CS 6220 Data Mining — Assignment 8 — Decision Trees — Samuel Steiner

In [ ]:
```python
# Import packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import graphviz
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

In [ ]:
```python
# load data
X, y = load_boston(return_X_y=True)

# Split the range of target values into three equal parts - low, mid, and high.
# Reassign the target values into into three categorical values 0, 1, and 2, representing low, mid and high range of values, respectively
diff = (y.max() - y.min())/3
d = y.min()
split = []
for _ in range(3):
    split.append([d, d+diff])
    d += diff

def categorize_target(value):
    for idx, rangx in enumerate(split):
        if value >= rangx[0] and value <= rangx[1]:
            return idx

y_ = np.array(list(map(categorize_target, y)))
```

In [ ]:
```python
# 1. Split the dataset into 70% training set and 30% test set.
X_train, X_test, y_train, y_test = train_test_split(X, y_, test_size=0.3)
```

In [ ]:
```python
# 2. Using scikit-learn's DecisionTreeClassifier, train a supervised learning model that can be used to generate predictions for your data.
# A reference to how you can do that can be found in the users manual at
# https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.

dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
```

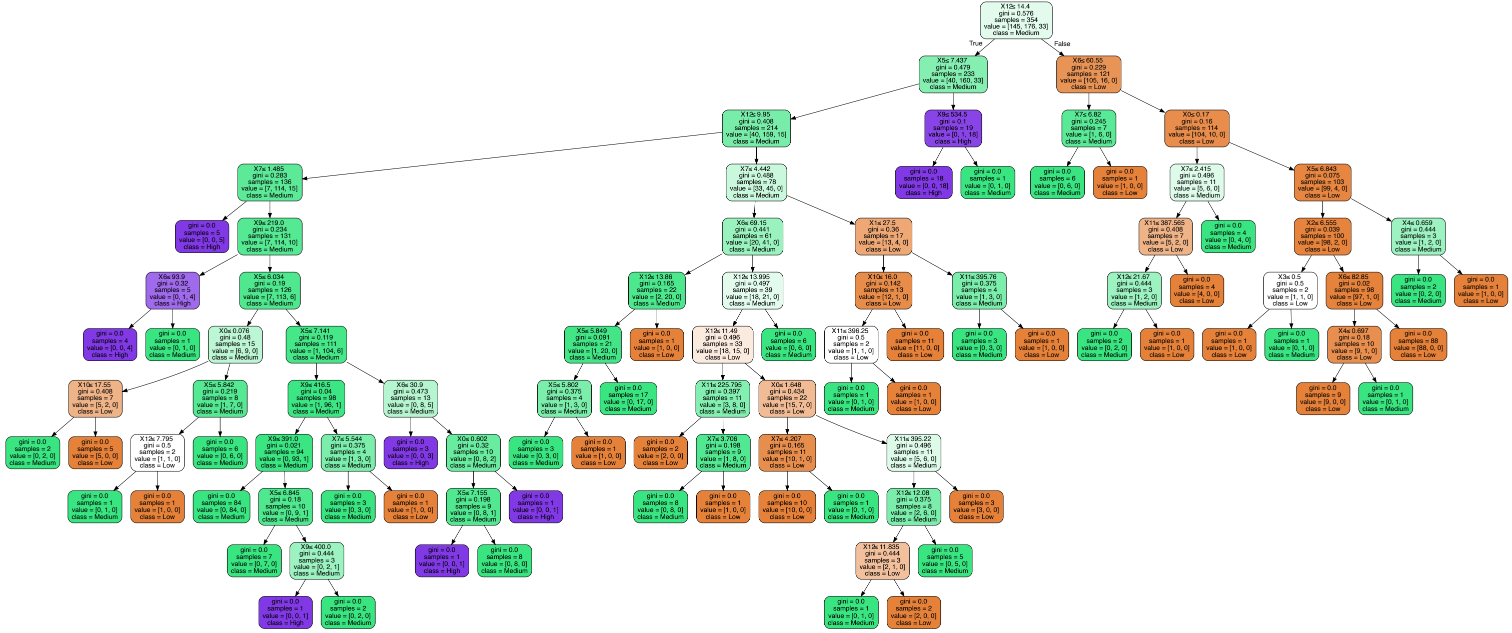
Out[ ]:
```
DecisionTreeClassifier()
```

In [ ]:
```python
# 3. Report the tree depth, number of leaves, feature importance, train score, and test score of the tree. Let the tree depth be Td.

td = dt_clf.get_depth()

def report_on_dt(dt_):
        print(f"""
        Tree Depth: {dt_.get_depth()}
        Number of Leaves: {dt_.get_n_leaves()}
        Feature importance: {' '.join([f'Feature {idx+1}: {val:.3f}' for idx, val in enumerate(dt_.feature_importances_)])}
        Train score: {dt_.score(X_train, y_train)}
        Test score: {dt_.score(X_test, y_test):.3f}""")

report_on_dt(dt_clf)
```

```
        Tree Depth: 11
        Number of Leaves: 52
        Feature importance: Feature 1: 0.056 Feature 2: 0.014 Feature 3: 0.005 Feature 4: 0.005 Feature 5: 0.015 Feature 6: 0.178 Feature 7: 0.080 Feature 8: 0.109 Feature 9: 0.000 Feature 10: 0.044 Feature 11: 0.018 Feature 12: 0.044 Feature 13: 0.432
        Train score: 1.0
        Test score: 0.796
```

In [ ]:
```python
# 4. Show the visual output of the decision tree.
dot_data = export_graphviz(dt_clf, class_names=['Low', 'Medium', 'High'],
                        filled=True, rounded=True,  special_characters=True)
graph = graphviz.Source(dot_data)
graph.render(f"boston_housing_{dt_clf.get_depth()}")
graph
```

Out[ ]:



In [ ]:

```python
# 5. Next, Generate (Td−1) decision trees on the same training set using fixed tree depths
# {1, 2, ...(T d −1)}. The tree depth can be set using max=d, where d is the depth of the tree.

