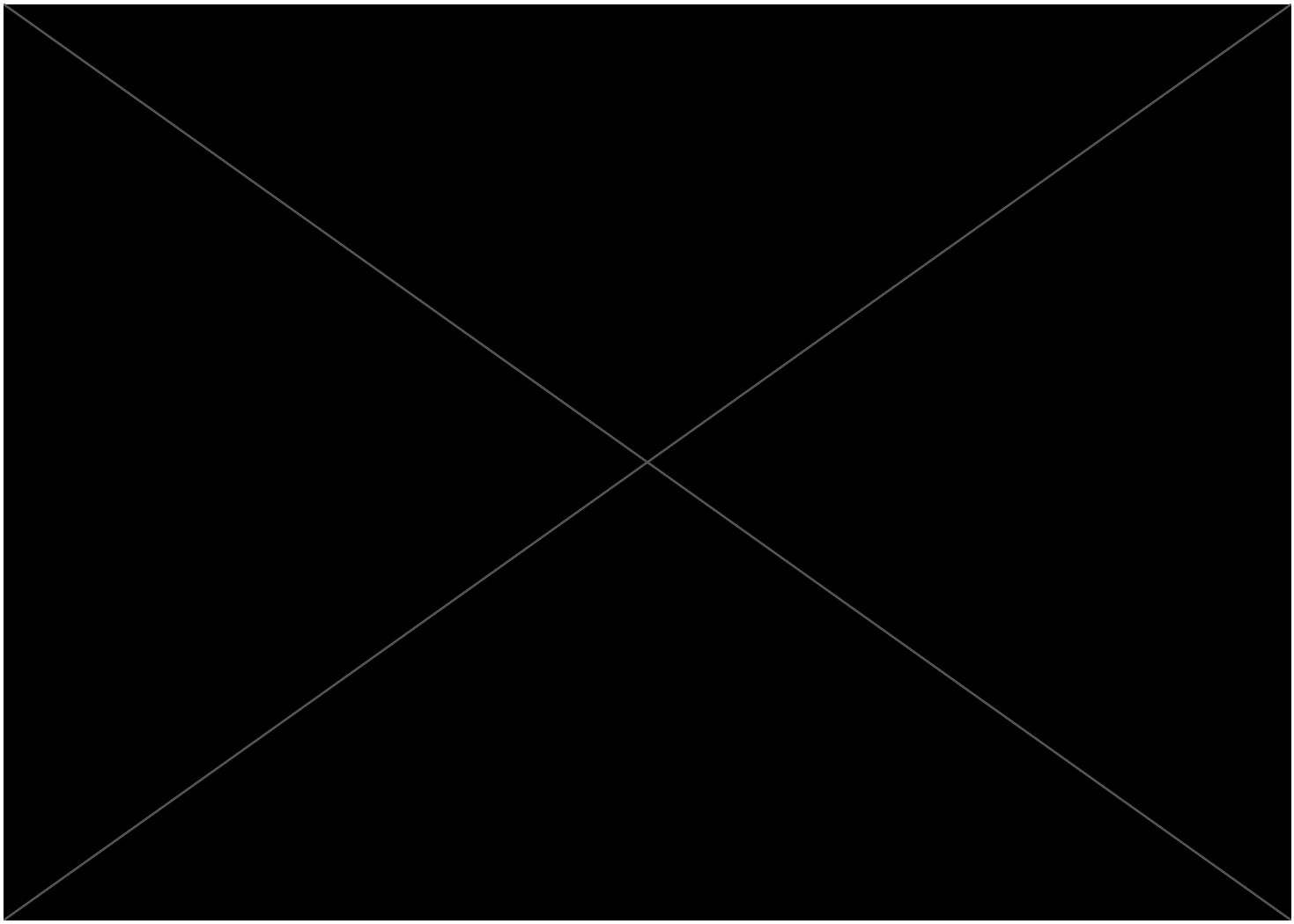
SOFTWARE DEVELOPMENT

Do We Need Object Oriented Programming in Data Science?

Let's discuss some pros and cons of switching to object oriented programming as a data scientist.

Rose Day · Follow
Published in Towards Data Science
4 min read · Oct 13, 2020



When I first got into data science, it became clear that object oriented programming was not the main focus. Many data scientists wrote code in notebooks or as single Python scripts to clean data, develop models, and run them but did not put the code into functions or classes. Most classes center around understanding machine learning (ML) models, feature engineering, training/testing/validation sets, and more. The work was commonly done in a notebook environment and shared around between students and professors. Often, data scientists do not consider the end-user in the same manner as a developer during software development.

When I began working as a data engineer, there was more focus on object oriented languages and cloud technologies to host, clean, and provide the data to other teams. Even then, most of the programming lived in Lambda functions on AWS or using open source libraries to do the necessary work. By the time I switched into a data science role, I was back to long or chained notebooks, with some functions, depending on the developer. There was minimal object oriented code, and that was only there to interact with other teams who ingested our work and required it for running the code.

Now I work on a team whose main codebase is almost all object oriented alongside other libraries and tools that follow similar coding standards. There is a mix of notebooks, libraries, and automation with testing pipelines and release pipelines. Shifting towards an object oriented mindset has been very beneficial. This shift has allowed our code to be production-ready, easily readable, and extendable to new use-cases.

Why Consider Object Oriented Programming?

When writing code in an object oriented manner, you focus on functions, inheritance, methods, and classes. These constructs can build applications or tools, create extendable and reusable libraries, and run repeatable analyses and reports. If you want to develop applications that utilize your analytic models, these applications will need to interface with your code, run it, and produce the same results as you produce in your local development environment. As you are developing and refining out your models, you can create extendable libraries for other applications or other teams to consume.

Learning how to implement object oriented programming will help you write classes, create objects, and execute complex programs. As you learn to read and write code in that manner, you can increase your productivity and efficiency by reusing code that has already been written once. It will also enable you to better utilize others' code, especially in open source libraries that you may want to extend to include your use cases.

Let's say a teammate has already written a library of data cleaning functions that they commonly use on your team's datasets. Through the creation of this library, you can clone down their work and utilize it. This library means you will now have access to the same functions that cleaned their data, making it easier for you to reproduce their work. Now, each teammate who wants to clean the data in the same manner can just run that same code, and there will no longer be slight differences in the data cleaning. You can apply this same concept to creating libraries for data ingestion, data cleaning, developing analytics, creating standard visualizations, and more. If your team begins to use the same set of conventional processes for their analyses, you can speed up your time to provide useful insights.

Pros and Cons of Object Oriented Programing

With that, there are pros and cons to using object oriented programming in your data science work.

Pros

- Object oriented code Allows for reusable and extendable code. You can use these functions and classes in repeated analyses, or create new projects that utilize some of the existing code, reducing time to produce results.
- It can be much easier to spot bugs if you write clean, object oriented code.
- It makes it easier to test and debug code when the pieces are more modular and broken down.
- Object oriented programming allows for parallel development between multiple data scientists who want to work with the same codebase for their projects.

Cons

- There is a possibility of the code base becoming very large, with many unneeded parts.
- You can run into duplication of code during the implementation of a project.
- There can be a learning curve when starting to utilize object oriented programming and may take some time to get used to.
- It can increase the time to run the code as there are typically more lines of code to execute.

Summary

In the end, it is up to you if you need to switch to object oriented programming, but it can provide many benefits. The main benefit I have found in this programming style is the ability to reuse and extend the code base quickly when starting a new project. A developed codebase like this allows for sharing between developers who can pick up your work and expand their use-cases on top of it. *And who knows, they may even find some inspiration when looking at what you have created.*

Additional Reading

- "A Simple Guide to Object Oriented Programming for Data Scientists" by Thu Vu
- "How a simple mix of object-oriented programming can sharpen your deep learning prototype" by Tirthajyoti Sarkar
- "Do You Need to Learn OOP for a Job in Data Analytics?" by Rohit Yadav
- "Object Oriented Programming (OOP) & Functional Programming — What are they & the Pros and Cons" by Darrick Mckirnan

Data Science | Software Development | Technology | Machine Learning | Programming