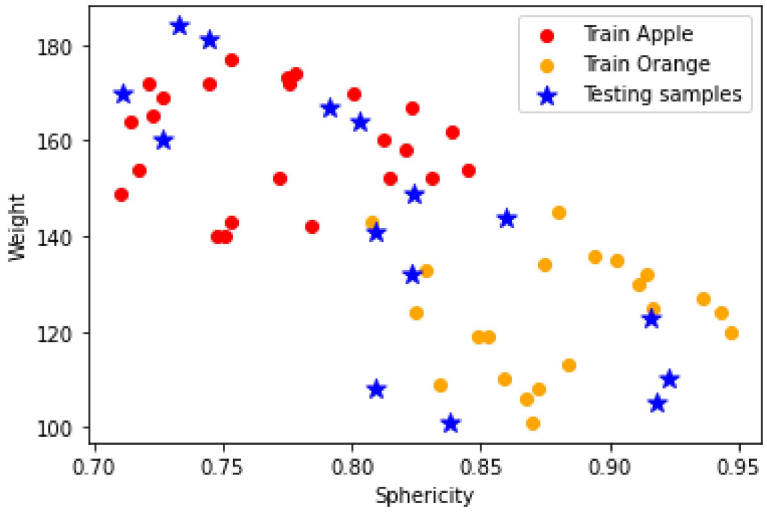
In [10]:
```
y_train.value_counts()
```

Out[10]:
```
Apple     24
Orange    21
Name: labels, dtype: int64
```

In [11]:
```
y_test.value_counts()
```

Out[11]:
```
Orange    9
Apple     6
Name: labels, dtype: int64
```

In [12]:
```
plt.scatter(x='Sphericity',y='Weight',data=X_train[y_train=='Apple'],c='red',labe
plt.scatter(x='Sphericity',y='Weight',data=X_train[y_train=='Orange'],c='orange',
plt.scatter(x='Sphericity',y='Weight',data=X_test,c='blue',label='Testing samples
plt.legend()
plt.xlabel("Sphericity")
plt.ylabel('Weight')
plt.show()
```



Train the Data model / Model Fitting

In [31]:
```
from sklearn.tree import DecisionTreeClassifier,export_text,plot_tree
clf = DecisionTreeClassifier(criterion='entropy')
```

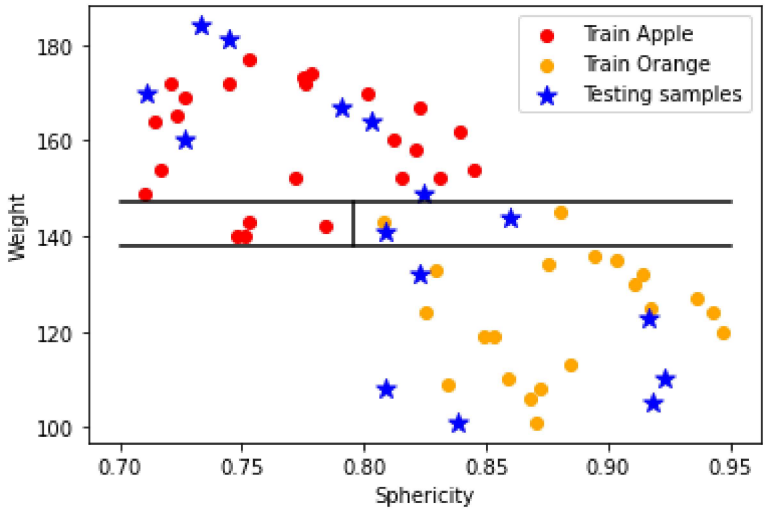
In [32]:
```
clf.fit(X_train,y_train)
```

Out[32]:
```
DecisionTreeClassifier(criterion='entropy')
```

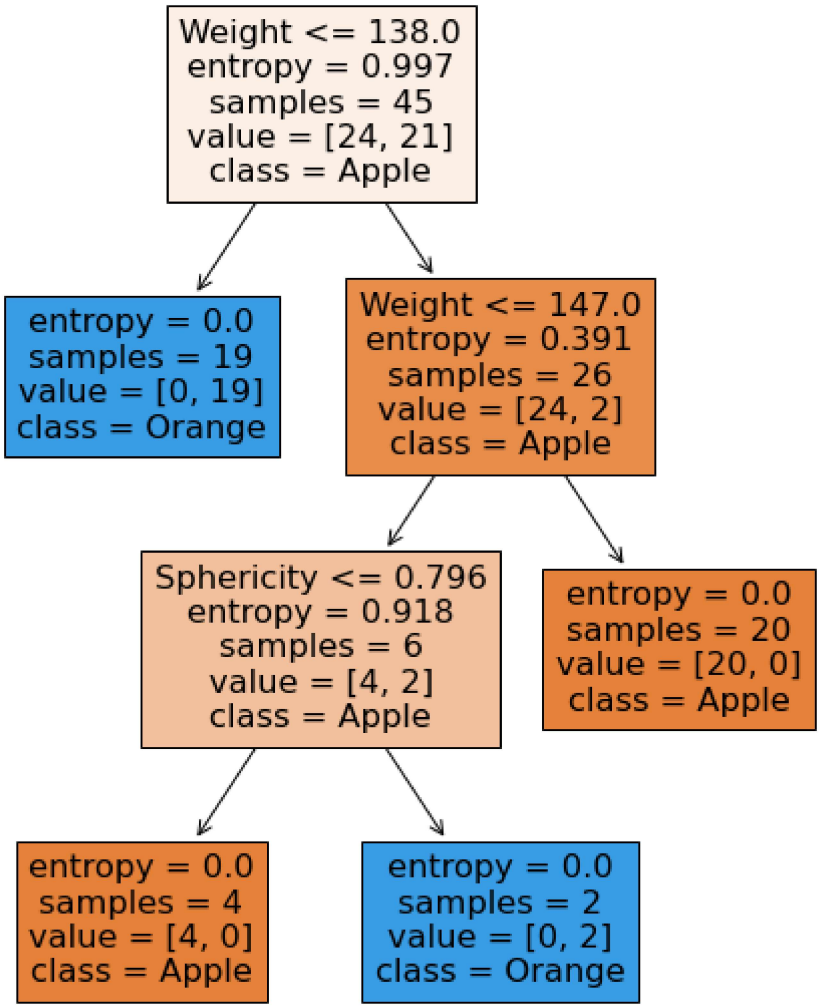
In [33]:
```
results = export_text(clf,feature_names=['Sphericity','Weight'],show_weights=True
print(results)
```

```
|--- Weight <= 138.000
|   |--- weights: [0.000, 19.000] class: Orange
|--- Weight >  138.000
|   |--- Weight <= 147.000
|   |   |--- Sphericity <= 0.796
|   |   |   |--- weights: [4.000, 0.000] class: Apple
|   |   |--- Sphericity >  0.796
|   |   |   |--- weights: [0.000, 2.000] class: Orange
|   |--- Weight >  147.000
|   |   |--- weights: [20.000, 0.000] class: Apple
```

In [34]:
```
plt.scatter(x='Sphericity',y='Weight',data=X_train[y_train=='Apple'],c='red',labe
plt.scatter(x='Sphericity',y='Weight',data=X_train[y_train=='Orange'],c='orange',
plt.scatter(x='Sphericity',y='Weight',data=X_test,c='blue',label='Testing samples
plt.plot([0.7,0.95],[138,138],c='black')
plt.plot([0.7,0.95],[147,147],c='black')
plt.plot([0.796,0.796],[138,147],c='black')
plt.legend()
plt.xlabel("Sphericity")
plt.ylabel('Weight')
plt.show()
```



In [35]:
```
plt.figure(figsize=[8,10])
plot_tree(clf,feature_names=['Sphericity','Weight'],class_names=['Apple','Orange'
plt.show()
```



Evaluation of the model with the Testing samples

In [38]:
```
y_predict = clf.predict(X_test)
y_predict
```

Out[38]:
```
array(['Apple', 'Orange', 'Apple', 'Orange', 'Apple', 'Orange', 'Apple',
       'Orange', 'Orange', 'Orange', 'Apple', 'Apple', 'Orange', 'Apple',
       'Orange'], dtype=object)
```

In [40]:
```
y_test.values
```

Out[40]:
```
array(['Apple', 'Orange', 'Apple', 'Orange', 'Orange', 'Orange', 'Apple',
       'Orange', 'Orange', 'Orange', 'Apple', 'Apple', 'Orange', 'Apple',
       'Orange'], dtype=object)
```

In [44]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,plot_confusion_matri
```

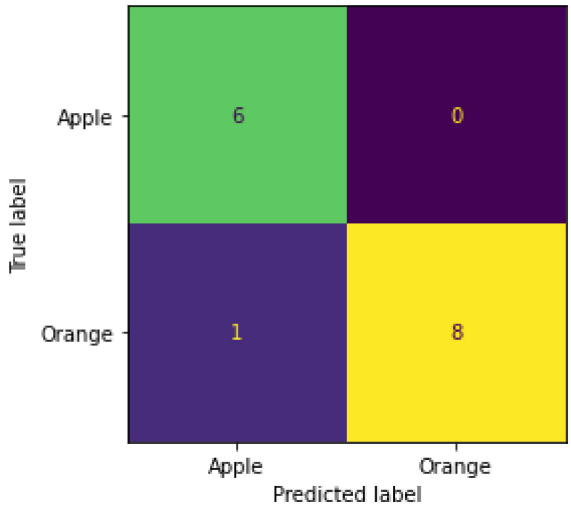
In [42]:
```
confusion_matrix(y_test,y_predict)
```

Out[42]:
```
array([[6, 0],
       [1, 8]], dtype=int64)
```

In [43]:
```
accuracy_score(y_test,y_predict)
```

Out[43]:
```
0.9333333333333333
```

In [47]:
```
plot_confusion_matrix(clf,X_test,y_test,colorbar=False)
plt.show()
```



In [190]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier,RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix,accuracy_score,confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
BaggingClassifier
```

In [171]:
```
fruits = pd.read_excel('c:/data/fruits.xlsx')
X = fruits[['Sphericity','Weight']]
y = fruits['labels']
```

In [172]:
```
encoding = LabelEncoder()
y = encoding.fit_transform(y)
```

In [173]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=10)
```

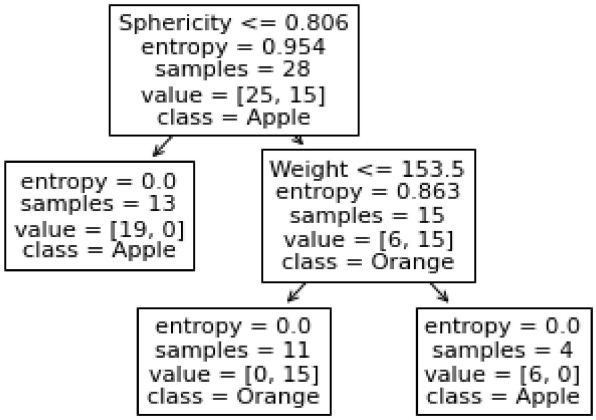
In [174]:
```
clf = DecisionTreeClassifier(criterion='entropy')
bag_clf = BaggingClassifier(base_estimator=clf,
                            n_estimators=3,
                            max_samples=40,
                           bootstrap=True,
                            random_state=10)
bag_clf.fit(X,y)
bag_clf.score(X,y)
```

Out[174]:
```
1.0
```

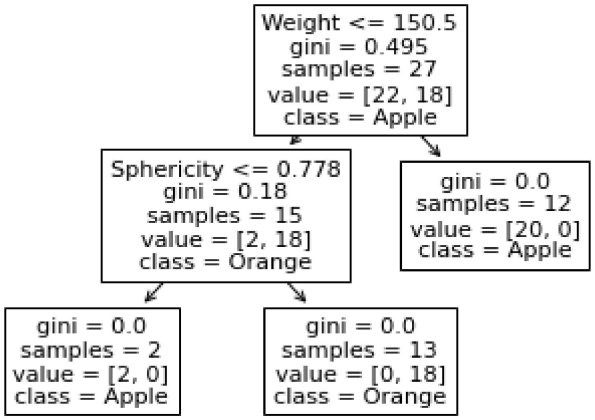
In [175]:
```
bag_clf.estimators_
```

Out[175]:
```
[DecisionTreeClassifier(criterion='entropy', random_state=396969381),
 DecisionTreeClassifier(criterion='entropy', random_state=208379487),
 DecisionTreeClassifier(criterion='entropy', random_state=2110940569)]
```

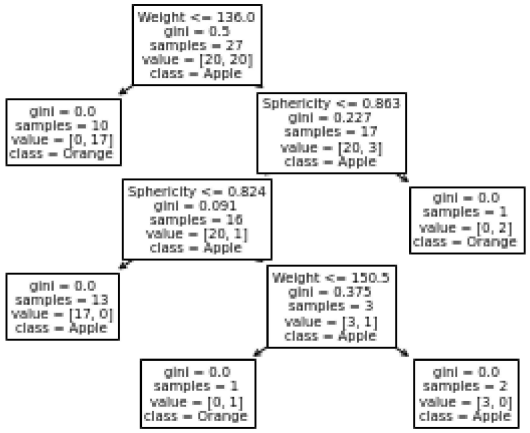
In [176]:
```
_ = plot_tree(bag_clf.estimators_[0],feature_names=X.columns,
              class_names=['Apple','Orange'])
```



In [130]:
```
_ = plot_tree(bag_clf.estimators_[1],feature_names=X.columns,
              class_names=['Apple','Orange'])
```



In [131]:
```
_ = plot_tree(bag_clf.estimators_[2],feature_names=X.columns,
              class_names=['Apple','Orange'])
```



In [132]:
```
y.iloc[bag_clf.estimators_samples_[0]].value_counts()
```

Out[132]:
```
Apple     25
Orange    15
Name: labels, dtype: int64
```

In [133]:
```
import numpy as np
len(np.unique(bag_clf.estimators_samples_[0]))
```

Out[133]:
```
28
```

In [134]:
```
y.iloc[bag_clf.estimators_samples_[1]].value_counts()
```

Out[134]:
```
Apple     22
Orange    18
Name: labels, dtype: int64
```

In [135]:
```
len(np.unique(bag_clf.estimators_samples_[1]))
```

Out[135]:
```
27
```

In [136]:
```
y.iloc[bag_clf.estimators_samples_[2]].value_counts()
```

Out[136]:
```
Orange    20
Apple     20
Name: labels, dtype: int64
```

In [137]:
```
len(np.unique(bag_clf.estimators_samples_[2]))
```

Out[137]:
```
27
```

In [152]:
```
bag_clf.get_params()
```

Out[152]:
```
{'base_estimator__ccp_alpha': 0.0,
 'base_estimator__class_weight': None,
 'base_estimator__criterion': 'entropy',
 'base_estimator__max_depth': None,
 'base_estimator__max_features': None,
 'base_estimator__max_leaf_nodes': None,
 'base_estimator__min_impurity_decrease': 0.0,
 'base_estimator__min_impurity_split': None,
 'base_estimator__min_samples_leaf': 1,
 'base_estimator__min_samples_split': 2,
 'base_estimator__min_weight_fraction_leaf': 0.0,
 'base_estimator__random_state': None,
 'base_estimator__splitter': 'best',
 'base_estimator': DecisionTreeClassifier(criterion='entropy'),
 'bootstrap': True,
 'bootstrap_features': False,
 'max_features': 1.0,
 'max_samples': 40,
 'n_estimators': 3,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 10,
 'verbose': 0,
 'warm_start': False}
```

In [182]:
```
knn_clf = KNeighborsClassifier(n_neighbors=5)
bag_clf = BaggingClassifier(base_estimator=knn_clf,n_estimators=5,
                            max_samples=45)
bag_clf.fit(X,y)
bag_clf.score(X,y)
```

Out[182]:
```
0.9166666666666666
```

In [183]:
```
bag_clf.estimators_
```

Out[183]:
```
[KNeighborsClassifier(),
 KNeighborsClassifier(),
 KNeighborsClassifier(),
 KNeighborsClassifier(),
 KNeighborsClassifier()]
```

In [184]:
```
y_predict = bag_clf.estimators_[0].predict(X)
accuracy_score(y_predict,y)
```

Out[184]:
```
0.9166666666666666
```

In [188]:
```
y_predict = bag_clf.estimators_[1].predict(X)
accuracy_score(y_predict,y)
```

Out[188]:
```
0.8666666666666667
```

In [185]:
```
y_predict = bag_clf.estimators_[2].predict(X)
accuracy_score(y_predict,y)
```

Out[185]: 0.95

In [186]:
```
y_predict = bag_clf.estimators_[3].predict(X)
accuracy_score(y_predict,y)
```

Out[186]: 0.9166666666666666

In [187]:
```
y_predict = bag_clf.estimators_[4].predict(X)
accuracy_score(y_predict,y)
```

Out[187]: 0.95

In [189]:
```
bag_clf.predict_proba(X)
```

Out[189]:
```
array([[0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.16, 0.84],
       [0.  , 1.  ],
       [0.04, 0.96],
       [0.  , 1.  ],
       [0.12, 0.88],
       [0.16, 0.84],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.04, 0.96],
       [0.  , 1.  ],
       [0.52, 0.48],
       [0.  , 1.  ],
       [0.76, 0.24],
       [0.2 , 0.8 ],
       [0.64, 0.36],
       [0.52, 0.48],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [0.52, 0.48],
       [0.  , 1.  ],
       [0.  , 1.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [0.88, 0.12],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [0.88, 0.12],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [1.  , 0.  ],
       [0.96, 0.04],
       [0.76, 0.24],
       [1.  , 0.  ],
       [0.6 , 0.4 ],
       [0.68, 0.32],
       [1.  , 0.  ],
       [0.88, 0.12],
       [1.  , 0.  ],
```

```
[1.  , 0.  ],
[0.56, 0.44],
[1.  , 0.  ],
[1.  , 0.  ],
[0.68, 0.32]])
```

In [62]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.metrics import plot_confusion_matrix,precision_score,recall_score,cc
```

In [2]:
```
fruits = pd.read_excel('c:/data/fruits.xlsx')
X = fruits[['Sphericity','Weight']]
y = fruits['labels']
```

In [3]:
```
X_train,X_test,y_train,y_test  = train_test_split(X,y,random_state=10)
```

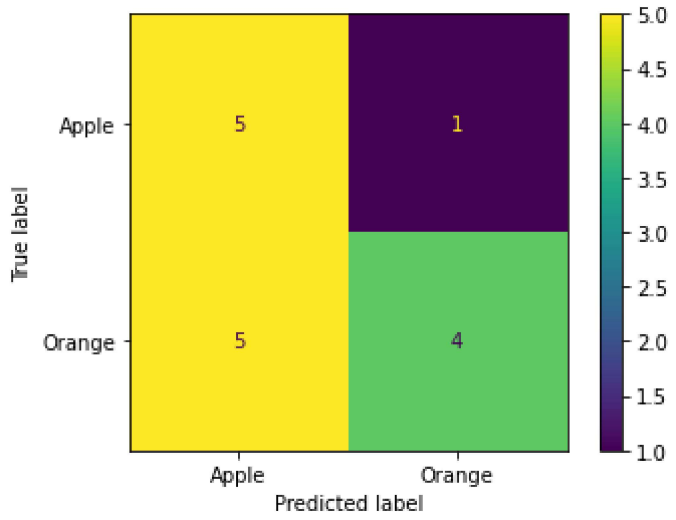
In [5]:
```
y_train.value_counts()
```

Out[5]:
```
Apple     24
Orange    21
Name: labels, dtype: int64
```

In [43]:
```
clf = DummyClassifier(strategy='stratified')
clf.fit(X_train,y_train)
print(clf.score(X_test,y_test))
y_predict = clf.predict(X_test)
```

```
0.4
```

In [50]:
```
_ = plot_confusion_matrix(clf,X_test,y_test)
```



In [18]:
```
y_predict
```

Out[18]:
```
array(['Apple', 'Orange', 'Apple', 'Apple', 'Apple', 'Orange', 'Apple',
       'Apple', 'Orange', 'Orange', 'Apple', 'Orange', 'Orange', 'Orange',
       'Orange'], dtype=object)
```

In [10]:
```
precision_score(y_test,y_predict,pos_label='Orange'),recall_score(y_test,y_predic
```

```
C:\Users\VAMSI KRISHNA\anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1248: UndefinedMetricWarning: Precision is ill-defined and being set to
0.0 due to no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[10]:
```
(0.0, 0.0)
```

In [135]:
```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train,y_train)
y_predict = clf.predict(X_test)
clf.score(X_test,y_test)
```

Out[135]:
```
0.8
```

In [136]:
```
knn_probs = clf.predict_proba(X_test)
knn_probs
```

Out[136]:
```
array([[1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [0.6, 0.4],
       [0.8, 0.2],
       [0. , 1. ],
       [1. , 0. ],
       [0. , 1. ],
       [0.8, 0.2],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [0. , 1. ]])
```

In [137]:
```
knn_results = pd.DataFrame({"actual_labels":y_test,
                            "predicted_labels":y_predict,
                            "prob_apple":knn_probs[:,0],
                            "prob_orange":knn_probs[:,1]})
knn_results
```

Out[137]:

	actual_labels	predicted_labels	prob_apple	prob_orange
31	Apple	Apple	1.0	0.0
3	Orange	Orange	0.0	1.0
38	Apple	Apple	1.0	0.0
27	Orange	Apple	0.6	0.4
21	Orange	Apple	0.8	0.2
17	Orange	Orange	0.0	1.0
46	Apple	Apple	1.0	0.0
2	Orange	Orange	0.0	1.0
23	Orange	Apple	0.8	0.2
26	Orange	Orange	0.0	1.0
35	Apple	Apple	1.0	0.0
39	Apple	Apple	1.0	0.0
20	Orange	Orange	0.0	1.0
37	Apple	Apple	1.0	0.0
7	Orange	Orange	0.0	1.0

In [138]:
```
knn_results[knn_results.actual_labels!=knn_results.predicted_labels]
```

Out[138]:

	actual_labels	predicted_labels	prob_apple	prob_orange
27	Orange	Apple	0.6	0.4
21	Orange	Apple	0.8	0.2
23	Orange	Apple	0.8	0.2

In [149]:
```
y_predict = np.array(['Apple' if i>0.8 else 'Orange' for i in knn_probs[:,0]])
print(y_predict)
clf.predict(X_test)
```

```
['Apple' 'Orange' 'Apple' 'Orange' 'Orange' 'Orange' 'Apple' 'Orange'
 'Orange' 'Orange' 'Apple' 'Apple' 'Orange' 'Apple' 'Orange']
```

Out[149]:
```
array(['Apple', 'Orange', 'Apple', 'Apple', 'Apple', 'Orange', 'Apple',
       'Orange', 'Apple', 'Orange', 'Apple', 'Apple', 'Orange', 'Apple',
       'Orange'], dtype=object)
```

In [145]:
```
accuracy_score(y_test,y_predict)
```

Out[145]:
```
0.8
```

In [150]:
```
confusion_matrix(y_test,y_predict)
```

Out[150]:
```
array([[6, 0],
       [0, 9]], dtype=int64)
```

In [2]:
```
import pandas as pd
fruits = pd.read_excel('c:/data/fruits.xlsx')
```

In [3]:
```
X = fruits[['Sphericity','Weight']]
y = fruits['labels']
```

In [31]:
```
from sklearn.preprocessing import LabelEncoder
encoding = LabelEncoder()
y = encoding.fit_transform(y)
```

In [4]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier,plot_tree
```

In [87]:
```
rf = RandomForestClassifier(n_estimators=10,
                            criterion='entropy',
                            max_depth=2,
                            bootstrap=True,
                            max_samples=30)
rf.fit(X,y)
rf.score(X,y)
```

Out[87]:
```
0.95
```

In [88]:
```
for model in rf.estimators_:
    print(model.score(X,y))
```

```
0.8666666666666667
0.9166666666666666
0.85
0.8666666666666667
0.9166666666666666
0.8833333333333333
0.9166666666666666
0.9166666666666666
0.8833333333333333
0.9333333333333333
```

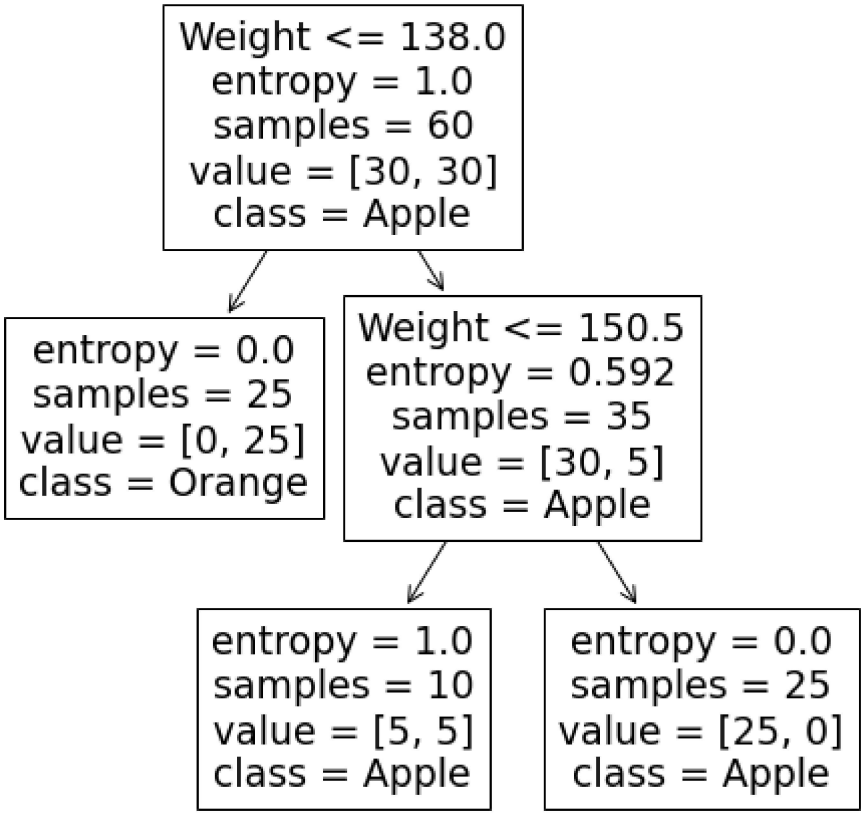
In [89]:
```
rf.estimators_
```

Out[89]:
```
[DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=1017983663),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=811008940),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=1709212023),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=1269136013),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=456183445),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=152287192),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=1352743546),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=666805439),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=2015561156),
 DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features='auto',
                        random_state=253486405)]
```

