This notebook is an exercise in the Introduction to Machine Learning course. You can reference the tutorial at this link.

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 [ ]:
```
# 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',
'BedroomAbvGr', '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")
```

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 [ ]:
```
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 [ ]:
```
candidate_max_leaf_nodes = [5, 25, 50, 100, 250, 500]
# Write loop to find the ideal tree size from candidate_max_leaf_nodes
pred=[]
for i in candidate_max_leaf_nodes:
    temp=get_mae(i,train_X, val_X, train_y, val_y)
    pred.append(temp)

low_mae=min(pred)
index=pred.index(low_mae)
# Store the best value of max_leaf_nodes (it will be either 5, 25, 50,
 100, 250 or 500)
best_tree_size = candidate_max_leaf_nodes[index]

# Check your answer
step_1.check()
```

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

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 [ ]:
```
# Fill in argument to make optimal size and uncomment
final_model = DecisionTreeRegressor(max_leaf_nodes=candidate_max_leaf_n
odes[index],random_state=1)

# fit the final model and uncomment the next two lines
final_model.fit(X,y)

# Check your answer
step_2.check()
```

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

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.

Keep Going

You are ready for **Random Forests**.

Have questions or comments? Visit the Learn Discussion forum to chat with other Learners.