In [2]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
from statsmodels.datasets import get_rdataset
import sklearn.model_selection as skm
from ISLP import load_data, confusion_table
from ISLP.models import ModelSpec as MS
```

In [3]:
```
from sklearn.tree import (DecisionTreeClassifier as DTC,
DecisionTreeRegressor as DTR,
plot_tree,
export_text)
from sklearn.metrics import (accuracy_score,
log_loss)
from sklearn.ensemble import \
(RandomForestRegressor as RF,
GradientBoostingRegressor as GBR)
from ISLP.bart import BART
```

In [7]:
```
Carseats = load_data('Carseats')
High = np.where(Carseats.Sales > 8,
"Yes",
"No")
High.dtype
```

Out[7]:
```
dtype('<U3')
```

In [4]:
```
model = MS(Carseats.columns.drop('Sales'), intercept=False)
D = model.fit_transform(Carseats)
feature_names = list(D.columns)
X = np.asarray(D)
```

In [5]:
```
clf = DTC(criterion='entropy',
max_depth=3,
random_state=0)
clf.fit(X, High)
```

Out[5]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [6]:
```
accuracy_score(High, clf.predict(X))
```

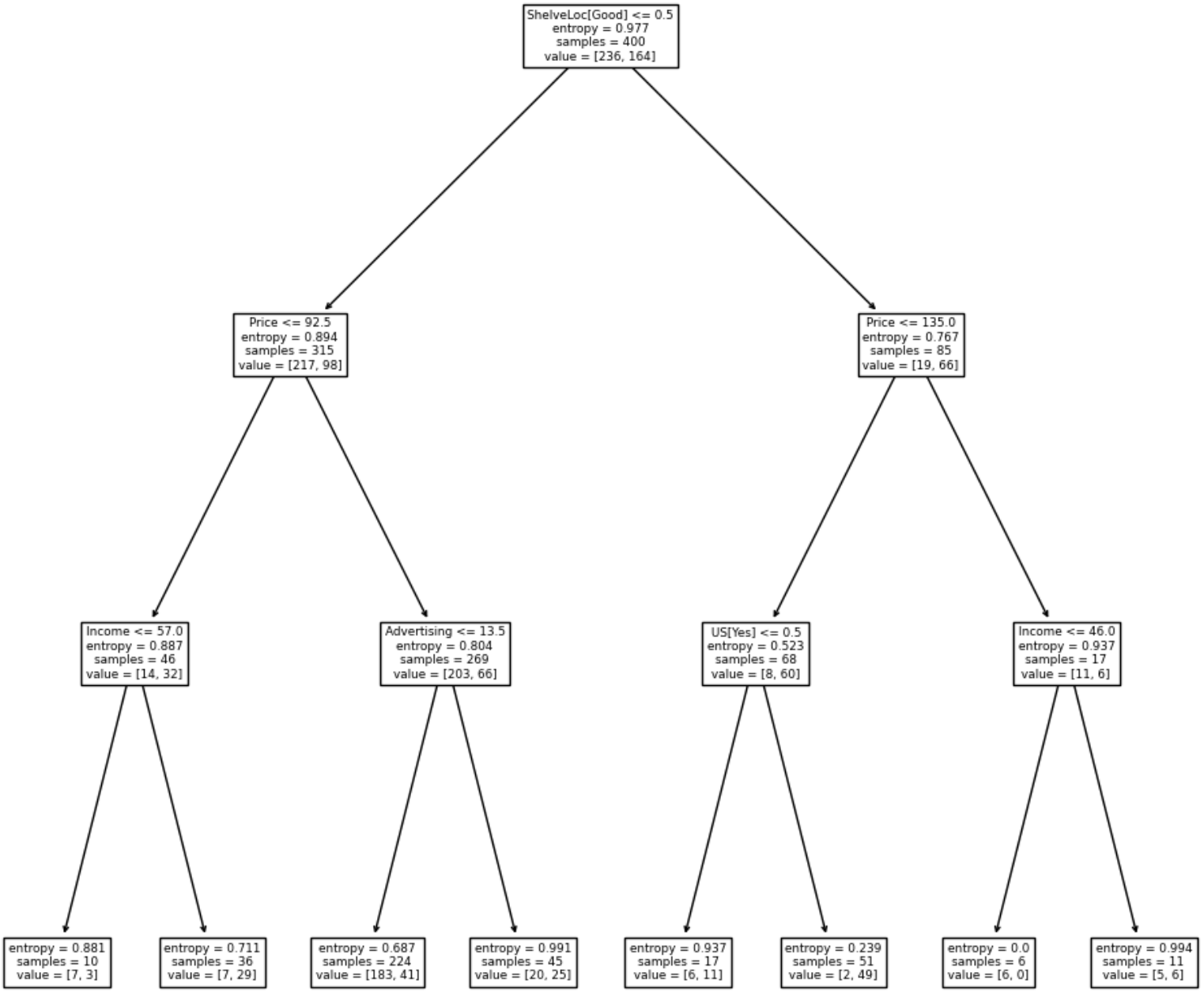
Out[6]:
```
0.79
```

In [7]:
```
resid_dev = np.sum(log_loss(High, clf.predict_proba(X)))
resid_dev
```

Out[7]:
```
0.4710647062649358
```

In [8]:

```
ax = subplots(figsize=(12,12))[1]
plot_tree(clf,
feature_names=feature_names,
ax=ax);
```


ShelveLoc[Good] <= 0.5
entropy = 0.977
samples = 400
value = [236, 164]
Price <= 92.5
entropy = 0.894
samples = 315
value = [217, 98]
Price <= 135.0
entropy = 0.767
samples = 85
value = [19, 66]
Income <= 57.0
entropy = 0.887
samples = 46
value = [14, 32]
Advertising <= 13.5
entropy = 0.804
samples = 269
value = [203, 66]
US[Yes] <= 0.5
entropy = 0.523
samples = 68
value = [8, 60]
Income <= 46.0
entropy = 0.937
samples = 17
value = [11, 6]
entropy = 0.881
samples = 10
value = [7, 3]
entropy = 0.711
samples = 36
value = [7, 29]
entropy = 0.687
samples = 224
value = [183, 41]
entropy = 0.991
samples = 45
value = [20, 25]
entropy = 0.937
samples = 17
value = [6, 11]
entropy = 0.239
samples = 51
value = [2, 49]
entropy = 0.0
samples = 6
value = [6, 0]
entropy = 0.994
samples = 11
value = [5, 6]

In [9]:

```
print(export_text(clf,
feature_names=feature_names,
show_weights=True))
```

```
|--- ShelveLoc[Good] <= 0.50
|   |--- Price <= 92.50
|   |   |--- Income <= 57.00
|   |   |   |--- weights: [7.00, 3.00] class: No
|   |   |--- Income >  57.00
|   |   |   |--- weights: [7.00, 29.00] class: Yes
|   |--- Price >  92.50
|   |   |--- Advertising <= 13.50
|   |   |   |--- weights: [183.00, 41.00] class: No
|   |   |--- Advertising >  13.50
|   |   |   |--- weights: [20.00, 25.00] class: Yes
|--- ShelveLoc[Good] >  0.50
|   |--- Price <= 135.00
|   |   |--- US[Yes] <= 0.50
|   |   |   |--- weights: [6.00, 11.00] class: Yes
|   |   |--- US[Yes] >  0.50
|   |   |   |--- weights: [2.00, 49.00] class: Yes
|   |--- Price >  135.00
|   |   |--- Income <= 46.00
|   |   |   |--- weights: [6.00, 0.00] class: No
|   |   |--- Income >  46.00
|   |   |   |--- weights: [5.00, 6.00] class: Yes
```

In [10]:
```
validation = skm.ShuffleSplit(n_splits=1,
test_size=200,
random_state=0)
results = skm.cross_validate(clf,
D,
High,
cv=validation)
results['test_score']
```

Out[10]:
```
array([0.685])
```

In [11]:
```
(X_train,
X_test,
High_train,
High_test) = skm.train_test_split(X,
High,
test_size=0.5,
random_state=0)
```

In [12]:
```
clf = DTC(criterion='entropy', random_state=0)
clf.fit(X_train, High_train)
accuracy_score(High_test, clf.predict(X_test))
```

Out[12]:
```
0.735
```

In [14]:
```
ccp_path = clf.cost_complexity_pruning_path(X_train, High_train)
kfold = skm.KFold(10,
random_state=1,
shuffle=True)
```

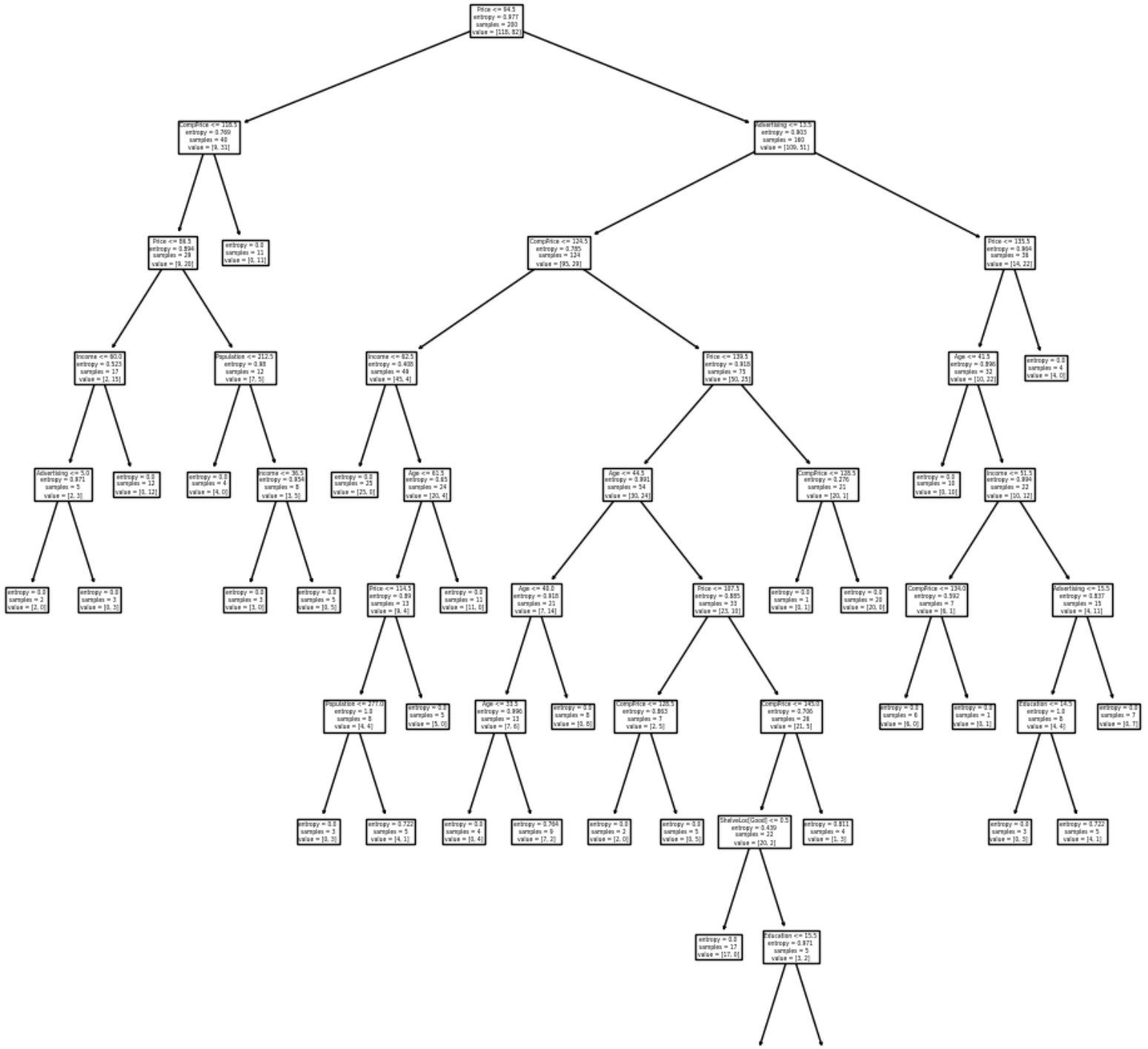
In [15]:

```
grid = skm.GridSearchCV(clf,
{'ccp_alpha': ccp_path.ccp_alphas},
refit=True,cv=kfold,
scoring='accuracy')
grid.fit(X_train, High_train)
grid.best_score_
```

Out[15]: 0.685

In [16]:

```
ax = subplots(figsize=(12, 12))[1]
best_ = grid.best_estimator_
plot_tree(best_,
feature_names=feature_names,
ax=ax);
```


Price <= 94.5
entropy = 0.977
samples = 280
value = [118, 82]
CompPrice <= 108.5
entropy = 0.769
samples = 48
value = [9, 33]
Advertising <= 13.5
entropy = 0.803
samples = 180
value = [109, 51]
Price <= 86.5
entropy = 0.894
samples = 29
value = [9, 20]
entropy = 0.0
samples = 11
value = [0, 13]
CompPrice <= 124.5
entropy = 0.795
samples = 124
value = [95, 29]
Price <= 135.5
entropy = 0.864
samples = 56
value = [14, 42]
Income <= 80.0
entropy = 0.523
samples = 17
value = [2, 15]
Population <= 212.5
entropy = 0.98
samples = 12
value = [7, 5]
Income <= 82.5
entropy = 0.408
samples = 69
value = [45, 4]
Price <= 139.5
entropy = 0.918
samples = 75
value = [50, 25]
Age <= 42.5
entropy = 0.896
samples = 32
value = [10, 22]
entropy = 0.0
samples = 4
value = [4, 0]
Advertising <= 5.0
entropy = 0.971
samples = 5
value = [2, 3]
entropy = 0.0
samples = 12
value = [0, 12]
entropy = 0.0
samples = 4
value = [4, 0]
Income <= 36.5
entropy = 0.954
samples = 8
value = [3, 5]
entropy = 0.0
samples = 25
value = [25, 0]
Age <= 61.5
entropy = 0.65
samples = 24
value = [20, 4]
Age <= 44.5
entropy = 0.991
samples = 54
value = [30, 24]
CompPrice <= 120.5
entropy = 0.276
samples = 21
value = [20, 1]
entropy = 0.0
samples = 10
value = [0, 10]
Income <= 51.5
entropy = 0.996
samples = 22
value = [10, 12]
entropy = 0.0
samples = 5
value = [0, 5]
entropy = 0.0
samples = 3
value = [3, 0]
entropy = 0.0
samples = 5
value = [0, 5]
Price <= 114.5
entropy = 0.89
samples = 13
value = [9, 4]
entropy = 0.0
samples = 11
value = [11, 0]
Age <= 48.0
entropy = 0.918
samples = 21
value = [7, 14]
Price <= 107.5
entropy = 0.885
samples = 33
value = [23, 10]
entropy = 0.0
samples = 1
value = [1, 0]
entropy = 0.0
samples = 20
value = [20, 0]
CompPrice <= 134.0
entropy = 0.592
samples = 7
value = [6, 1]
Advertising <= 15.5
entropy = 0.837
samples = 15
value = [4, 11]
Population <= 277.0
entropy = 1.0
samples = 8
value = [4, 4]
entropy = 0.0
samples = 5
value = [5, 0]
Age <= 33.5
entropy = 0.996
samples = 13
value = [7, 6]
entropy = 0.0
samples = 8
value = [0, 8]
CompPrice <= 128.5
entropy = 0.863
samples = 7
value = [2, 5]
CompPrice <= 143.0
entropy = 0.706
samples = 26
value = [21, 5]
entropy = 0.0
samples = 6
value = [6, 0]
entropy = 0.0
samples = 1
value = [0, 1]
Education <= 14.5
entropy = 1.0
samples = 8
value = [4, 4]
entropy = 0.0
samples = 7
value = [0, 7]
entropy = 0.0
samples = 3
value = [0, 3]
entropy = 0.722
samples = 5
value = [4, 1]
entropy = 0.0
samples = 4
value = [0, 4]
entropy = 0.764
samples = 9
value = [7, 2]
entropy = 0.0
samples = 2
value = [2, 0]
entropy = 0.0
samples = 5
value = [0, 5]
ShelveLoc[Good] <= 0.5
entropy = 0.439
samples = 22
value = [20, 2]
entropy = 0.811
samples = 4
value = [1, 3]
entropy = 0.0
samples = 3
value = [0, 3]
entropy = 0.722
samples = 5
value = [4, 1]
entropy = 0.0
samples = 17
value = [17, 0]
Education <= 15.5
entropy = 0.971
samples = 5
value = [3, 2]


In [17]:
```
best_.tree_.n_leaves
```

Out[17]: 30

In [18]:
```
print(accuracy_score(High_test,
best_.predict(X_test)))
confusion = confusion_table(best_.predict(X_test),
High_test)
confusion
```

0.72

Out[18]:

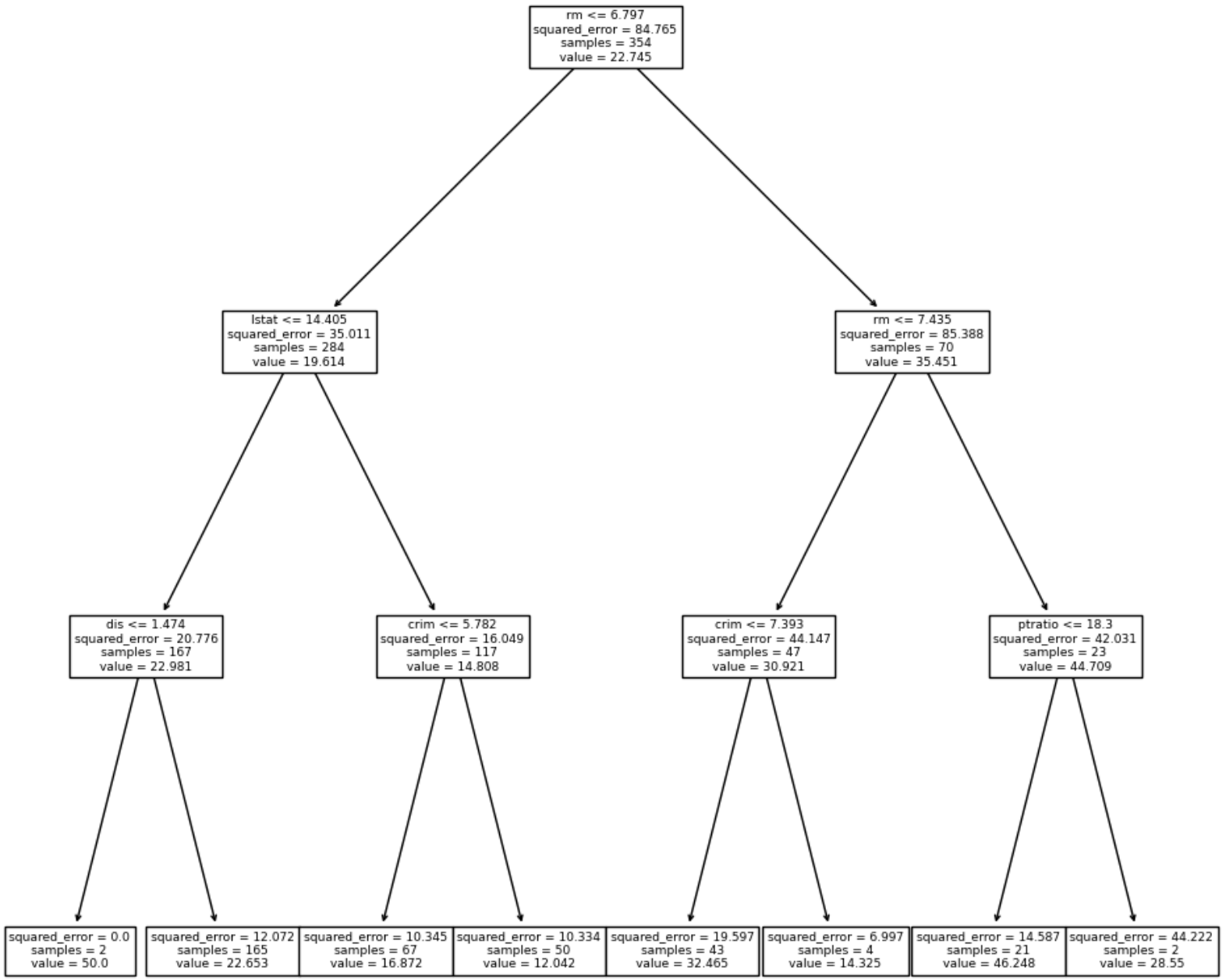
Truth / Predicted	No	Yes
No	94	32
Yes	24	50

In [19]:
```
Boston = load_data("Boston")
model = MS(Boston.columns.drop('medv'), intercept=False)
D = model.fit_transform(Boston)
feature_names = list(D.columns)
X = np.asarray(D)
```

In [20]:
```
(X_train,
X_test,
y_train,
y_test) = skm.train_test_split(X,
Boston['medv'],
test_size=0.3,
random_state=0)
```

In [21]:

```
reg = DTR(max_depth=3)
reg.fit(X_train, y_train)
ax = subplots(figsize=(12,12))[1]
plot_tree(reg,
feature_names=feature_names,
ax=ax);
```


rm <= 6.797
squared_error = 84.765
samples = 354
value = 22.745
lstat <= 14.405
squared_error = 35.011
samples = 284
value = 19.614
rm <= 7.435
squared_error = 85.388
samples = 70
value = 35.451
dis <= 1.474
squared_error = 20.776
samples = 167
value = 22.981
crim <= 5.782
squared_error = 16.049
samples = 117
value = 14.808
crim <= 7.393
squared_error = 44.147
samples = 47
value = 30.921
ptratio <= 18.3
squared_error = 42.031
samples = 23
value = 44.709
squared_error = 0.0
samples = 2
value = 50.0
squared_error = 12.072
samples = 165
value = 22.653
squared_error = 10.345
samples = 67
value = 16.872
squared_error = 10.334
samples = 50
value = 12.042
squared_error = 19.597
samples = 43
value = 32.465
squared_error = 6.997
samples = 4
value = 14.325
squared_error = 14.587
samples = 21
value = 46.248
squared_error = 44.222
samples = 2
value = 28.55

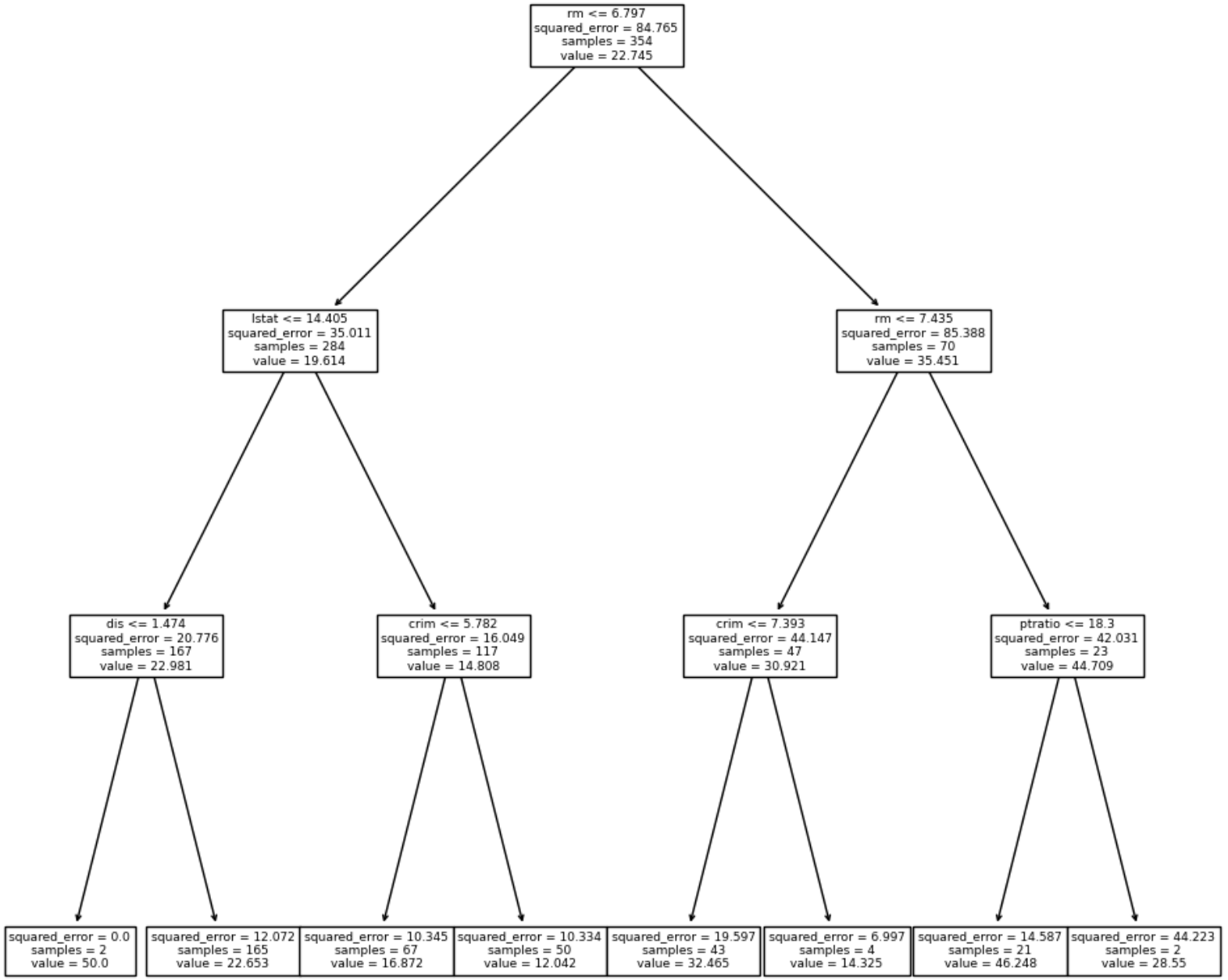
In [22]:
```
ccp_path = reg.cost_complexity_pruning_path(X_train, y_train)
kfold = skm.KFold(5,
shuffle=True,
random_state=10)
grid = skm.GridSearchCV(reg,
{'ccp_alpha': ccp_path.ccp_alphas},
refit=True,
cv=kfold,
scoring='neg_mean_squared_error')
G = grid.fit(X_train, y_train)
```

In [23]:
```
best_ = grid.best_estimator_
np.mean((y_test - best_.predict(X_test))**2)
```

Out[23]:
```
28.06985754975404
```

In [24]:

```
ax = subplots(figsize=(12,12))[1]
plot_tree(G.best_estimator_,
feature_names=feature_names,
ax=ax);
```


rm <= 6.797
squared_error = 84.765
samples = 354
value = 22.745
lstat <= 14.405
squared_error = 35.011
samples = 284
value = 19.614
rm <= 7.435
squared_error = 85.388
samples = 70
value = 35.451
dis <= 1.474
squared_error = 20.776
samples = 167
value = 22.981
crim <= 5.782
squared_error = 16.049
samples = 117
value = 14.808
crim <= 7.393
squared_error = 44.147
samples = 47
value = 30.921
ptratio <= 18.3
squared_error = 42.031
samples = 23
value = 44.709
squared_error = 0.0
samples = 2
value = 50.0
squared_error = 12.072
samples = 165
value = 22.653
squared_error = 10.345
samples = 67
value = 16.872
squared_error = 10.334
samples = 50
value = 12.042
squared_error = 19.597
samples = 43
value = 32.465
squared_error = 6.997
samples = 4
value = 14.325
squared_error = 14.587
samples = 21
value = 46.248
squared_error = 44.223
samples = 2
value = 28.55

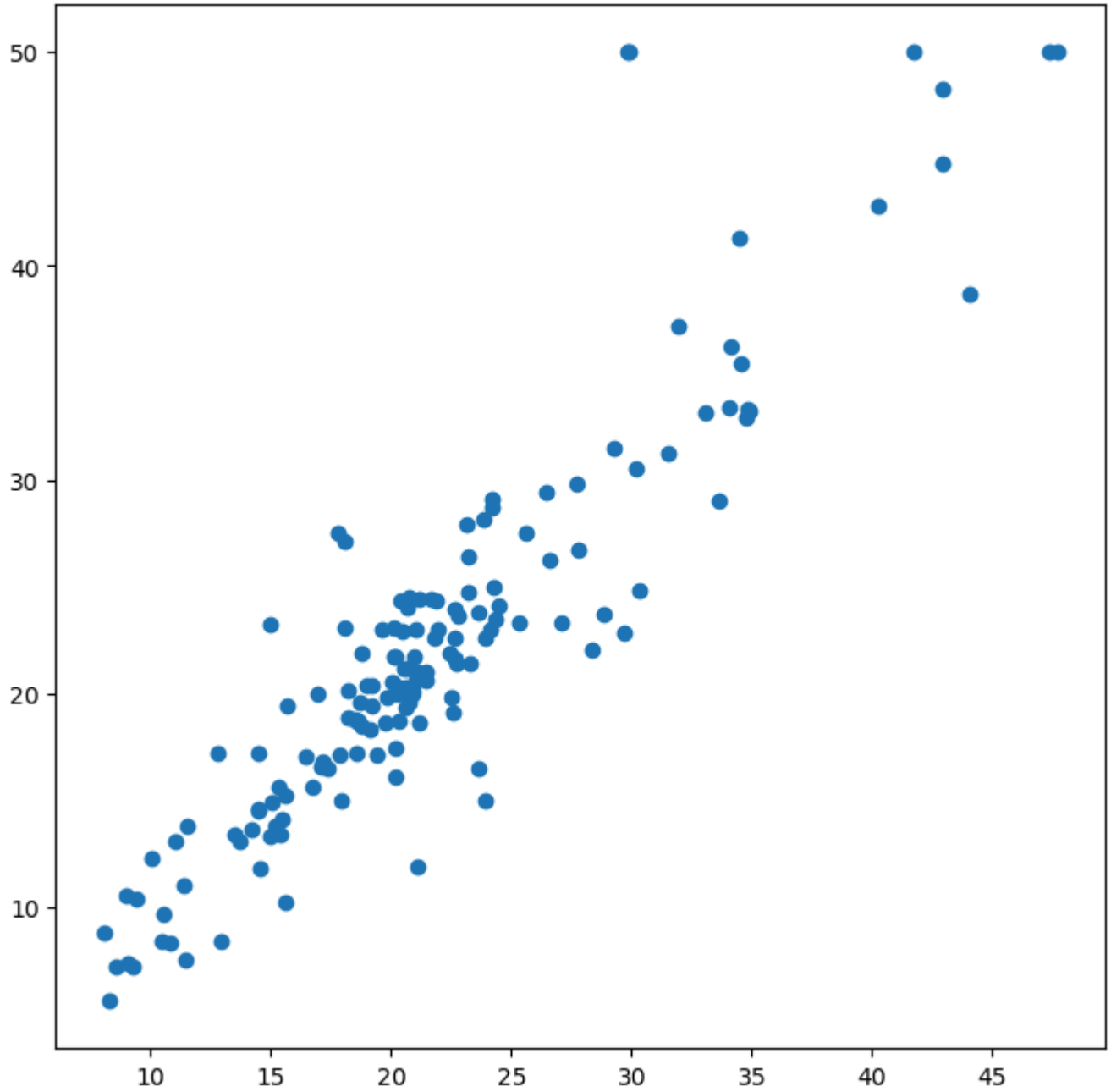
In [25]:
```
bag_boston = RF(max_features=X_train.shape[1], random_state=0)
bag_boston.fit(X_train, y_train)
```

Out[25]:
```
RandomForestRegressor(max_features=12, random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:
```
ax = subplots(figsize=(8,8))[1]
y_hat_bag = bag_boston.predict(X_test)
ax.scatter(y_hat_bag, y_test)
np.mean((y_test - y_hat_bag)**2)
```

Out[26]:
```
14.684333796052627
```


50
40
30
20
10
10
15
20
25
30
35
40
45

In [27]:
```
bag_boston = RF(max_features=X_train.shape[1],
n_estimators=500,
random_state=0).fit(X_train, y_train)
y_hat_bag = bag_boston.predict(X_test)
np.mean((y_test - y_hat_bag)**2)
```

Out[27]:
```
14.565312103157904
```

In [28]:
```
RF_boston = RF(max_features=6,
random_state=0).fit(X_train, y_train)
y_hat_RF = RF_boston.predict(X_test)
np.mean((y_test - y_hat_RF)**2)
```

Out[28]:
```
19.998839111842113
```

In [29]:
```
feature_imp = pd.DataFrame(
{'importance':RF_boston.feature_importances_},
index=feature_names)
feature_imp.sort_values(by='importance', ascending=False)
```

Out[29]:

	importance
lstat	0.353808
rm	0.334349
ptratio	0.069519
crim	0.056386
indus	0.053183
dis	0.043762
nox	0.033085
tax	0.025047
age	0.019238
rad	0.005169
chas	0.004331
zn	0.002123

In [30]:

```
boost_boston = GBR(n_estimators=5000,
learning_rate=0.001,
max_depth=3,
random_state=0)
boost_boston.fit(X_train, y_train)
```

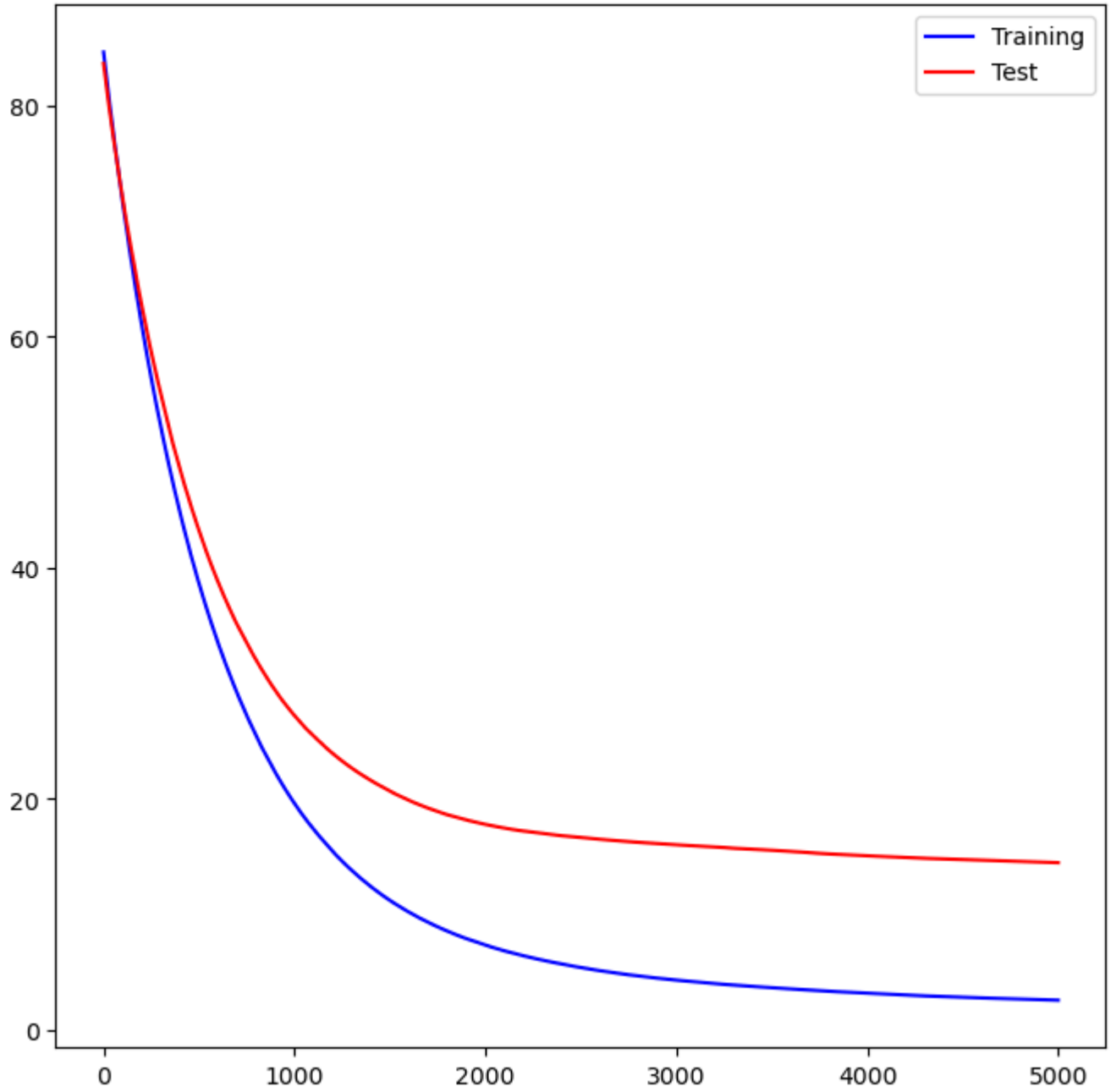
Out[30]:

```
GradientBoostingRegressor(learning_rate=0.001, n_estimators=5000,
                          random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:

```
test_error = np.zeros_like(boost_boston.train_score_)
for idx, y_ in enumerate(boost_boston.staged_predict(X_test)):
    test_error[idx] = np.mean((y_test - y_)**2)
plot_idx = np.arange(boost_boston.train_score_.shape[0])
ax = subplots(figsize=(8,8))[1]
ax.plot(plot_idx,
boost_boston.train_score_,
'b',
label='Training')
ax.plot(plot_idx,
test_error,
    'r',
    label='Test')
ax.legend();
```


Training
Test
80
60
40
20
0
0
1000
2000
3000
4000
5000

In [35]:
```
ax.plot(plot_idx,
test_error,
'r',
label='Test')
ax.legend();
```

In [36]:
```
boost_boston = GBR(n_estimators=5000,
learning_rate=0.2,
max_depth=3,
random_state=0)
boost_boston.fit(X_train,
y_train)
y_hat_boost = boost_boston.predict(X_test);
np.mean((y_test - y_hat_boost)**2)
```

Out[36]:
```
14.501514553719568
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: