Alhassane Samassekou

ITAI 1371

Reflective Journal: Unpacking the Data Scientist's Toolkit

The first big breakthrough for me was with Pandas. I'd heard it was "like a powerful Excel," which sounds cool, but it didn't really sink in until I got past just looking at the data with df.head(). The real lightbulb moment was using df.groupby('species').mean(). I've seriously wasted so much time in Excel making PivotTables to do exactly that, and here it was, done in one line. It hit me then that I wasn't just looking at a spreadsheet anymore. I was actually talking to the data, asking it questions and getting answers back instantly. It felt so much more interactive.

NumPy was the next piece of the puzzle. When we had to get the mean and standard deviation, my first instinct from my intro to programming classes was to write a for loop. Seeing np.mean() do it all in one go felt like a magic trick. It's more than just a shortcut, though. It's a whole different way of thinking about code. I realized I wasn't giving the computer a recipe of instructions anymore. Instead, I was just telling it the result I wanted. This idea of vectorization, where you operate on the whole array at once, is huge. It makes the code so much cleaner and lets NumPy do all the heavy lifting in the background, which is way faster.

But honestly, the best part was using Matplotlib. Before that, the Iris dataset was just a bunch of numbers in a table. Sure, the stats were useful, but they were pretty boring. The second that scatter plot popped up on my screen, everything changed. The data suddenly had a story. I could literally see how the setosa flowers were different from the others, and how the versicolor and virginica species were all mixed up. It wasn't just a table of numbers I had to interpret; it was a picture that made sense instantly. That's when I realized that making graphs isn't just for the final report. It's part of how you figure things out. The plot made me curious and want to ask more questions, like if looking at petals instead of sepals would make a clearer picture.

Using the Jupyter Notebook to mix code and text also started to make a lot more sense. At first, writing the markdown analysis felt like extra homework after the coding was done. But having to actually write down my thoughts in plain English forced me to stop and really think about what the code's output meant. It's easy to just run the code and move on, but explaining it makes you understand it on a deeper level. Tying this all together with

GitHub really framed it for me. This isn't just about writing a script that runs. It’s about creating a whole project that someone else can actually understand.