1. Split the dataset into 70% training set and 30% test set.

In [ ]:

```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(’ignore’)

boston = load_boston()
X = boston.data
Y = boston.target
name_data = boston.feature_names

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3)

y_train = y_train.astype(’int’)
y_test = y_test.astype(’int’)
```

1. Using scikit-learn’s DecisionTreeClassifier, train a supervised learning model that can be used to generate predictions for your data

In [ ]:

```
dtc = DecisionTreeClassifier()
dtc = dtc.fit(x_train,y_train)
dtc_predict = dtc.predict(x_test)
```

1. Report the tree depth, number of leaves, feature importance, train score, and test score of the tree. Let the tree depth be Td.

In [ ]:

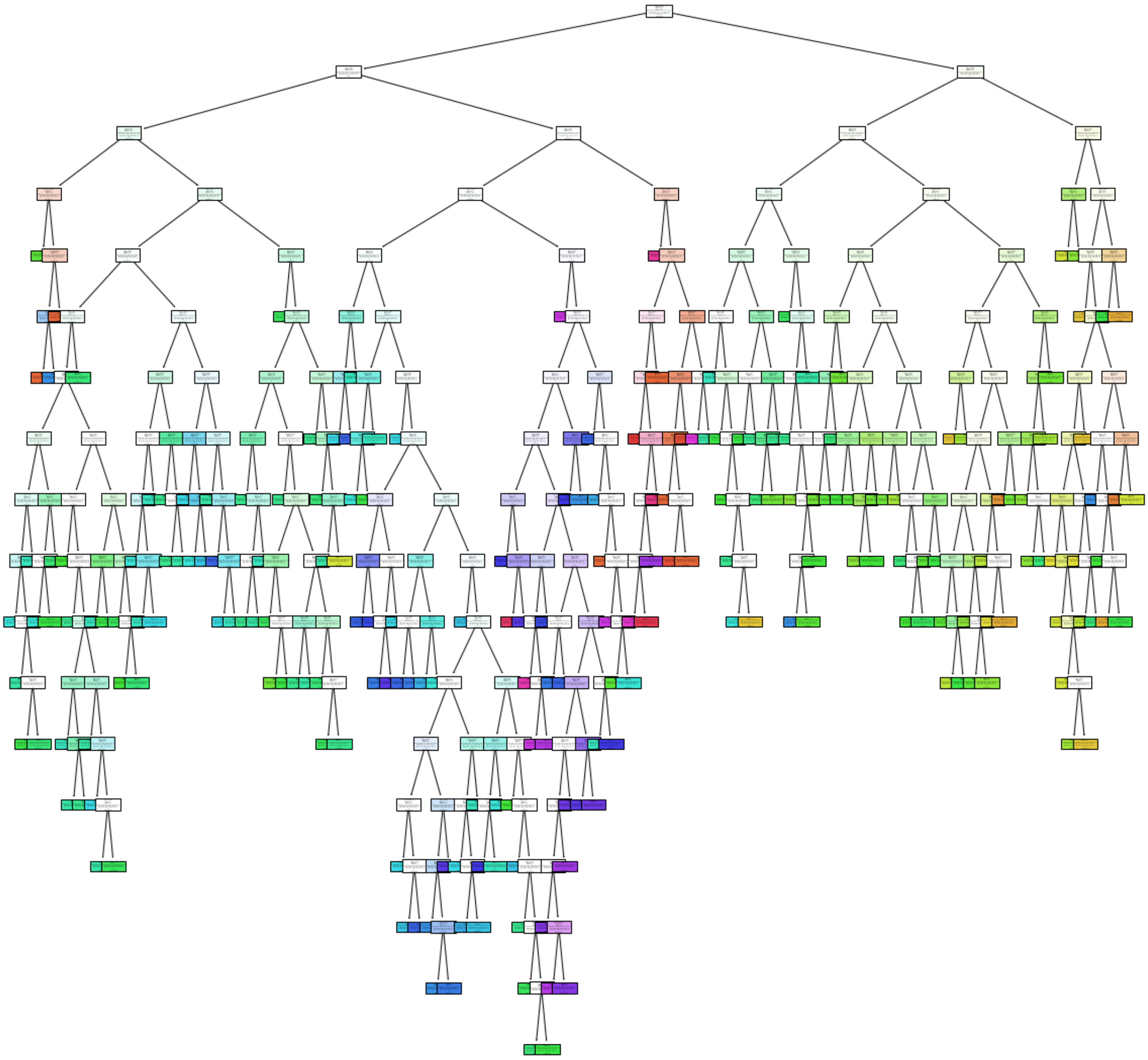
```
Td = dtc.get_depth()
print(”Tree depth:”,Td)
print(”Number of leaves:”,dtc.get_n_leaves())
print(”Feature importance:”,dtc.feature_importances_)
print(”Train score:”,dtc.score(x_train, y_train))
print(”Test score:”,dtc.score(x_test, y_test))
```

```
Tree depth: 17
Number of leaves: 208
Feature importance: [0.11639892 0.01376425 0.03746704 0.0198121  0.05327806 0.16850041
 0.07952211 0.10194358 0.03242634 0.04350267 0.04476839 0.0949689
 0.19364723]
Train score: 1.0
Test score: 0.14473684210526316
```

