Exercise: Underfitting and Overfitting

## 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.

# 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")

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.

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.

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, 250 or 500)

best\_tree\_size = 100

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

## 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.

# Fill in argument to make optimal size and uncomment

final\_model = DecisionTreeRegressor(max\_leaf\_nodes=100)

# fit the final model and uncomment the next two lines

final\_model.fit(X, y)

step\_2.check()

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