In [ ]:
```
URK21CS1181
ISHAN CHASKAR
```

In [ ]:
```
Aim: Develop a Decision Tree classification model for the Cancer dataset using
the scikit-learn
```

In [ ]:
```
Description:
    A Decision Tree is a supervised machine learning algorithm used for both
    classification and regression tasks. It works by recursively partitioning
    the dataset into subsets based on the values of input features.
    At each step, the algorithm selects the feature that best separates or
    distinguishes the data points according to a certain criterion. This
    process is repeated until a stopping condition is met, resulting in a
    tree-like structure where each internal node represents a decision based
    on a feature, each branch represents the outcome of the decision, and
    each leaf node represents
    the final predicted class or value.

The main components and concepts related to Decision Trees:

Root Node:
The topmost node in the tree, representing the entire dataset.
It is split into child nodes based on the feature that best
separates the data

Internal Nodes:
Nodes that are not leaf nodes. Each internal node represent
s a decision based on a specific feature and threshold. The
decision determines which branch to follow down the tree.

Leaves:
Terminal nodes at the bottom of the tree. Each leaf
represents the predicted outcome (class or value) for the
subset of data that reaches that particular leaf.

Features:
The attributes or variables used to make decisions at each
node. The  algorithm selects the best feature at each step
to maximize the information gain or another specified criterion.

Thresholds:
Values used to split the data at each node. The decision
tree algorithm chooses the threshold that optimally separates
the data into subsets.

Criterion:The metric used to measure the quality of a split.
Common criteria include entropy and Gini impurity for classification
tasks and mean squared error for regression tasks.

Pruning:
The process of removing unnecessary branches or nodes from the
tree to prevent overfitting. Pruning is often controlled by
hyperparameters like the maximum depth of the tree.
```

In [ ]:
```
1. Develop a Decision Tree classification model for the Cancer dataset using
the scikit-learn
```

a. Use the columns: 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean','smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean' as the independent variables

In [16]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import matplotlib.pyplot as plt
data = pd.read_csv('Cancer.csv')
X = data[['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
          'smoothness_mean', 'compactness_mean', 'concavity_mean',
          'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean']]
```

In [ ]:
```
b. Use the target variable as 'diagnosis' (Malignant - M, Benign - B)
```

In [17]:
```
y = data['diagnosis'].map({'M': 1, 'B': 0})
```

In [ ]:
```
c. Encode the categorical value of the target column to numerical value
```

In [ ]:
```
le = LabelEncoder()
data['diagnosis'] = le.fit_transform(data['diagnosis'])
```

In [ ]:
```
d. Divide the data into training (75%) and testing set (25%)
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    random_state=42)
```

In [ ]:
```
e. Perform the classification with entropy and information gain as decision criteria in decision tree
```

In [20]:
```
def evaluate_classifier(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f_score = f1_score(y_true, y_pred)

    fpr, tpr, thresholds = roc_curve(y_true, y_pred)
    auc_score = auc(fpr, tpr)

    return cm, accuracy, precision, recall, f_score, auc_score
```

In [ ]:
```
f. Analyse the performance of the classifier with various performance measures such as confusion matrix, accuracy, recall, precision, specificity, f-score, Receiver operating characteristic (ROC) curve and Area Under Curve (AUC) score.
```

In [21]:
```
print("URK21CS1181")
def train_and_evaluate_decision_tree(criterion, max_depth=None):
    clf=DecisionTreeClassifier(criterion=criterion,max_depth=
                               max_depth,random_state=42)
    clf.fit(X_train, y_train)
```

```
    y_pred = clf.predict(X_test)

    cm, accuracy, precision, recall, f_score, auc_score =
    evaluate_classifier(y_test, y_pred)

    return cm, accuracy, precision, recall, f_score, auc_score

clf = DecisionTreeClassifier(criterion='entropy',
                             random_state=42)
clf.fit(X_train, y_train)
```

```
URK21CS1181
```

Out[21]:

DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

In [ ]:

```
g. Display the constructed decision tree sklearn.tree.plot_tree
method
```

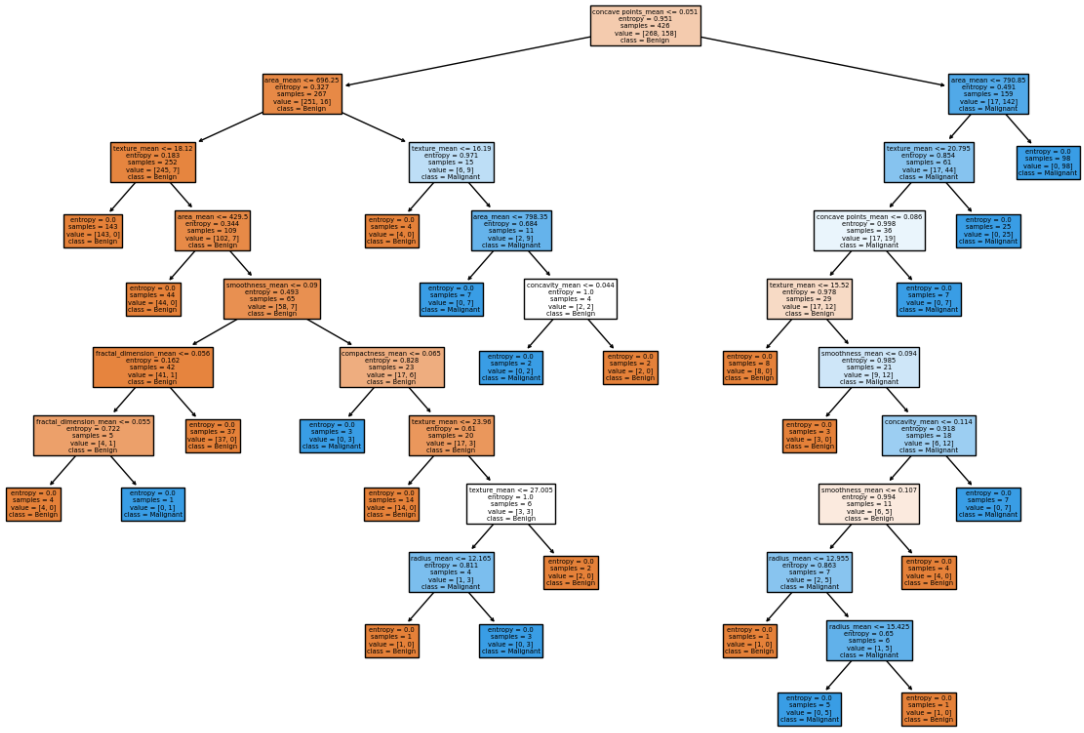
In [22]:

```
print("URK21CS1181")
plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True, feature_names=list(X.columns),
          class_names=['Benign', 'Malignant'])
plt.show()

results = pd.DataFrame(columns=['Max Depth', 'TP', 'TN', 'Accuracy',
                                'Precision', 'Recall', 'F-score',
                                'AUC Score'])

cm, accuracy, precision, recall, f_score, auc_score =
train_and_evaluate_decision_tree(criterion='entropy')
results = results.append({'Max Depth': 'Default', 'TP': cm[1, 1],
                          'TN': cm[0, 0],
                          'Accuracy': accuracy, 'Precision': precision,
                          'Recall': recall, 'F-score': f_score,
                          'AUC Score': auc_score}, ignore_index=True)
```

```
URK21CS1181
```



```
/tmp/ipykernel_2077082/2369501180.py:10: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pandas.c
oncat instead.
  results = results.append({'Max Depth': 'Default', 'TP': cm[1, 1],
```

h. Prune the tree with maximum depth as 3,5,7 and tabulate the various TP, TN, accuracy, f-score and AUC score obtained

In [23]:
```
print("URK21CS1181")
cm, accuracy, precision, recall, f_score, auc_score =
train_and_evaluate_decision_tree(criterion='entropy', max_depth=3)
results = results.append({'Max Depth': 3, 'TP': cm[1, 1],
                          'TN': cm[0, 0],
                          'Accuracy': accuracy,
                          'Precision': precision,
                          'Recall': recall, 'F-score':
                          f_score,
                          'AUC Score': auc_score},
                         ignore_index=True)

cm, accuracy, precision, recall, f_score, auc_score =
train_and_evaluate_decision_tree(criterion='entropy', max_depth=5)
results = results.append({'Max Depth': 5, 'TP': cm[1, 1],
                          'TN': cm[0, 0],
                          'Accuracy': accuracy,
                          'Precision': precision,
                          'Recall': recall, 'F-score':
                          f_score,
                          'AUC Score': auc_score},
                         ignore_index=True)

cm, accuracy, precision, recall, f_score, auc_score =
train_and_evaluate_decision_tree(criterion='entropy', max_depth=7)
results = results.append({'Max Depth': 7, 'TP': cm[1, 1],
                          'TN': cm[0, 0],
                          'Accuracy': accuracy,
```

```
                          'Precision': precision,
                          'Recall': recall, 'F-score':
                          f_score,
                          'AUC Score': auc_score},
                         ignore_index=True)

print(results)
```

```
URK21CS1181
  Max Depth    TP    TN  Accuracy  Precision    Recall   F-score  AUC Score
0   Default    48    83  0.916084   0.888889  0.888889  0.888889   0.910737
1       3.0  52.0  78.0  0.909091   0.825397  0.962963  0.888889   0.919684
2       5.0  51.0  84.0  0.944056   0.910714  0.944444  0.927273   0.944132
3       7.0  49.0  85.0  0.937063   0.924528  0.907407  0.915888   0.931232
```

```
/tmp/ipykernel_2077082/734511199.py:3: FutureWarning: The frame.append method i
s deprecated and will be removed from pandas in a future version. Use pandas.co
ncat instead.
  results = results.append({'Max Depth': 3, 'TP': cm[1, 1], 'TN': cm[0, 0],
/tmp/ipykernel_2077082/734511199.py:9: FutureWarning: The frame.append method i
s deprecated and will be removed from pandas in a future version. Use pandas.co
ncat instead.
  results = results.append({'Max Depth': 5, 'TP': cm[1, 1], 'TN': cm[0, 0],
/tmp/ipykernel_2077082/734511199.py:15: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pandas.c
oncat instead.
  results = results.append({'Max Depth': 7, 'TP': cm[1, 1], 'TN': cm[0, 0],
```

In [ ]:
```
Result:Hence we got the desired output without any error
```