1. Show the visual output of the decision tree.

In [ ]:

```
plt.figure(figsize = (20, 20))
plot_tree(dtc, class_names=True, filled=True)
plt.show()
```



5.6. 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. For each of the (Td-1) trees report, tree depth, number of leaves, feature importance, train score, and test score of the tree.

In [ ]:

```
high = 0
dtc_high = dtc.fit(x_train, y_train)
for depth in range(1, Td):
    dtc = DecisionTreeClassifier(max_depth=depth)
    dtc = dtc.fit(x_train, y_train)
    dtc_predict = dtc.predict(x_test)

    print(”Tree depth:”, depth)
    print(”Num of leaves:”, dtc.get_n_leaves())
    print(”Feature importance:”, dtc.feature_importances_)
    print(”Train score:”, dtc.score(x_train, y_train))
    score = dtc.score(x_test, y_test)
    print(”Test score:”, score, ”\n”)
    if high < score:
        high = score
        dtc_high = dtc
```

```
Tree depth: 1
Num of leaves: 2
Feature importance: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
Train score: 0.12429378531073447
Test score: 0.07894736842105263

Tree depth: 2
Num of leaves: 4
Feature importance: [0.20598851 0.         0.         0.         0.         0.37193639
 0.         0.         0.         0.         0.         0.
 0.42207509]
Train score: 0.1751412429378531
Test score: 0.11842105263157894

Tree depth: 3
Num of leaves: 8
Feature importance: [0.20999612 0.         0.         0.         0.         0.37759274
 0.         0.17169695 0.         0.         0.         0.
 0.24071418]
Train score: 0.2288135593220339
Test score: 0.13157894736842105

Tree depth: 4
Num of leaves: 16
Feature importance: [0.1610538  0.         0.         0.         0.         0.3659411
 0.04798914 0.10961723 0.         0.         0.05621963 0.
 0.2591791 ]
Train score: 0.2796610169491525
Test score: 0.13815789473684212

Tree depth: 5
Num of leaves: 28
Feature importance: [0.11430989 0.         0.         0.         0.03347662 0.30265243
 0.06281812 0.10148587 0.04636842 0.02617196 0.03680926 0.0715402
 0.20436723]
Train score: 0.3531073446327684
Test score: 0.13815789473684212

Tree depth: 6
Num of leaves: 45
Feature importance: [0.07846657 0.         0.01914967 0.016414   0.05229683 0.29543665
 0.09803161 0.06966369 0.03182901 0.01796541 0.05865467 0.05995934
 0.20213256]
Train score: 0.4322033898305085
Test score: 0.125

Tree depth: 7
Num of leaves: 70
Feature importance: [0.0781501  0.00910401 0.01337732 0.02038448 0.03519508 0.20717841
 0.12042176 0.08431121 0.04102605 0.03930466 0.0548185  0.06628326
 0.23044516]
Train score: 0.5423728813559322
Test score: 0.15789473684210525

Tree depth: 8
Num of leaves: 100
Feature importance: [0.10274438 0.00700229 0.03488026 0.00881921 0.05974347 0.16130982