# 6. For each of the (Td−1) trees report, tree depth, number of leaves, feature importance,
# train score, and test score of the tree.
best = 0
best_clf = None
for depth in range(1, td):
    dt_set_clf = DecisionTreeClassifier(max_depth=depth)
    dt_set_clf.fit(X_train, y_train)
    report_on_dt(dt_set_clf)
    score = dt_set_clf.score(X_test, y_test)
    if score > best:
        best = score
        best_clf = dt_set_clf
```

```
        Tree Depth: 1
        Number of Leaves: 2
        Feature importance: Feature 1: 0.000 Feature 2: 0.000 Feature 3: 0.000 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.000 Feature 7: 0.000 Feature 8: 0.000 Feature 9: 0.000 Feature 10: 0.000 Feature 11: 0.000 Feature 12: 0.000 Feature 13: 1.000
        Train score: 0.748587570621469
        Test score: 0.776

        Tree Depth: 2
        Number of Leaves: 4
        Feature importance: Feature 1: 0.000 Feature 2: 0.000 Feature 3: 0.000 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.236 Feature 7: 0.082 Feature 8: 0.000 Feature 9: 0.000 Feature 10: 0.000 Feature 11: 0.000 Feature 12: 0.000 Feature 13: 0.682
        Train score: 0.8107344632768362
        Test score: 0.849

        Tree Depth: 3
        Number of Leaves: 8
        Feature importance: Feature 1: 0.045 Feature 2: 0.000 Feature 3: 0.015 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.212 Feature 7: 0.068 Feature 8: 0.000 Feature 9: 0.000 Feature 10: 0.000 Feature 11: 0.000 Feature 12: 0.000 Feature 13: 0.660
        Train score: 0.8192090395480226
        Test score: 0.829

        Tree Depth: 4
        Number of Leaves: 12
        Feature importance: Feature 1: 0.052 Feature 2: 0.000 Feature 3: 0.000 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.187 Feature 7: 0.059 Feature 8: 0.117 Feature 9: 0.014 Feature 10: 0.000 Feature 11: 0.000 Feature 12: 0.000 Feature 13: 0.571
        Train score: 0.8700564971751412
        Test score: 0.829

        Tree Depth: 5
        Number of Leaves: 18
        Feature importance: Feature 1: 0.044 Feature 2: 0.030 Feature 3: 0.016 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.158 Feature 7: 0.079 Feature 8: 0.105 Feature 9: 0.010 Feature 10: 0.034 Feature 11: 0.000 Feature 12: 0.013 Feature 13: 0.511
        Train score: 0.8898305084745762
        Test score: 0.822

        Tree Depth: 6
        Number of Leaves: 27
        Feature importance: Feature 1: 0.031 Feature 2: 0.017 Feature 3: 0.015 Feature 4: 0.000 Feature 5: 0.018 Feature 6: 0.173 Feature 7: 0.083 Feature 8: 0.114 Feature 9: 0.009 Feature 10: 0.031 Feature 11: 0.014 Feature 12: 0.000 Feature 13: 0.494
        Train score: 0.9124293785310734
        Test score: 0.783

        Tree Depth: 7
        Number of Leaves: 33
        Feature importance: Feature 1: 0.044 Feature 2: 0.026 Feature 3: 0.014 Feature 4: 0.000 Feature 5: 0.020 Feature 6: 0.181 Feature 7: 0.085 Feature 8: 0.089 Feature 9: 0.011 Feature 10: 0.047 Feature 11: 0.005 Feature 12: 0.006 Feature 13: 0.473
        Train score: 0.940677966101695
        Test score: 0.776

        Tree Depth: 8
        Number of Leaves: 40
        Feature importance: Feature 1: 0.059 Feature 2: 0.015 Feature 3: 0.013 Feature 4: 0.000 Feature 5: 0.044 Feature 6: 0.169 Feature 7: 0.096 Feature 8: 0.082 Feature 9: 0.000 Feature 10: 0.039 Feature 11: 0.020 Feature 12: 0.024 Feature 13: 0.439
        Train score: 0.9661016949152542
        Test score: 0.783
```

```
        Tree Depth: 9
        Number of Leaves: 47
        Feature importance: Feature 1: 0.064 Feature 2: 0.014 Feature 3: 0.005 Feature 4: 0.000 Feature 5: 0.000 Feature 6: 0.172 Feature 7: 0.116 Feature 8: 
0.099 Feature 9: 0.013 Feature 10: 0.039 Feature 11: 0.026 Feature 12: 0.026 Feature 13: 0.426
        Train score: 0.9887005649717514
        Test score: 0.809

        Tree Depth: 10
        Number of Leaves: 50
        Feature importance: Feature 1: 0.073 Feature 2: 0.014 Feature 3: 0.005 Feature 4: 0.005 Feature 5: 0.009 Feature 6: 0.172 Feature 7: 0.099 Feature 8: 
0.099 Feature 9: 0.000 Feature 10: 0.049 Feature 11: 0.025 Feature 12: 0.039 Feature 13: 0.410
        Train score: 0.9943502824858758
        Test score: 0.803
```

In [ ]:

```
# 7. Show the visual output of the decision tree with highest test score from the (Td-1) trees.
dot_data = export_graphviz(best_clf, class_names=['Low', 'Medium', 'High'],
                           filled=True, rounded=True,  special_characters=True)
graph = graphviz.Source(dot_data)
graph.render(f"boston_housing_{best_clf.get_depth()}")
graph
```

Out[ ]: