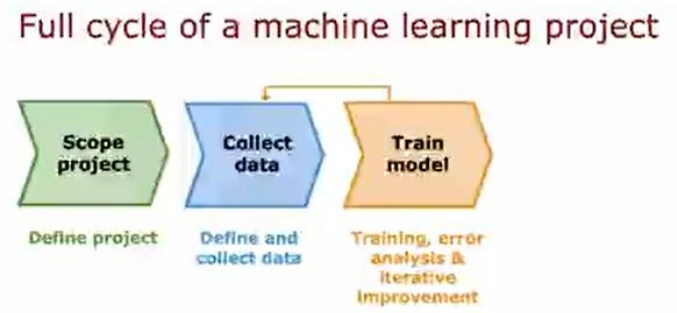
Complete cycle of a machine learning project

So far we've talked a lot about how to train a model and also talked a bit about how to get data for your machine learning application. But when I'm building a machine learning system I find that training a model is just part of the puzzle. In this video I'd like to share with you what I think of as the full cycle of a machine learning project.

That is, when you're building a valuable machine learning system, what are the steps to think about and plan for? Let's take a look, let me use speech recognition as an example to illustrate the full cycle of a machine learning project. The first step of machine learning project is to scope the project. In other words, decide what is the project and what you want to work on.

For example, I once decided to work on speech recognition for voice search. That is to do web search using speaking to your mobile phone rather than typing into your mobile phone. This project scoping. After deciding what to work on you have to collect data. Decide what data you need to train your machine learning system and go and do the work to get the audio and get the transcripts of the labels for your dataset. That's data collection. After you have your initial data collection you can then start to train the model.



Here you would train a speech recognition system and carry out error analysis and iteratively improve your model. Is not at all unusual. After you've started training the model for error analysis or for a bias-variance analysis to tell you that you might want to go back to collect more data.

Maybe collect more data of everything or just collect more data of a specific type where your error analysis tells you you want to improve the performance of your learning algorithm. For example, once when working on speech I realized that my model was doing particularly poorly when there was car noise in the background.

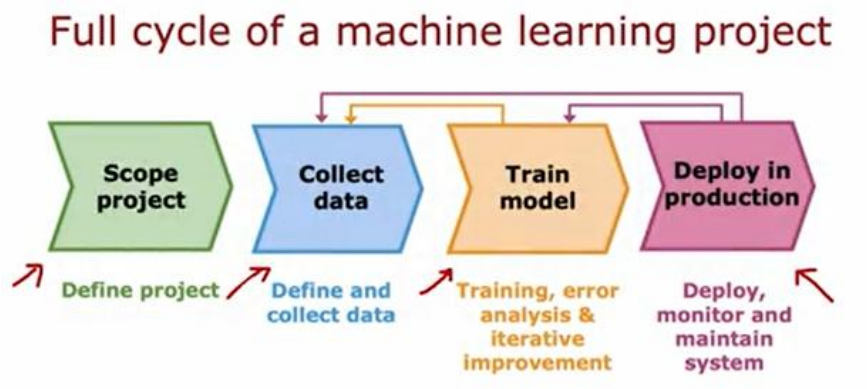
That sounded like someone was speaking in a car. My speech system perform poorly decided to get more data, actually using data augmentation to get more speech data that sounds like it was a car in order to improve the performance of my learning algorithm.

You go around this loop a few times, train the model, error analysis, go back to collect more data, maybe do this for a while until eventually you say the model is good enough to then deploy in a production environment. What that means is you make it available for users to use.

When you deploy a system you have to also make sure that you continue to monitor the performance of the system and to maintain the system in case the performance gets worse to bring us performance back up instead of just hosting your machine learning model on a server.

I'll say a little bit more about why you need to maintain these machine learning systems on the next slide. But after this deployment, sometimes you realize that is not working as well as you hoped and you go back to train the model to improve it again or even go back and get more data.

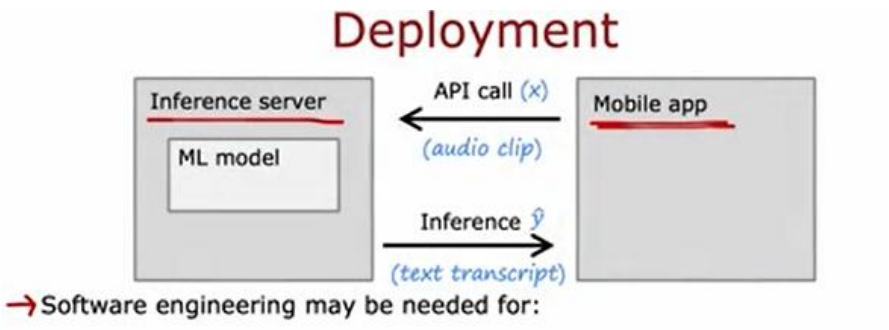
In fact, if users and if you have permission to use data from your production deployment, sometimes that data from your working speech system can give you access to even more data with which to keep on improving the performance of your system. Now, I think you have a sense of what scoping a project means and we've talked a bunch about collecting data and training models in this course.



about what deploying in production might look like. After you've trained a high performing machine learning model, say a speech recognition model, a common way to deploy the model would be to take your machine learning model and implement it in a server, which I'm going to call an inference server, whose job it is to call your machine learning model, your trained model, in order to make predictions.

Then if your team has implemented a mobile app, say a search application, then when a user talks to the mobile app, the mobile app can then make an API call to pass to your inference server the audio clip that was recorded and the inference server's job is supply the machine learning model to it and then return to it the prediction of your model, which in this case would be the text transcripts of what was said.

This would be a common way of implementing an application that calls via the API and inference server that has your model repeatedly make predictions based on the input, x. These were common pattern where depend on the application does implemented. You have an API call to give your learning algorithm the input, x, and your machine learning model within output to prediction, say y hat. To implement this some software engineering may be needed to write all the code that does all of these things.

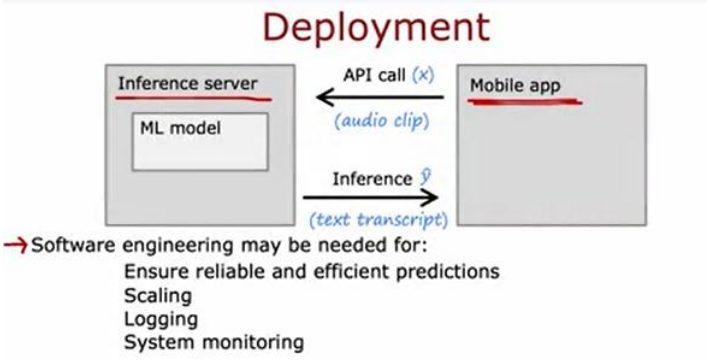


Depending on whether your application needs to serve just a few handful of users or millions of users the amounts of software engineer needed can be quite different. I've build software that

serves just a handful of users on my laptop and I've also built software that serves hundreds of millions of users requiring significant data center resources.

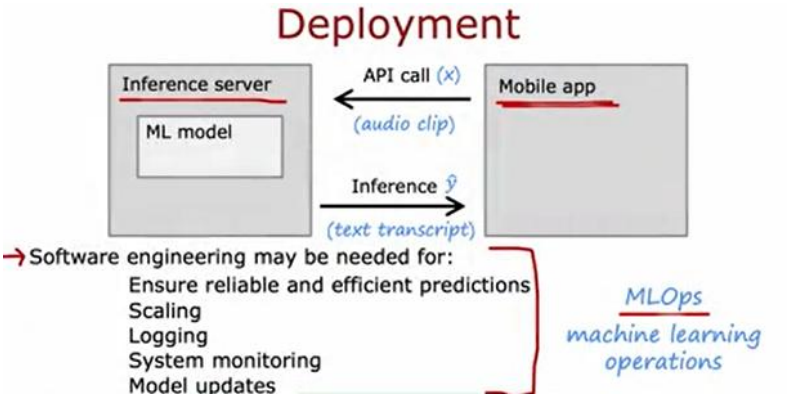
Depending on scale application needed, software engineering may be needed to make sure that your inference server is able to make reliable and efficient predictions hopefully not too high of computational cost. Software engineering may be needed to manage scaling to a large number of users. You often want to log the data you're getting both the inputs, x, as well as the predictions, y hat, assuming that user privacy and consent allows you to store this data.

This data, if you can access it, is also very useful for system monitoring. For example, I once built a speech recognition system on a certain dataset that I had but when there were new celebrities that suddenly became well-known or elections cause new politicians to become elected and people will search for these new names that were not in the training set and then my system did poorly on.



It was because we were monitoring the system allowed us to figure out when the data was shifting and the algorithm was becoming less accurate. This allowed us to retrain the model and then to carry out a model update to replace the old model with a new one. The deployment process can require some amounts of software engineering.

For some applications, if you're just running it on a laptop or on a one or two servers, maybe not that much software engineering is needed. Depending on the team you're working on, it is possible that you built the machine learning model but there could be a different team responsible for deploying it. But there is a growing field in machine learning called MLOps. This stands for Machine Learning Operations.



This refers to the practice of how to systematically build and deploy and maintain machine learning systems. To do all of these things to make sure that your machine learning model is reliable, scales well, has good laws, is monitored, and then you have the opportunity to make updates to the model as appropriate to keep it running well. For example, if you are deploying your system to millions of

people you may want to make sure you have a highly optimized implementations so that the compute cost of serving millions of people is not too expensive.

In this and the last class I spent a lot of time talking about how to train a machine learning model and got this absolutely the critical piece to making sure you have a high performance system. If you ever have to deploy system to millions of people, these are some additional steps that you probably have to address. Think about the [inaudible] at that point as well.

Before moving on from the topic of the machine learning development process, there's one more set of ideas that I want to share with you that relates to the ethics of building machine learning systems.