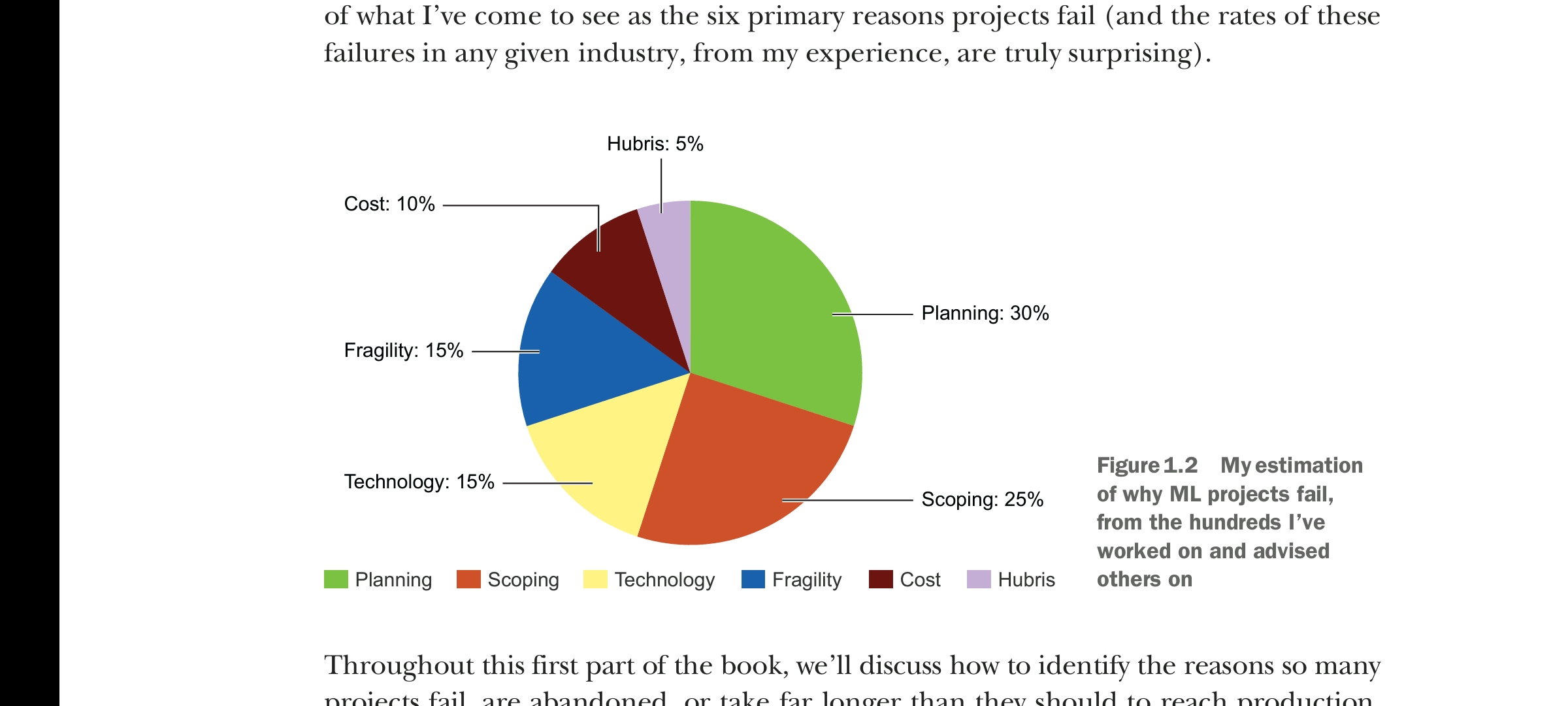
**Por que Engenharia de Machine Learning?**

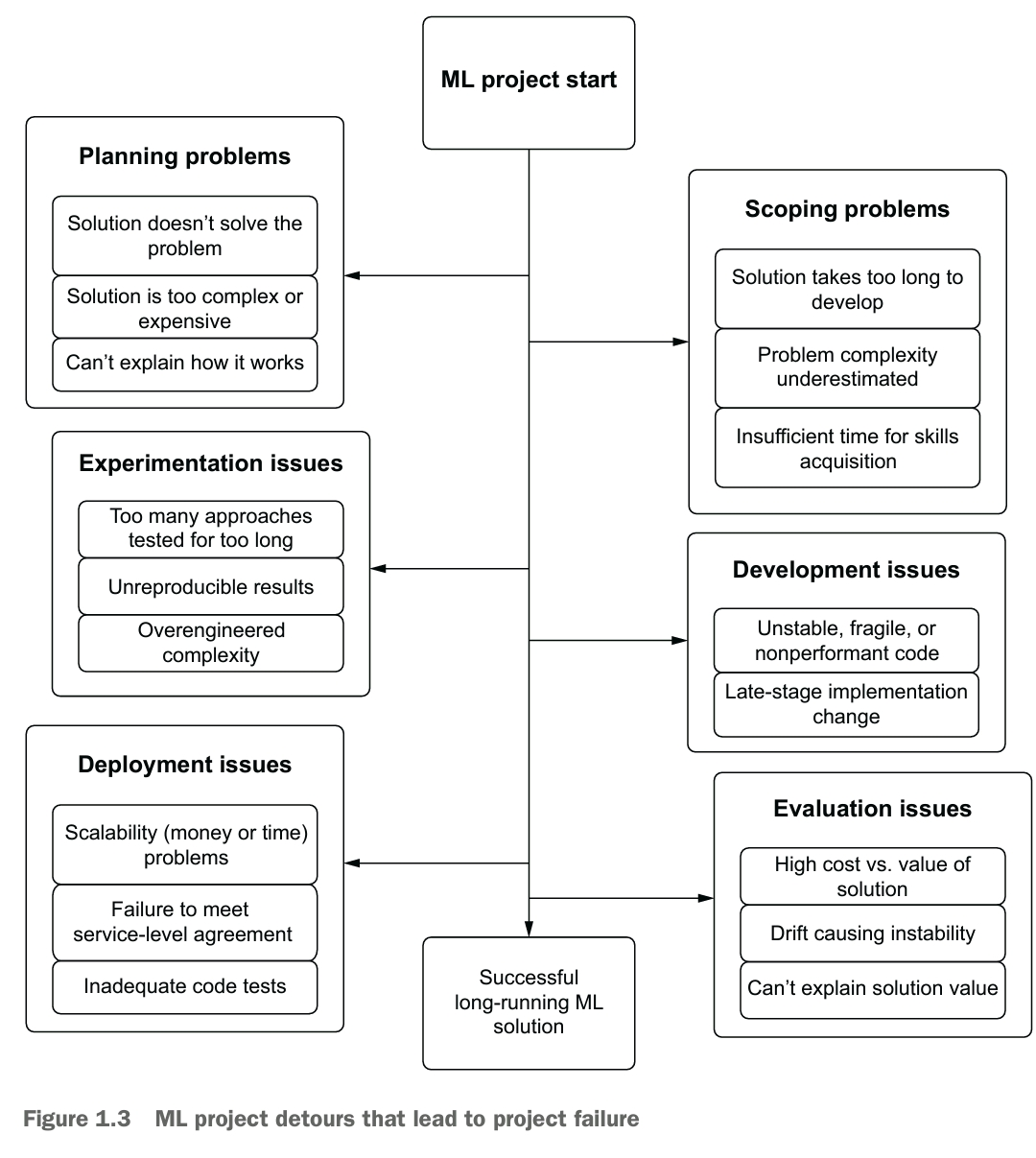
Os engenheiros de ML precisam saber habilidades de desenvolvimento de software suficientes para serem capazes de escrever códigos modulares e implementar testes unitários. Não precisa saber sobre as complexidades da intermediação de mensagens assíncronas sem bloqueio. Eles precisam de habilidades de engenharia de dados suficientes para criar (e agendar o ETL para) conjuntos de dados de recursos para seus modelos, mas não para construir uma estrutura de ingestão de streaming em escala de petabyte. Eles precisam de habilidades de visualização suficientes para criar gráficos e gráficos que comuniquem claramente o que suas pesquisas e modelos estão fazendo, mas não para desenvolver aplicativos da web dinâmicos que tenham componentes complexos do UserExperience (UX). Eles também precisam de experiência suficiente em gerenciamento de projetos para saber como definir, escopo e controlar adequadamente um projeto para resolver um problema, mas não precisam passar por uma certificação Project Management Professional (PMP). Um elefante gigante permanece na sala quando se trata de ML. Especificamente, por que – com tantas empresas participando de ML, contratando grandes equipes de cientistas de dados altamente compensados e dedicando enormes recursos financeiros e temporais a projetos – muitos empreendimentos acabam falhando? A Figura 1.2 mostra estimativas aproximadas do que passei a ver como as seis principais razões pelas quais os projetos falham (e as taxas dessas falhas em qualquer indústria, por minha experiência, são realmente surpreendentes).



