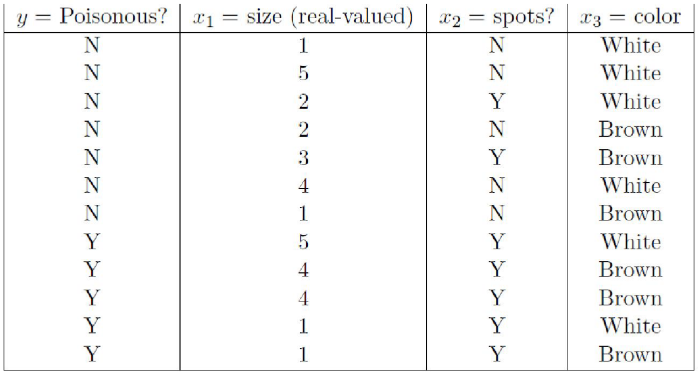
Boosting Decision Trees & Random Forest

Part 1. Determining Splits

sample dataset of mushrooms and it's attributes the intention is to classify them as poisonous or not. Build a decision tree by hand show all your calculations in either pen or notepad to classify



y = Poisonous?	x_1 = size (real-valued)	x_2 = spots?	x_3 = color
N	1	N	White
N	5	N	White
N	2	Y	White
N	2	N	Brown
N	3	Y	Brown
N	4	N	White
N	1	N	Brown
Y	5	Y	White
Y	4	Y	Brown
Y	4	Y	Brown
Y	1	Y	White
Y	1	Y	Brown

1. What is the entropy of the target variable, "poisonous"?

   Answer: Poisonous: samples = 12, values = [5, 7]

   -> **entropy** = -5/12*log$_2$(5/12) -7/12* log$_2$(7/12) = 0.98

2. What is the first attribute a decision tree trained using the entropy and the information gain method

   Answer: The first attribute the decision tree using is 'spots' ['Y', 'N']

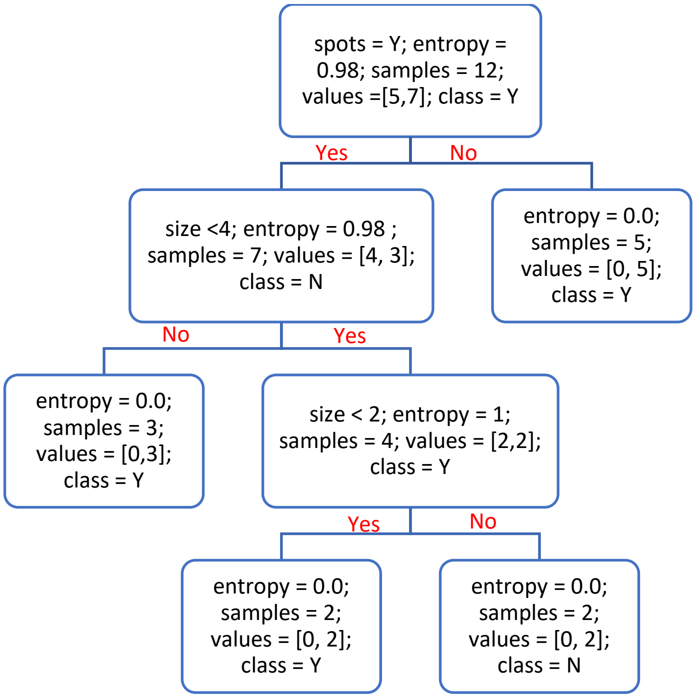
   The information gain method is entropy: 'Y': samples = 7, values = [5, 2], **entropy** = 0.86

   'N': samples = 5, values = [0, 5], **entropy** = 0.00

3. What is the information gain of this attribute?

   Answer: The information gain of this attribute = 0.98 – (0.86 + 0.00) = 0.12

4. Draw the full decision tree learned from this data set .


spots = Y; entropy = 0.98; samples = 12; values =[5,7]; class = Y
Yes
No
size <4; entropy = 0.98 ; samples = 7; values = [4, 3]; class = N
entropy = 0.0; samples = 5; values = [0, 5]; class = Y
No
Yes
entropy = 0.0; samples = 3; values = [0,3]; class = Y
size < 2; entropy = 1; samples = 4; values = [2,2]; class = Y
Yes
No
entropy = 0.0; samples = 2; values = [0, 2]; class = Y
entropy = 0.0; samples = 2; values = [0, 2]; class = N

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import graphviz
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
```

Part 2. Decision tree using python

In [2]:

```
wbc = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data',
                  header=None)
```

In [3]:

```
wbc.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30	
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07

5 rows × 32 columns

In [4]:

```
X, y = wbc.iloc[:, 2:], wbc.iloc[:, 1]
X.shape, y.shape
```

Out[4]:

```
((569, 30), (569,))
```

In [5]:

```
# 1) Divide the data into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=1,
                                                    stratify=y,
                                                    test_size=0.2)
```

In [6]:

```
# 2) train the Decision Tree Classifier using all the features of the data and test your model on the test data
crts = ["gini", "entropy"]
for crt in crts:
    tree_rotat = DecisionTreeClassifier(criterion=crt, random_state=1)
    tree_rotat.fit(X_train, y_train)
    y_pred = tree_rotat.predict(X_test)
    print(f'Accuracy for Decision Tree with criteria as {crt} Index is: {accuracy_score(y_test,y_pred)*100}')
```

```
Accuracy for Decision Tree with criteria as gini Index is: 92.98245614035088
Accuracy for Decision Tree with criteria as entropy Index is: 92.98245614035088
```

In [7]:

```
# 3) Use the Grid Search method to run the model for trees of depth 1, 2, 3, 4, 5, and 6 and for the
# Gini Impurity and Entropy impurity measures
para = {'criterion':["gini", "entropy"], 'max_depth':[1, 2, 3, 4, 5, 6]}
clf=GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid=para, cv=5)
clf.fit(X_train, y_train)

result_df = pd.DataFrame(clf.cv_results_)
result_df.iloc[:, [4, 5, 12, 13, 14]]
```

Out[7]:

	param_criterion	param_max_depth	mean_test_score	std_test_score	rank_test_score
0	gini	1	0.881319	0.008223	11
1	gini	2	0.931868	0.022413	7
2	gini	3	0.929670	0.017855	9
3	gini	4	0.940659	0.028317	2
4	gini	5	0.938462	0.036513	3

	param_criterion	param_max_depth	mean_test_score	std_test_score	rank_test_score
5	gini	6	0.936264	0.036380	5
6	entropy	1	0.876923	0.008223	12
7	entropy	2	0.894505	0.039682	10
8	entropy	3	0.938462	0.016447	3
9	entropy	4	0.947253	0.020143	1
10	entropy	5	0.931868	0.032894	7
11	entropy	6	0.936264	0.026374	5

In [8]:

```python
# 4) Determine the best model use the plot.tree() method to visualize it.
# Answer: best model is 'entropy' with max_depth=4

# train the model
tree_entropy_bc = DecisionTreeClassifier(criterion = "entropy", random_state = 1,
                                         max_depth=4)
tree_entropy_bc.fit(X_train, y_train)
y_pred_entropy_bc = tree_entropy_bc.predict(X_test)
y_pred_entropy_bc

print(f'The accuracy is: {accuracy_score(y_test, y_pred_entropy_bc)*100}')
```

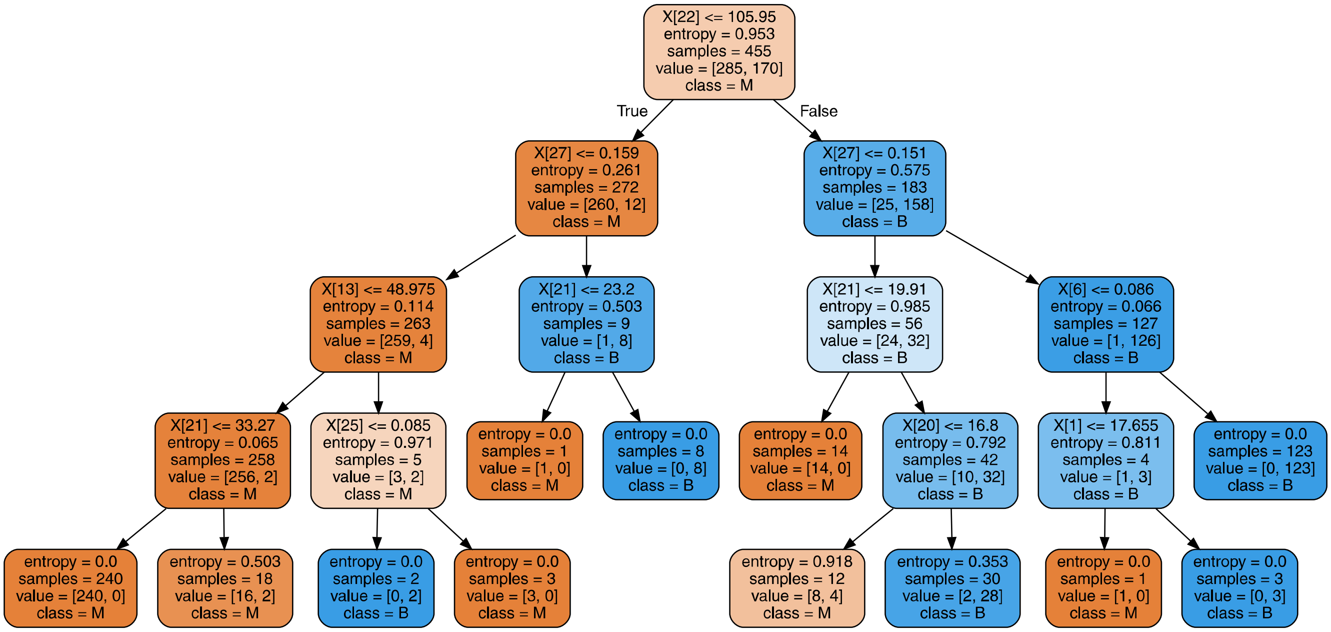
```
The accuracy is: 94.73684210526315
```

In [9]:

```python
# plot the tree
# feature_names = X.columns
class_names_entropy_bc = y.unique()

# Visualizing the Entropy based Decision Tree
graph_data_entropy_bc = tree.export_graphviz(tree_entropy_bc, out_file=None, filled=True, rounded=True,
                                  class_names=class_names_entropy_bc)
graph_bc = graphviz.Source(graph_data_entropy_bc)
graph_bc
```

Out[9]:



In [10]:

```python
# 5) Use Adaboost to improve the model and evaluate the performance using the test set
tree_entropy_2 = AdaBoostClassifier(
                                    DecisionTreeClassifier(criterion = "entropy",max_depth=4, random_state=1),
                                    n_estimators=300,random_state=1)
tree_entropy_2.fit(X_train, y_train)
```

Out[10]:

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(criterion='entropy',
                                                         max_depth=4,
                                                         random_state=1),
                   n_estimators=300, random_state=1)
```

In [11]:

```python
# 6) What is the accuracy?
# predict & accuracy
y_pred_entropy_2 = tree_entropy_2.predict(X_test)
```

```
y_pred_entropy_2
print(f'The accuracy is: {accuracy_score(y_test, y_pred_entropy_2)*100}')
```

```
The accuracy is: 97.36842105263158
```

Part 3: Random Forest using Python

1) Use the same training and test set as above. 2) Use the RandomForest classifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) to create a model.

In [12]:

```
forest_entroy_bc = RandomForestClassifier(random_state=1)
forest_entroy_bc.fit(X_train, y_train)
y_pred_forest_bc = forest_entroy_bc.predict(X_test)
print(f'The accuracy of random forest model is: {accuracy_score(y_test, y_pred_forest_bc)*100}')
```

```
The accuracy of random forest model is: 95.6140350877193
```

3) Compare the parameters that are provided for the Random Forest classifier and Decision Tree classifier. How many are the same and how many are different?

- same parameters: (14 parameters in total)
  - *,
  - criterion='gini',
  - splitter='best',
  - max_depth=None,
  - min_samples_split=2,
  - min_samples_leaf=1,
  - min_weight_fraction_leaf=0.0,
  - max_features=None,
  - random_state=None,
  - max_leaf_nodes=None,
  - min_impurity_decrease=0.0,
  - min_impurity_split=None,
  - class_weight=None,
  - ccp_alpha=0.0

- Different parameters (7 parameters -- RandomForestClassifier is provided with the following additinal parameters)
  - n_estimators=100,
  - bootstrap=True,
  - oob_score=False,
  - n_jobs=None,
  - verbose=0,
  - warm_start=False,
  - max_samples=None.

4) Use the Grid Search method to run the model for trees of depth 1, 2, 3, 4, 5, and 6 and for the Gini Impurity and Entropy impurity measures. Also set the parameter so it will use the "outof-bag" samples for calculating accuracy.

In [13]:

```
para = {'criterion':["gini", "entropy"], 'max_depth':[1, 2, 3, 4, 5, 6]}
clf_forest=GridSearchCV(RandomForestClassifier(oob_score=True, random_state=1), param_grid=para, cv=5)
clf_forest.fit(X_train, y_train)

result_df_forest = pd.DataFrame(clf_forest.cv_results_)
result_df_forest.iloc[:, [4, 5, 12, 13, 14]]
```

Out[13]:

	param_criterion	param_max_depth	mean_test_score	std_test_score	rank_test_score
0	gini	1	0.907692	0.025631	11
1	gini	2	0.947253	0.010767	9
2	gini	3	0.949451	0.008791	8
3	gini	4	0.953846	0.008223	5
4	gini	5	0.956044	0.009829	3
5	gini	6	0.953846	0.014579	5
6	entropy	1	0.907692	0.024670	11
7	entropy	2	0.931868	0.017582	10
8	entropy	3	0.951648	0.008791	7
9	entropy	4	0.958242	0.017582	1

	param_criterion	param_max_depth	mean_test_score	std_test_score	rank_test_score
10	entropy	5	0.958242	0.012815	1
11	entropy	6	0.956044	0.013900	3

best model: entropy with max_depth=4

5) Test the accuracy of RandomForest using the Test set.

6) Compare the performance of Decision Tree with Boost and Random Forest.

In [14]:

```
forest_entropy = RandomForestClassifier(criterion='entropy',
                                oob_score=True,
                                random_state=1,
                                max_depth=4)
forest_entropy.fit(X_train,y_train)
y_pred_f_entropy = forest_entropy.predict(X_test)

print(f'The accuracy is: {accuracy_score(y_test, y_pred_f_entropy)*100}')
```

```
The accuracy is: 94.73684210526315
```

The performance

- RandomTreeDecision with boost **97.37**
- RandomForestClassifier (gini & entropy) **94.73**

random tree decision with boost performed better.

In [ ]:

```

```