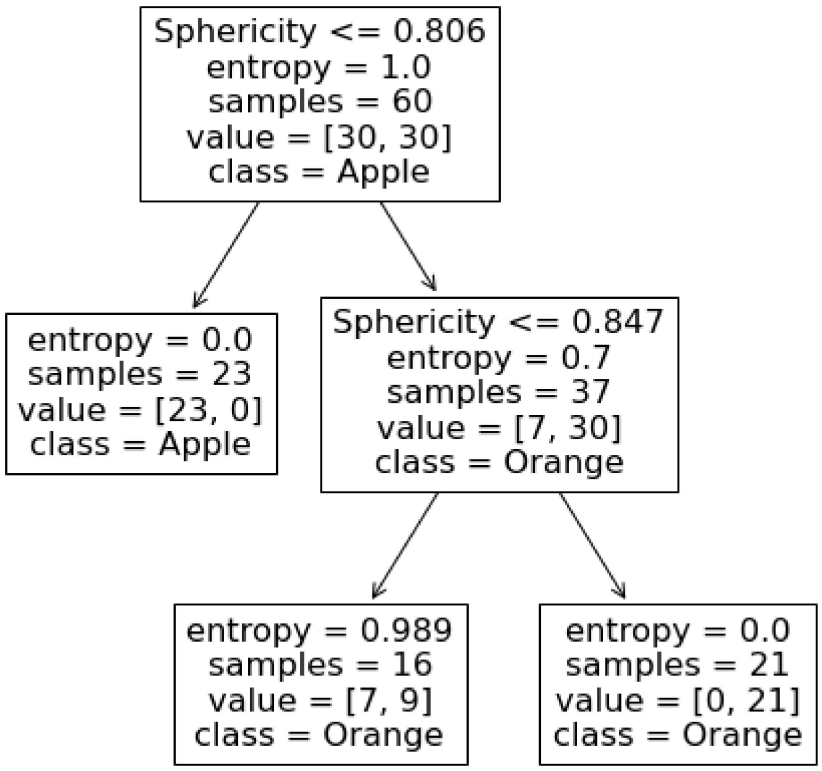
In [90]:
```
rf.n_estimators
```

Out[90]:
```
10
```

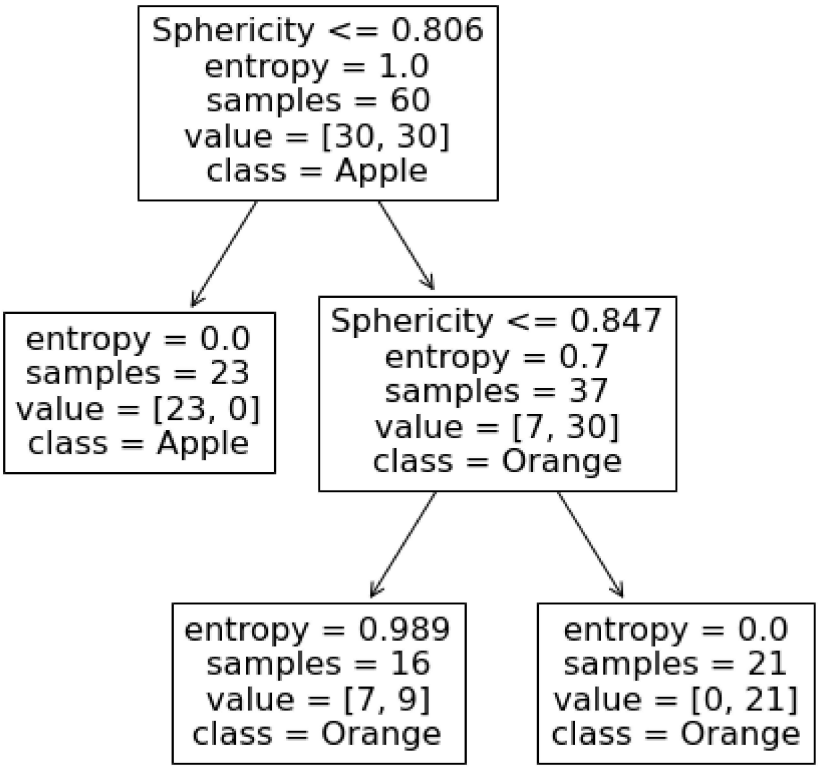
In [70]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=[8,8])
_ = plot_tree(rf.estimators_[0],feature_names=['Sphericity','Weight'],
              class_names=['Apple','Orange'])
```



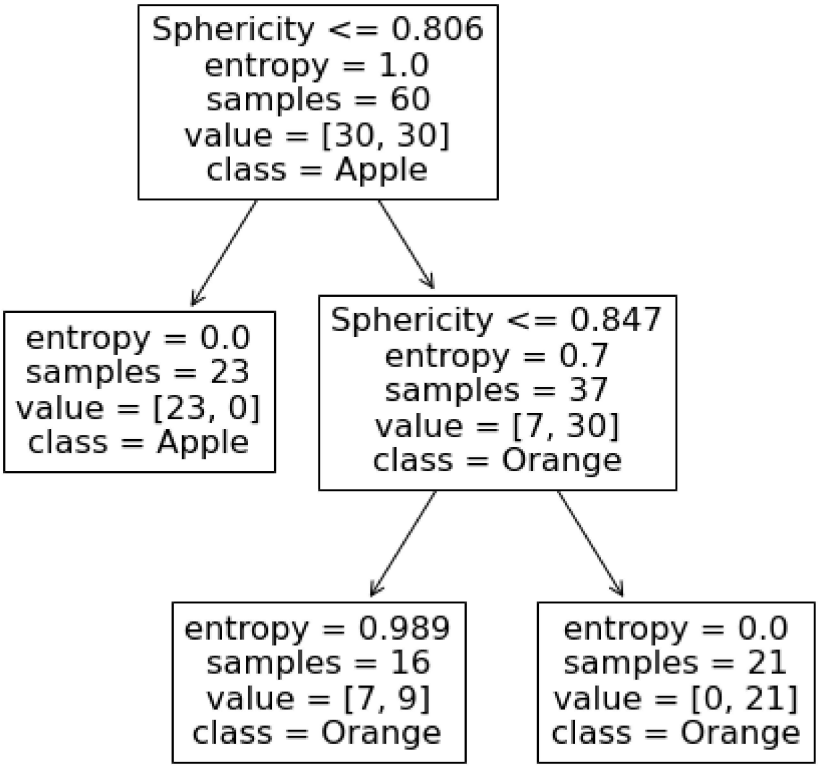
In [74]:
```
plt.figure(figsize=[8,8])
_ = plot_tree(rf.estimators_[1],feature_names=['Sphericity','Weight'],
              class_names=['Apple','Orange'])
```



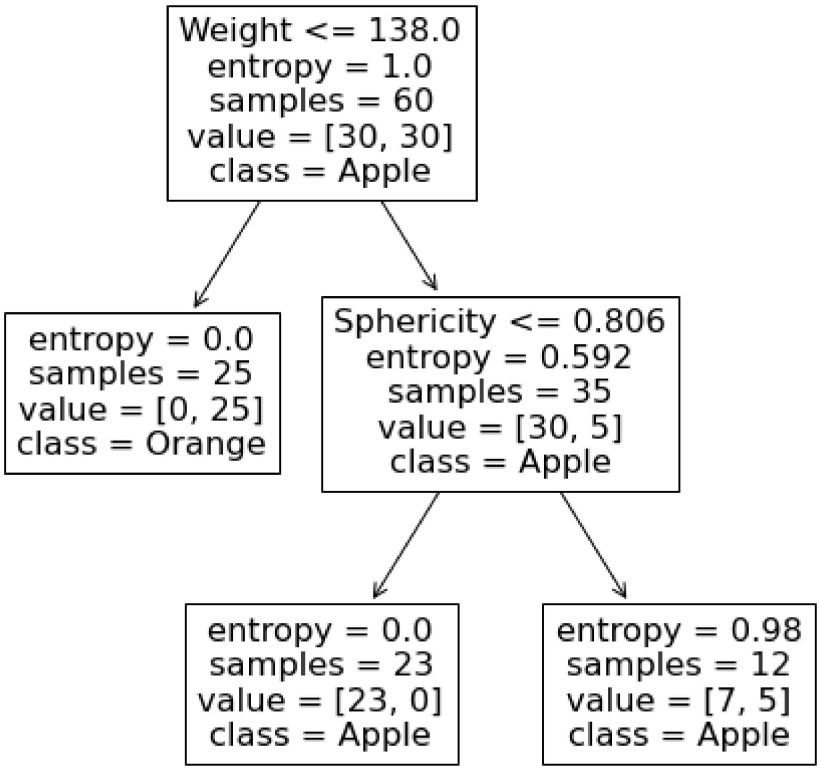
In [75]:
```
plt.figure(figsize=[8,8])
_ = plot_tree(rf.estimators_[2],feature_names=['Sphericity','Weight'],
              class_names=['Apple','Orange'])
```



In [76]:
```
plt.figure(figsize=[8,8])
_ = plot_tree(rf.estimators_[3],feature_names=['Sphericity','Weight'],
              class_names=['Apple','Orange'])
```



In [77]:
```
plt.figure(figsize=[8,8])
_ = plot_tree(rf.estimators_[4],feature_names=['Sphericity','Weight'],
              class_names=['Apple','Orange'])
```



In [58]:
```
from sklearn.metrics import plot_confusion_matrix
_ = plot_confusion_matrix(rf.estimators_[6],X,y)
```