Step 1: Import the dataset and split it into training and test parts

In [1]:
```
from sklearn.datasets import load_breast_cancer

dataset = load_breast_cancer()
print(dataset.feature_names, len(dataset.feature_names))
print(dataset.target_names, len(dataset.target_names))
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension'] 30
['malignant' 'benign'] 2
```

In [2]:
```
x = dataset.data
y = dataset.target
print(x.shape, y.shape)
print(x[0:6])
```

```
(569, 30) (569,)
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]
 [1.245e+01 1.570e+01 8.257e+01 4.771e+02 1.278e-01 1.700e-01 1.578e-01
  8.089e-02 2.087e-01 7.613e-02 3.345e-01 8.902e-01 2.217e+00 2.719e+01
  7.510e-03 3.345e-02 3.672e-02 1.137e-02 2.165e-02 5.082e-03 1.547e+01
  2.375e+01 1.034e+02 7.416e+02 1.791e-01 5.249e-01 5.355e-01 1.741e-01
  3.985e-01 1.244e-01]]
```

In [3]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

Step 2: Build the tree

Import the class *DecisionTreeClassifier* from sklearn.tree module.

We can use parameter *croterion* to set the metric for the split. Possible metrics are 'gini' (default), 'entropy' or 'log_loss'.

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

tree = DecisionTreeClassifier(criterion="entropy")
tree.fit(x_train, y_train)
```

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

Step 3: Evaluate the performance of the tree

In [5]:
```
from sklearn.metrics import accuracy_score
y_pred = tree.predict(x_test)
print("Training Accuracy:", accuracy_score(y_train,tree.predict(x_train)))
print("Test Accuracy:", accuracy_score(y_test, y_pred))
```

```
Training Accuracy: 1.0
Test Accuracy: 0.9122807017543859
```

Step 4: Visualize the tree

Visual tree can be denerated using *sklearn.tree.export_graphviz(tree, out_file, class_names, feature_names, impurity, filled)* function. Which return a .dot file with the reqired graph.

In [6]:
```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file="tree.dot", class_names=dataset.target_names, f
```

Install graphviz using *conda install python-graphviz* to view the .dot file.

In [7]:
```
import graphviz
with open("tree.dot") as f:
    graph = f.read()
graphviz.Source(graph)
```

Out[7]:

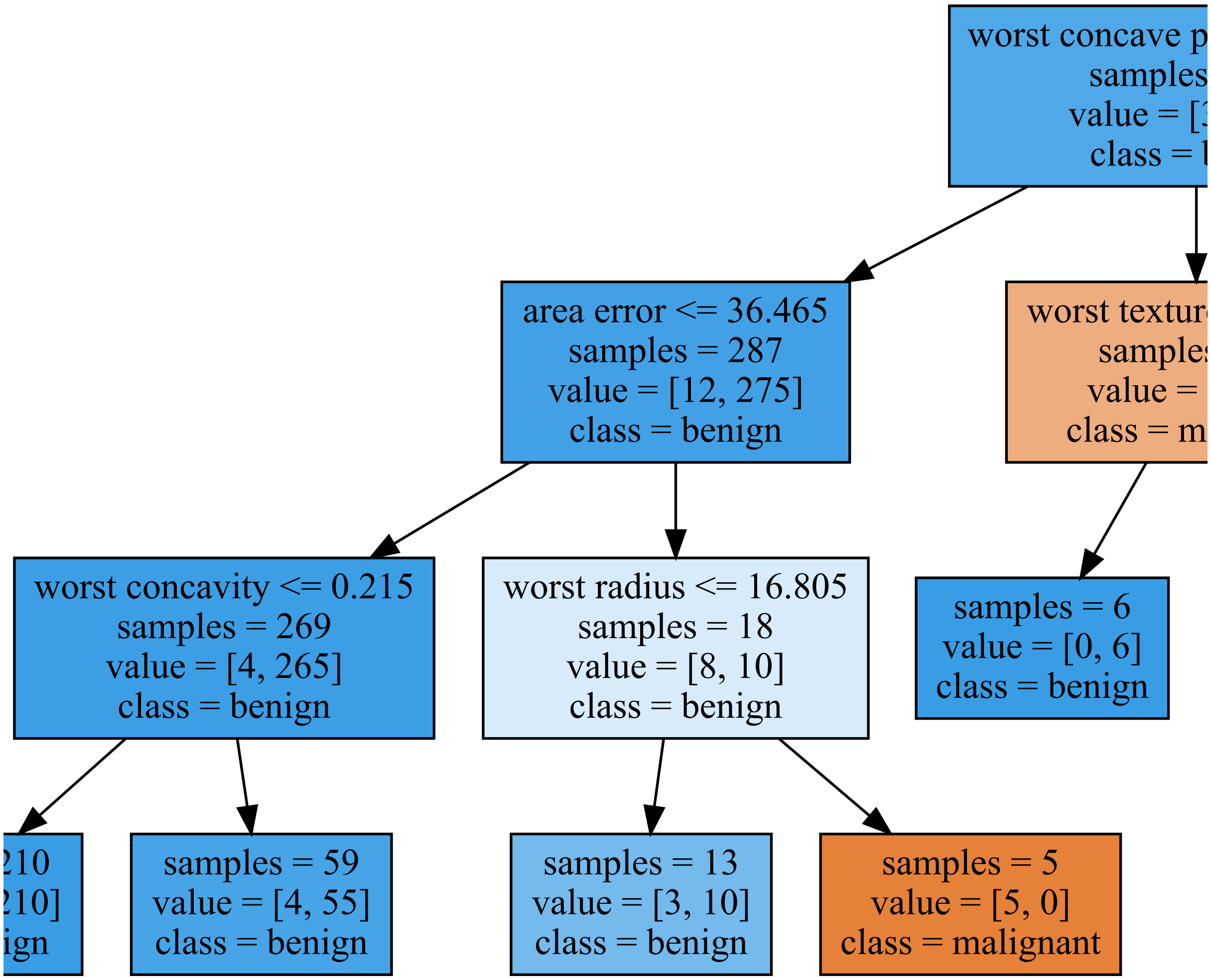
Step 5: Handle the overfitting by limiting the depth

Use *max_depth* parameter with DecisionTreeClassifier() to limit the depth

In [8]:
```
tree1 = DecisionTreeClassifier(criterion="entropy",max_depth=4)
tree1.fit(x_train, y_train)
print("Training Accuracy:", accuracy_score(y_train,tree1.predict(x_train)))
print("Test Accuracy:", accuracy_score(y_test, tree1.predict(x_test)))
export_graphviz(tree1, out_file="tree1.dot", class_names=dataset.target_names,
with open("tree1.dot") as f:
    graph = f.read()
graphviz.Source(graph)
```

```
Training Accuracy: 0.9846153846153847
Test Accuracy: 0.9298245614035088
```

Out[8]:


worst concave p
samples
value = [3
class = b
area error <= 36.465
samples = 287
value = [12, 275]
class = benign
worst texture
samples
value =
class = m
worst concavity <= 0.215
samples = 269
value = [4, 265]
class = benign
worst radius <= 16.805
samples = 18
value = [8, 10]
class = benign
samples = 6
value = [0, 6]
class = benign
210
210]
ign
samples = 59
value = [4, 55]
class = benign
samples = 13
value = [3, 10]
class = benign
samples = 5
value = [5, 0]
class = malignant

In [ ]: