Python vs R

A Comparison for KNN Models

## Introduction

While building a KNN model in R for project 2 in DATA 624, I decided to try and build the same model in Python and compare both the process of building the models and their outcomes. All code is attached in a separate file, as this write-up is solely meant to be a comparison of the process in each language.

## R

To complete this project in R, I used three different libraries: caret, fnn, and kknn. Each library used different inputs to control the model, though caret and fnn produced the same output. The kknn library ended up producing the best output in terms of MAPE score. R was relatively easy to use for this and caret is an amazing library for pre-processing data. One of the nice parts of working in R is the ability to code in a more stream-of-consciousness was using rmd files. This allows a linear flow of code where the user can go through their process and show mistakes.

## Python

In Python, I used the scikit-learn library to build my model and pandas and numpy to perform data manipulation. The only file type in Python that can do this is a notebook style, which often ends up in a different IDE from where your running program is written, so the code was built to run in a single pass and ends up a little less linear and a little less thoughtful than the code in R.

## Comparison

The first big difference to note (in my opinion), is that I was able to use SO MANY libraries to complete this process in R. I’m sure there are other libraries for building KNN models in Python besides just scikit-learn, but scikit-learn has such a monopoly on the Python model building world that almost all of the available online help was focused on that one library alone. R, on the other hand, had almost too many libraries available for me to choose from. Any internet search I did popped up with tons of libraries to choose from and tons of resources on how to use those models. I typically think of Python as one of the strongest languages when it comes to number of libraries and help available, but in this area, I really felt like R was more robust (this was a big surprise to me).

The next big difference is that I found it so much easier to pre-process my data in R. In fact, it was so much easier in R that I just scraped it all in Python and inserted the pre-processed data from R to use for model building in Python. I think this speaks more to the nature of why R was designed and what it’s meant to do than an actual negative against Python though. I’m just not a big fan of scikit-learn’s pre-processing methods.

When it actually came time to build a model, I felt like the tables really turned. After the data had been wrangled and processed, I REALLY enjoyed using scikit-learn to build my model. It gave me an incredible amount of control over the models I was building and essentially acted as all three libraries I used from R and more wrapped up in a single library. I was able to choose the algorithm for finding neighbors, decide on a distance metric (from a vast variety of options), and even build my own custom weights all from a single command in Python. To do more constrained versions of these same things in R, I had to use three different packages, which still couldn’t run all of these options at once.

Outside of model building, there were plusses and minuses to both languages. It was definitely easier to manipulate data in R (this may speak to my lack of comfortability with pandas in Python more than anything), but the plotting in both (ggplot for R and matplotlib for Python) were very similar in terms of how easy they were to use.