Minha estimativa de por que os projetos de ML falham, desde as centenas em que trabalhei e em que aconselhei outras pessoas.

Geralmente, essas falhas acontecem porque a equipe do DS não tem experiência em resolver um problema da escala necessária (uma falha tecnológica ou orientada por processos) ou não entendeu completamente o resultado desejado dos negócios (uma falha orientada por comunicação). Eu nunca vi isso acontecer por causa de uma intenção maliciosa. Em vez disso, a maioria dos projetos de ML são incrivelmente desafiadores, complexos e compostos por ferramentas de software algorítmica que são difíceis de explicar a um leigo – daí as falhas na comunicação com as unidades de negócios que a maioria dos projetos dura.

Somando-se à complexidade dos projetos de ML, existem dois outros elementos críticos que não são compartilhados pela (a maioria) dos projetos tradicionais de desenvolvimento de software: uma frequente falta de detalhes nas expectativas do projeto e a relativa imaturidade da indústria nas ferramentas. Ambos os aspectos não são diferentes do estado da engenharia de software no início dos anos 1990. As empresas então não tinham certeza de como aproveitar melhor os novos aspectos da capacidade tecnológica, as ferramentas foram lamentavelmente subdesenvolvidas e muitos projetos não atenderam às expectativas daqueles que estavam embarcando em equipes de engenharia para construí-los. O trabalho de ML está (da minha visão tendenciosa de trabalhar apenas com tantas empresas) no mesmo lugar agora na segunda década do século 21 que a engenharia de software estava há 30 anos. Este livro não é um tratado prejudicial aos desafios da ML. Em vez disso, pretende mostrar como esses elementos podem ser um risco para projetos. A intenção é ensinar os processos e ferramentas que ajudam a minimizar esse risco de falha. A Figura 1.3 mostra uma visão geral dos desvios que podem surgir na execução de um projeto. Cada um traz um elemento de risco diferente para a execução bem-sucedida de um projeto.



**Problemas de planejamento**

A solução não resolve o problema

A solução é muito complexa ou cara

Não pode explicar como funciona

**Problemas de escopo**

A solução demora muito para desenvolver

A complexidade do problema subestimada

O tempo insuficiente para a aquisição de habilidades

**Desafios da experimentação**

Muitas abordagens testadas por um tempo longo

Resultados não reproduzíveis

Complexidade excessiva

**Desafios de desenvolvimento**

Mudança de implementação em estágio avançado

Código instável, frágil ou não-performativo

**Problemas de implantação**

Escalabilidade (dinheiro ou tempo)

Falha em cumprir o contrato de nível de serviço

Testes de código inadequados

**Problemas de avaliação**

Alto custo versus valor da solução

Deriva causando instabilidade

Não conseguir explicar o valor da solução.

A estrutura usada na engenharia de ML é exatamente dedicada a abordar cada um desses modos de falha primários. A eliminação dessas chances de falha está no centro dessa metodologia. Isso é feito fornecendo os processos para tomar melhores decisões, facilitar a comunicação com os clientes internos, eliminar o retrabalho durante as fases de experimentação e desenvolvimento, criar bases de código que podem ser facilmente mantidas e trazer uma abordagem de práticas recomendadas para qualquer projeto que seja fortemente influenciado pelo trabalho do DS. Assim como engenheiros de software, décadas atrás, refinaram seus processos, desde implementações em cascata em grande escala até um processo ágil mais flexível e produtivo. A ML Engineering busca definir um novo conjunto de práticas e ferramentas que otimizarão o domínio totalmente exclusivo do desenvolvimento de software para cientistas de dados.