 0.10707537 0.10141325 0.03669948 0.06341322 0.04975763 0.06460214
 0.20253946]
Train score: 0.652542372881356
Test score: 0.15789473684210525

Tree depth: 9
Num of leaves: 130
Feature importance: [0.08649764 0.02661426 0.0261091  0.01344825 0.05800136 0.19216052
 0.09402781 0.08568403 0.03319371 0.03483648 0.05441274 0.08898376
 0.20603034]
Train score: 0.7598870056497176
Test score: 0.16447368421052633

Tree depth: 10
```

```
Num of leaves: 158
Feature importance: [0.09592624 0.02780741 0.03525161 0.01649873 0.03630112 0.1696883
 0.10420866 0.09516966 0.04450461 0.04522942 0.06117668 0.07413729
 0.19410027]
Train score: 0.844632768361582
Test score: 0.14473684210526316

Tree depth: 11
Num of leaves: 174
Feature importance: [0.12857419 0.02246443 0.02552845 0.01420496 0.06293884 0.14751935
 0.12459135 0.07183835 0.02629236 0.03284036 0.0712833  0.07562026
 0.1963038 ]
Train score: 0.8983050847457628
Test score: 0.16447368421052633

Tree depth: 12
Num of leaves: 185
Feature importance: [0.11032197 0.02155151 0.03376092 0.01847822 0.05146535 0.15750548
 0.10052826 0.10907895 0.03747597 0.02692472 0.0491383  0.09033245
 0.1934379 ]
Train score: 0.9293785310734464
Test score: 0.13815789473684212

Tree depth: 13
Num of leaves: 192
Feature importance: [0.12476522 0.01759045 0.03873441 0.00998127 0.05707406 0.16902527
 0.09861898 0.08518433 0.03464224 0.03232493 0.05332374 0.07846475
 0.20027034]
Train score: 0.9548022598870056
Test score: 0.14473684210526316

Tree depth: 14
Num of leaves: 200
Feature importance: [0.10497855 0.02629659 0.04686025 0.02341222 0.04119144 0.14284505
 0.10820655 0.0704489  0.03284558 0.04972244 0.04864691 0.10835099
 0.19619453]
Train score: 0.9745762711864406
Test score: 0.14473684210526316

Tree depth: 15
Num of leaves: 204
Feature importance: [0.09708057 0.02253332 0.02392806 0.00961694 0.05031052 0.19136921
 0.10157598 0.06826156 0.03879172 0.04667459 0.05304427 0.10232863
 0.19448461]
Train score: 0.9887005649717514
Test score: 0.14473684210526316

Tree depth: 16
Num of leaves: 208
Feature importance: [0.08840204 0.02697232 0.03048363 0.00800915 0.06908558 0.17056124
 0.1158841  0.08624847 0.03586887 0.02874899 0.05509597 0.10706693
 0.17757274]
Train score: 1.0
Test score: 0.15789473684210525
```

1. Show the visual output of the decision tree with highest test score from the (Td-1) trees.

In [ ]:

```
print("The output of the decision tree with highest test score")
print("Tree depth:", dtc_high.get_depth())
print("Test score:", high)
plt.figure(figsize = (20, 20))
plot_tree(dtc_high, class_names=True, filled=True)
plt.show()
```

```
The output of the decision tree with highest test score
Tree depth: 9
Test score: 0.16447368421052633
```