CST 383 - Intro to Data Science

Dr. Glenn Bruns

# Lab: Decision trees, part 3

With Scikit-Learn, the quality of your decision tree will depend a lot on how you select hyperparameters. The purpose of this lab is to help you get familiar with the hyperparameters of DecisionTreeRegressor.

The lab is quite open-ended. I want you to predict CPU performance using a regression tree. First create a file with this starter code shown on the next page.

Next, do the following:

* pick some predictors, and create training and test sets
* create a DecisionTreeRegressor, and set some of the hyperparameters listed in lecture:
  + min\_samples\_split, max\_depth, min\_samples\_leaf, max\_leaf\_nodes, min\_impurity\_decrease
* train your tree and see what it looks like
* compute the RMSE for your model on the test data ("test RMSE")

Do this a bunch of times to see how the hyperparameters affect the appearance of your tree and your test RMSE.

Be careful not to create a giant tree. Trying to plot a giant tree can bring your machine to a standstill.

If you have time, try doing a grid search with cross validation to find the best hyperparameter settings.

(Starter code on next page)

import numpy as np

import pandas as pd

from matplotlib import rcParams

from sklearn.tree import DecisionTreeRegressor, export\_graphviz

from sklearn.model\_selection import train\_test\_split

import seaborn as sns

import graphviz

# switch to seaborn default stylistic parameters

sns.set()

# larger plot fonts

sns.set\_context('talk')

# change default plot size

rcParams['figure.figsize'] = 10,8

# read the cpu data

df = pd.read\_csv("https://raw.githubusercontent.com/grbruns/cst383/master/machine.csv")

df.index = df['vendor']+' '+df['model']

df.drop(['vendor', 'model'], axis=1, inplace=True)

df['cs'] = np.round(1e3/df['myct'], 2) # clock speed in MHz

# get ready for Scikit-Learn

predictors = ['mmin', 'chmax'] # choose predictors as you like

target = 'prp'

X = df[predictors].values

y = df[target].values

# test/train split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=0)

# train the model

reg = DecisionTreeRegressor(max\_depth=2)

reg.fit(X\_train, y\_train)

# view the tree

# conda install python-graphviz

dot\_data = export\_graphviz(reg, precision=2,

feature\_names=predictors,

proportion=True,

filled=True, rounded=True)

graph = graphviz.Source(dot\_data)

graph

# make predictions and compute error

y\_predict = reg.predict(X\_test)

errors = y\_test - y\_predict

rmse = np.sqrt((errors\*\*2).mean())

print('rmse: {:0.2f}'.format(rmse))