Foreword

When Henry Ford's company built its first moving assembly line in 1913 to produce its legendary Model T, it cut the time it took to build each car from 12 to 3 hours. This drastically reduced costs, allowing the Model T to become the first affordable automobile in history. It also made mass production possible: soon, roads were flooded with Model Ts.

Since the production process was now a clear sequence of well-defined steps (aka, a *pipeline*), it became possible to automate some of these steps, saving even more time and money. Today, cars are mostly built by machines.

But it's not just about time and money. For many repetitive tasks, a machine will produce much more consistent results than humans, making the final product more predictable, consistent, and reliable. Lastly, by keeping humans away from heavy machinery, safety is greatly improved, and many workers went on to perform higher-level jobs (although to be fair, many others just lost their jobs).

On the flip side, setting up an assembly line can be a long and costly process. And it's not ideal if you want to produce small quantities or highly customized products. Ford famously said, "Any customer can have a car painted any color that he wants, so long as it is black."

The history of car manufacturing has repeated itself in the software industry over the last couple of decades: every significant piece of software nowadays is typically built, tested, and deployed using automation tools such as Jenkins or Travis. However, the Model T metaphor isn't sufficient anymore. Software doesn't just get deployed and forgotten; it must be monitored, maintained, and updated regularly. Software pipelines now look more like dynamic loops than static production lines. It's crucial to be able to quickly update the software (or the pipeline itself) without ever breaking it. And software is much more customizable than the Model T

ever was: *software can be painted any color* (e.g., try counting the number of MS Office variants that exist).

Unfortunately, "classical" automation tools are not well suited to handle a full machine learning pipeline. Indeed, an ML model is not a regular piece of software.

For one, a large part of its behavior is driven by the data it trains on. Therefore, the training data itself must be treated as code (e.g., versioned). This is quite a tricky problem because new data pops up every day (often in large quantities), usually evolves and drifts over time, often includes private data, and must be labelled before you can feed it to supervised learning algorithms.

Second, the behavior of a model is often quite opaque: it may pass all the tests on some data but fail entirely on others. So you must ensure that your tests cover all the data domains on which your model will be used in production. In particular, you must make sure that it doesn't discriminate against a subset of your users.

For these (and other) reasons, data scientists and software engineers first started building and training ML models manually, "in their garage," so to speak, and many of them still do. But new automation tools have been developed in the past few years that tackle the challenges of ML pipelines, such as TensorFlow Extended (TFX) and Kubeflow. More and more organizations are starting to use these tools to create ML pipelines that automate most (or all) of the steps involved in building and training ML models. The benefits of this automation are mostly the same as for the car industry: save time and money; build better, more reliable, and safer models; and spend more time doing more useful tasks than copying data or staring at learning curves. However, building an ML pipeline is not trivial. So where should you start?

Well, right here!

In this book, Hannes and Catherine provide a clear guide to start automating your ML pipelines. As a firm believer in the hands-on approach, especially for such a technical topic, I particularly enjoyed the way this

book guides you step by step through a concrete example project from start to finish. Thanks to the many code examples and the clear, concise explanations, you should have your own ML pipeline up and running in no time, as well as all the conceptual tools required to adapt these ML pipelines to your own use cases. I highly recommend you grab your laptop and actually try things out as you read; you will learn much faster.

I first met Hannes and Catherine in October 2019 at the TensorFlow World conference in Santa Clara, CA, where I was speaking on building ML pipelines using TFX. They were working on this book on the same topic, and we shared the same editor, so naturally we had a lot to talk about. Some participants in my course had asked very technical questions about TensorFlow Serving (which is part of TFX), and Hannes and Catherine had all the answers I was looking for. Hannes even kindly accepted my invitation to give a talk on advanced features of TensorFlow Serving at the end of my course on very short notice. His talk was a treasure trove of insights and helpful tips, all of which you will find in this book, along with many, many more.

Now it's time to start building professional ML pipelines!

Aurélien Géron

Former YouTube Video Classification Team Lead

Author of Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (O'Reilly)

Auckland, New Zealand, June 18, 2020