Recap

You've built your first model, and now it's time to optimize the size of the tree to make better predictions. Run this cell to set up your coding environment where the previous step left off.

In [1]:
```
# Code you have previously used to load data
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor


# Path of the file to read
iowa_file_path = '../input/home-data-for-ml-course/train.csv'

home_data = pd.read_csv(iowa_file_path)
# Create target object and call it y
y = home_data.SalePrice
# Create X
features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'Bedro
omAbvGr', 'TotRmsAbvGrd']
X = home_data[features]

# Split into validation and training data
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)

# Specify Model
iowa_model = DecisionTreeRegressor(random_state=1)
# Fit Model
iowa_model.fit(train_X, train_y)

# Make validation predictions and calculate mean absolute error
val_predictions = iowa_model.predict(val_X)
val_mae = mean_absolute_error(val_predictions, val_y)
print("Validation MAE: {:,.0f}".format(val_mae))

# Set up code checking
from learntools.core import binder
binder.bind(globals())
from learntools.machine_learning.ex5 import *
print("\nSetup complete")
```

```
Validation MAE: 29,653

Setup complete
```

Exercises

You could write the function `get_mae` yourself. For now, we'll supply it. This is the same function you read about in the previous lesson. Just run the cell below.

In [2]:
```
def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_state=
0)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)
```

Step 1: Compare Different Tree Sizes

Write a loop that tries the following values for *max_leaf_nodes* from a set of possible values.

Call the *get_mae* function on each value of max_leaf_nodes. Store the output in some way that allows you to select the value of `max_leaf_nodes` that gives the most accurate model on your data.

In [3]:
```
candidate_max_leaf_nodes = [5, 25, 50, 100, 250, 500]
# Write loop to find the ideal tree size from candidate_max_leaf_nodes
for max_leaf_nodes in candidate_max_leaf_nodes:
    my_mae = get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y)
    print("Max leaf nodes: %d  \t\t Mean Absolute Error:  %d" %(max_leaf_nodes
, my_mae))

# Store the best value of max_leaf_nodes (it will be either 5, 25, 50, 100, 25
0 or 500)
scores = {leaf_size: get_mae(leaf_size, train_X, val_X, train_y, val_y) for le
af_size in candidate_max_leaf_nodes}
print("\n",scores)

best_tree_size = min(scores, key=scores.get)

print("\n", best_tree_size, scores[best_tree_size])

step_1.check()
```

```
Max leaf nodes: 5                Mean Absolute Error:  35044
Max leaf nodes: 25               Mean Absolute Error:  29016
Max leaf nodes: 50               Mean Absolute Error:  27405
Max leaf nodes: 100              Mean Absolute Error:  27282
Max leaf nodes: 250              Mean Absolute Error:  27893
Max leaf nodes: 500              Mean Absolute Error:  29454

 {5: 35044.51299744237, 25: 29016.41319191076, 50: 27405.930473214907, 100: 2
7282.50803885739, 250: 27893.822225701646, 500: 29454.18598068598}

 100 27282.50803885739
```

Correct

In [4]:
```
# The lines below will show you a hint or the solution.
step_1.hint()
step_1.solution()
```

Hint: You will call get_mae in the loop. You'll need to map the names of your data structure to the names in get_mae

Solution:
```
# Here is a short solution with a dict comprehension.
# The lesson gives an example of how to do this with an explicit loop.
scores = {leaf_size: get_mae(leaf_size, train_X, val_X, train_y, val_y) for l
eaf_size in candidate_max_leaf_nodes}
best_tree_size = min(scores, key=scores.get)
```

Step 2: Fit Model Using All Data

You know the best tree size. If you were going to deploy this model in practice, you would make it even more accurate by using all of the data and keeping that tree size. That is, you don't need to hold out the validation data now that you've made all your modeling decisions.

In [5]:
```
# Fit the model with best_tree_size. Fill in argument to make optimal size
final_model = DecisionTreeRegressor(max_leaf_nodes=best_tree_size, random_stat
e=0)

# fit the final model
final_model.fit(X, y)

step_2.check()
```

Correct

In [6]:
```
step_2.hint()
step_2.solution()
```

Hint: Fit with the ideal value of max_leaf_nodes. In the fit step, use all of the data in the dataset

Solution:
```
# Fit the model with best_tree_size. Fill in argument to make optimal size
final_model = DecisionTreeRegressor(max_leaf_nodes=best_tree_size, random_sta
te=1)

# fit the final model
final_model.fit(X, y)
```

Keep Going

You've tuned this model and improved your results. But we are still using Decision Tree models, which are not very sophisticated by modern machine learning standards. In the next step you will learn to use **Random Forests** to improve your models even more. Random Forests are based on decision trees, but they typically give you much better predictions.

Course Home Page