

O'REILLY®

Practical MLOps

Operationalizing Machine Learning Models



Noah Gift & Alfredo Deza

VERSÃO
PORTUGUESA

MLOps práticos

Operacionalização de modelos de aprendizagem
automática

Noah Gift e Alfredo Deza



Beijing • Boston • Farnham • Sebastopol • Tokyo

MLOps práticos

por Noah Gifte Alfredo Deza

Direitos de autor © 2021 Noah Gift e Alfredo Deza. Todos os direitos reservados.

Impresso nos Estados Unidos da América.

Publicado por O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Os livros da O'Reilly podem ser adquiridos para uso educacional, comercial ou promocional de vendas. As edições on-line também estão disponíveis para a maioria dos títulos(<http://oreilly.com>). Para mais informações, contacta o nosso departamento de vendas empresariais/institucionais: 800-998-9938 ou *corporate@oreilly.com*.

Editora de aquisições:Rebecca Novack

Editora de desenvolvimento:Melissa Potter

Editor de produção:Daniel Elfanbaum

Redator:Kim Cofer

Revisor:Piper Editorial Consulting, LLC

Indexador:WordCo Indexing Services, Inc.

Designer de interiores:David Futato

Desenhador da capa:Karen Montgomery

Ilustrador:Kate Dullea

setembro de 2021:Primeira edição

Histórico de revisões da primeira edição

- 2021-09-14:Primeiro lançamento

Consulta <http://oreilly.com/catalog/errata.csp?isbn=9781098103019>para obteres informações sobre o lançamento.

O logótipo O'Reilly é uma marca registada da O'Reilly Media, Inc. *Practical MLOps*, a imagem da capa e a imagem comercial relacionada são marcas comerciais da O'Reilly Media, Inc.

As opiniões expressas nesta obra são as dos autores e não representam as opiniões da editora. Embora a editora e os autores tenham evidiado esforços de boa fé para garantir que as informações e instruções contidas nesta obra são exactas, a editora e os autores declinam qualquer responsabilidade por erros ou omissões, incluindo, sem limitação, a responsabilidade por danos resultantes da utilização ou confiança nesta obra. A utilização das informações e instruções contidas nesta obra é feita por tua conta e risco. Se qualquer amostra de código ou outra tecnologia que esta obra contenha ou descreva estiver sujeita a licenças de código aberto ou a direitos de propriedade intelectual de terceiros, é da tua responsabilidade garantir que a tua utilização está em conformidade com essas licenças e/ou direitos.

978-1-098-10301-9

[LSI]

Prefácio

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Porque escrevemos este livro

Ambos passámos a maior parte das nossas carreiras a automatizar coisas. Quando nos conhecemos e o Alfredo não sabia Python, o Noah sugeriu-nos que automatizássemos uma tarefa por semana. A automação é um pilar fundamental para MLOps, DevOps e para este livro. Deves considerar todos os exemplos e opiniões deste livro no contexto da automatização futura.

Se Noah pudesse resumir a forma como passou os anos 2000-2020, foi a automatizar praticamente tudo o que podia, desde condutas de filmes a instalação de software e condutas de aprendizagem automática. Como gestor de engenharia e CTO em startups na Bay Area, construiu muitas equipas de ciência de dados a partir do zero. Como resultado, viu muitos dos principais problemas na colocação da aprendizagem automática em produção nas fases iniciais da revolução da IA/ML.

Nos últimos anos, Noah tem sido professor adjunto na Duke, Northwestern e UC Davis, ensinando tópicos que se centram principalmente na computação Cloud, ciência de dados e engenharia de aprendizagem automática. Esta experiência de ensino e de trabalho dá-lhe uma perspetiva única sobre as questões envolvidas na implementação de soluções de aprendizagem automática no mundo real.

Alfredo tem uma sólida experiência em operações desde os seus tempos de administrador de sistemas, com uma paixão semelhante pela automatização. Não é possível construir uma infraestrutura resiliente sem a automação de

botões. Não há nada mais gratificante quando acontecem situações de desastre do que voltar a executar um script ou um pipeline para recriar o que falhou.

Quando a COVID-19 chegou, acelerou uma questão que ambos tínhamos, que era: "porque não estamos a pôr mais modelos em produção?" Noah abordou algumas destas questões num [artigo que escreveu para a Forbes](#). A premissa resumida do artigo é que algo está errado com a ciência dos dados porque as organizações não estão a ver o retorno dos seus investimentos.

Mais tarde, no ["Foo Camp" da O'Reilly](#), Noah liderou uma sessão sobre "Por que não podemos ser 10 vezes mais rápidos em ML na produção?", onde tivemos uma grande discussão com muitas pessoas, incluindo Tim O'Reilly, Mike Loukides, Roger Magoulas e outros. O resultado dessa discussão foi: "Sim, podemos ser 10 vezes mais rápidos". Por isso, obrigado ao Tim e ao Mike por terem suscitado uma discussão tão fascinante e por terem posto este livro a caminho.

A aprendizagem automática assemelha-se a muitas outras tecnologias que surgiram nas últimas décadas. No início, demora anos a obter resultados. Steve Jobs falou sobre como a NeXT queria tornar 10 vezes mais rápido o desenvolvimento de software (e conseguiu). Podes ver a entrevista no [YouTube](#). Quais são alguns dos problemas com a aprendizagem automática atualmente?

- Concentra-te no "código" e nos detalhes técnicos em vez de te concentrares no problema comercial
- Falta de automatização
- HiPPO (Highest Paid Person's Opinions - Opiniões da pessoa mais bem paga)
- Não é nativo da Cloud
- Falta de urgência para resolver problemas solucionáveis

Citando uma das coisas que o Noah referiu no debate: "Sou anti-elitismo em todos os sectores. A programação é um direito humano. A ideia de que

há um sacerdócio que só pode fazer isso é simplesmente errada". Tal como a aprendizagem automática, é demasiado importante que a tecnologia esteja apenas nas mãos de um grupo selecionado de pessoas. Com o MLOps e o AutoML, estas tecnologias podem ir para as mãos do público. Podemos fazer melhor com a aprendizagem automática e a inteligência artificial, democratizando esta tecnologia. Os profissionais "reais" de IA/ML enviam modelos para a produção e, no futuro "real", pessoas como médicos, advogados, mecânicos e professores utilizarão a IA/ML para os ajudar a fazer o seu trabalho.

Como este livro está organizado

Concebemos este livro para que possas consumir cada capítulo como uma secção autónoma concebida para te dar ajuda imediata. No final de cada capítulo, há perguntas para discussão que visam estimular o pensamento crítico e exercícios técnicos para melhorar a tua compreensão do material.

Estas questões e exercícios de discussão também são adequados para utilização na sala de aula de um programa de Ciência de Dados, Ciência da Computação ou MBA e para o auto-aprendiz motivado. O capítulo final contém vários estudos de caso úteis para a construção de um portfólio de trabalho como especialista em MLOps.

O livro está dividido em 12 capítulos, que analisaremos um pouco mais na secção seguinte. No final do livro, há um apêndice com uma coleção de recursos valiosos para a implementação de MLOps.

Capítulos

Os primeiros capítulos cobrem a teoria e a prática de DevOps e MLOps. Um dos itens abordados é como configurar a integração contínua e a entrega contínua. Outro tópico crítico é o Kaizen, ou seja, a ideia de melhoria contínua em tudo.

Há três capítulos sobre computação em Cloud que abrangem a AWS, o Azure e o GCP. Alfredo, um defensor dos programadores da Microsoft, é

uma fonte ideal de conhecimentos para os MLOps sobre a plataforma Azure. Da mesma forma, Noah passou anos a formar estudantes em computação em nuvem e a trabalhar com os departamentos de educação da Google, AWS e Azure. Estes capítulos são uma excelente forma de te familiarizares com os MLOps baseados na Cloud.

Outros capítulos abrangem áreas técnicas críticas dos MLOps, incluindo AutoML, contentores, computação de ponta e portabilidade de modelos. Estes tópicos englobam muitas tecnologias emergentes de ponta com tração ativa.

Por fim, no último capítulo, Noah apresenta um estudo de caso real sobre o tempo que passou numa startup de redes sociais e os desafios que enfrentaram com os MLOps.

Apêndices

Os apêndices são uma coleção de ensaios, idéias e itens valiosos que surgiram nos anos entre a finalização do *Python for DevOps* (O'Reilly) e este livro. A principal maneira de usá-los é para ajudar-te a tomar decisões sobre o futuro.

Perguntas de exercício

Nos exercícios deste livro, uma heurística útil considera como os podes aproveitar num portefólio utilizando o GitHub e um passo a passo no YouTube do que fizeste. De acordo com a expressão "uma imagem vale mais que mil palavras", um link do YouTube para um passo a passo de um projeto reproduzível do GitHub em um currículo pode valer 10.000 palavras e coloca o currículo em uma nova categoria de qualificação para um emprego.

Ao leres o livro e os exercícios, considera o seguinte quadro de pensamento crítico.

Perguntas para discussão

De acordo com Jonathan Haber, em *Critical Thinking* (série MIT Press Essential Knowledge) e com a organização sem fins lucrativos **Foundation for Critical Thinking**, as perguntas para discussão são componentes essenciais do pensamento crítico. O mundo precisa urgentemente de pensamento crítico devido à proliferação de desinformação e de conteúdos superficiais nas redes sociais. O domínio das seguintes competências distingue um indivíduo do resto:

Humildade intelectual

Reconhecimento dos limites do teu conhecimento.

Coragem intelectual

A capacidade de defender as tuas convicções mesmo perante a pressão social.

Empatia intelectual

A capacidade de te colocares na mente dos outros para compreenderes a sua posição.

Autonomia intelectual

A capacidade de pensares por ti próprio, independentemente dos outros.

Integridade intelectual

A capacidade de pensar e argumentar com os mesmos padrões intelectuais que esperas que os outros apliquem a ti.

Perseverança intelectual

A capacidade de apresentar provas que apoiem a tua posição.

Confiança na razão

A crença de que existem factos indiscutíveis e que a razão é a melhor solução para obter conhecimento.

Equidade

A capacidade de fazer um esforço de boa-fé para tratar todos os pontos de vista de forma justa.

Utilizando estes critérios, avalia as questões de debate de cada capítulo.

Origem das citações do capítulo

Por Noah

Terminei a faculdade no final de 1998 e passei um ano a treinar para jogar basquetebol profissional nas ligas menores dos Estados Unidos ou da Europa, enquanto trabalhava como treinador pessoal. O meu plano de reserva era arranjar um emprego em TI. Candidatei-me a administrador de sistemas na Caltech em Pasadena e consegui um lugar de especialista em TI para Mac por acaso. Decidi que a relação risco/recompensa de ser um atleta profissional mal pago não valia a pena e aceitei a oferta de emprego.

Dizer que o Caltech mudou a minha vida é um eufemismo. Ao almoço, jogava frisbee e ouvi falar da linguagem de programação Python, que aprendi para me "integrar" com os meus amigos do frisbee, que eram funcionários ou estudantes do Caltech. Mais tarde, trabalhei diretamente para a administração do Caltech e fui o especialista pessoal em Mac no Caltech para o Dr. David Baltimore, que recebeu o Prémio Nobel aos 30 anos. Interagi com muitas pessoas famosas de formas inesperadas, o que aumentou a minha auto-confiança e fez crescer a minha Network+.

Também tive muitos encontros ao acaso, ao estilo Forrest Gump, com pessoas que mais tarde viriam a fazer coisas incríveis em IA/ML. Uma vez, jantei com a Dra. Fei-Fei Li, diretora de IA em Stanford, e o namorado dela; lembro-me de ter ficado impressionado por o namorado dela ter passado o verão a escrever um jogo de vídeo com o pai. Fiquei muito impressionado e pensei: "Quem é que faz este tipo de coisas?" Mais tarde, instalei um servidor de correio eletrónico debaixo da secretaria do famoso físico Dr. David Goodstein, porque ele estava sempre a ser incomodado

pelo departamento de TI por ter atingido o limite de armazenamento da sua caixa de correio. Foi nestas experiências que adquiri o gosto pela construção de "infra-estruturas sombra". Como trabalhava diretamente para a administração, podia desrespeitar as regras se houvesse uma boa razão para isso.

Uma das pessoas que conheci por acaso foi o Dr. Joseph Bogen, neurocirurgião e professor visitante do Caltech. De todas as pessoas que conheci no Caltech, ele teve o impacto mais profundo na minha vida. Um dia, respondi a uma chamada do serviço de assistência para ir a casa dele arranjar o computador e, mais tarde, isso transformou-se num jantar semanal em sua casa com ele e a mulher, Glenda. Desde cerca de 2000 até ao dia da sua morte, foi um amigo e um mentor.

Na altura, estava muito interessado em inteligência artificial e lembro-me de um professor de Informática do Caltech me dizer que era um campo morto e que não me devia concentrar nele. Apesar desse conselho, elaborei um plano para ser fluente em muitas linguagens de programação de software aos 40 anos e escrever programas de inteligência artificial nessa altura. Olha e vê, o meu plano resultou.

Posso dizer claramente que não estaria a fazer o que faço hoje se não tivesse conhecido Joe Bogen. Ele deixou-me boquiaberto quando me disse que tinha feito a primeira hemisferectomy, removendo metade de um cérebro, para ajudar um doente com epilepsia grave. Falávamos durante horas sobre as origens da consciência, a utilização de redes neurais nos anos 70 para descobrir quem seria piloto da Força Aérea e se o teu cérebro continha "dois de ti", um em cada hemisfério. Acima de tudo, o que Bogen me deu foi um sentimento de confiança no meu intelecto. Até então, eu tinha sérias dúvidas sobre o que podia fazer, mas as nossas conversas foram como um mestrado em pensamento de alto nível. Como professor, penso no grande impacto que ele teve na minha vida e espero retribuir isso a outros estudantes com quem interajo, quer como professor formal, quer como alguém que eles conhecem. Podes ler estas citações num arquivo da [página inicial do Dr. Bogen no Caltech](#) e na sua [biografia](#).

Convenções utilizadas neste livro

Neste livro são utilizadas as seguintes convenções tipográficas:

Itálico

Indica novos termos, URLs, endereços de e-mail, nomes de ficheiros e extensões de ficheiros.

Constant width

Usado para listagens de programas, bem como dentro de parágrafos para se referir a elementos do programa, como nomes de variáveis ou funções, bases de dados, tipos de dados, variáveis de ambiente, instruções e palavras-chave.

Constant width bold

Mostra comandos ou outro texto que deve ser digitado literalmente pelo utilizador.

Constant width italic

Mostra o texto que deve ser substituído por valores fornecidos pelo utilizador ou por valores determinados pelo contexto.

DICA

Este elemento significa uma dica ou sugestão.

NOTA

Este elemento significa uma nota geral.

AVISO

Este elemento indica um aviso ou precaução.

Utilizar exemplos de código

O material suplementar (exemplos de código, exercícios, etc.) está disponível para transferência em <https://github.com/paiml/practical-mlops-book>.

Se tiveres uma questão técnica ou um problema ao utilizar os exemplos de código, envia um e-mail para bookquestions@oreilly.com.

Este livro está aqui para te ajudar a fazer o teu trabalho. Em geral, se for fornecido código de exemplo com este livro, podes utilizá-lo nos teus programas e documentação. Não precisas de nos contactar para obter permissão, a menos que estejas a reproduzir uma parte significativa do código. Por exemplo, escrever um programa que use vários trechos de código deste livro não requer permissão. A venda ou distribuição de exemplos dos livros da O'Reilly requer permissão. Responder a uma pergunta citando este livro e citando um exemplo de código não requer permissão. Incorporar uma quantidade significativa de código de exemplo deste livro na documentação do teu produto requer permissão.

Apreciamos, mas geralmente não exigimos, a atribuição. Uma atribuição inclui normalmente o título, o autor, a editora e o ISBN. Por exemplo: "MLOps práticos por Noah Gift e Alfredo Deza (O'Reilly). Copyright 2021 Noah Gift e Alfredo Deza, 978-1-098-10301-9."

Se considerares que a tua utilização de exemplos de código não se enquadra na utilização justa ou na permissão dada acima, não hesites em contactar-nos em permissions@oreilly.com.

Aprendizagem em linha da O'Reilly

NOTA

Há mais de 40 anos que *a O'Reilly Media* fornece formação em tecnologia e negócios, conhecimentos e ideias para ajudar as empresas a ter sucesso.

A nossa rede única de especialistas e inovadores partilha os seus conhecimentos e experiência através de livros, artigos e da nossa plataforma de aprendizagem online. A plataforma de aprendizagem online da O'Reilly dá-te acesso a cursos de formação em direto, percursos de aprendizagem aprofundados, ambientes de codificação interactivos e uma vasta coleção de textos e vídeos da O'Reilly e de mais de 200 outras editoras. Para obter mais informações, visita <http://oreilly.com>.

Como contactar-nos

Por favor, dirige os teus comentários e perguntas sobre este livro ao editor:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (nos Estados Unidos ou no Canadá)

707-829-0515 (internacional ou local)

707-829-0104 (fax)

Temos uma página web para este livro, onde listamos erratas, exemplos e qualquer informação adicional. Podes aceder a esta página em <https://oreil.ly/practical-mlops>.

Envia um e-mail para bookquestions@oreilly.com para fazeres comentários ou perguntas técnicas sobre este livro.

Para notícias e informações sobre os nossos livros e cursos, visita <http://oreilly.com>.

Encontra-nos no Facebook: <http://facebook.com/oreilly>.

Segue-nos no Twitter: <http://twitter.com/oreillymedia>.

Vê-nos no YouTube: <http://www.youtube.com/oreillymedia>.

Agradecimentos

De Noé

Como já referi, sem Mike Loukides me ter convidado para o Foo Camp e ter tido uma excelente discussão comigo e com Tim O'Reilly, este livro não estaria aqui. De seguida, gostaria de agradecer ao Alfredo, o meu coautor.

Tive o prazer de escrever cinco livros, dois para a O'Reilly e três auto-publicados, em pouco mais de dois anos com o Alfredo, e isso deve-se principalmente à sua capacidade de aceitar o trabalho e de fazer as coisas. A apetência pelo trabalho árduo é talvez o melhor talento, e o Alfredo tem essa competência em abundância.

A nossa editora, Melissa Potter, fez um trabalho tremendo para pôr as coisas em forma, e o livro antes de ela editar e depois são quase dois livros diferentes. Sinto-me sortuda por ter trabalhado com uma editora tão talentosa.

Os nossos editores técnicos, incluindo Steve Depp, Nivas Durairaj e Shubham Saboo, desempenharam um papel crucial ao darem-nos um excelente feedback sobre onde fazer ziguezagues e quando fazer zaguezagues. Muitos dos melhoramentos devem-se particularmente ao feedback minucioso do Steve. Além disso, gostaria de agradecer a Julien Simon e Piero Molino por melhorarem o nosso livro com pensamentos reais sobre MLOps.

Quero agradecer à minha família, Liam, Leah e Theodore, por me terem dado espaço para terminar este livro num prazo apertado, no meio de uma

pandemia. Também estou ansioso por ler alguns dos livros que eles escrevem no futuro. Outro grande grupo de agradecimentos vai para todos os antigos alunos a quem dei aulas na Northwestern, Duke, UC Davis e outras escolas. Muitas das suas perguntas e comentários foram incluídos neste livro.

Os meus agradecimentos finais vão para o Dr. Joseph Bogen, um dos primeiros pioneiros da IA/ML e da Neurociência. Se não nos tivéssemos cruzado no Caltech, não havia qualquer hipótese de eu ser professor ou de este livro existir. O seu impacto foi assim tão grande na minha vida.

De Alfredo

Estou absolutamente grato pelo apoio da minha família enquanto escrevia este livro: Claudia, Efrain, Ignacio e Alana - o teu apoio e paciência foram essenciais para chegar à meta. Mais uma vez, obrigado por todas as oportunidades de trabalhar contigo, Noah; esta foi outra viagem incrível. Valorizo a tua amizade e a nossa relação profissional.

Agradece à Melissa Potter (sem dúvida a melhor editora com quem já trabalhei) pelo seu trabalho fantástico. Os nossos editores técnicos foram óptimos a encontrar problemas e a destacar sítios que precisavam de ser melhorados, o que é sempre uma coisa difícil de fazer bem.

Também estou extremamente grato pela ajuda de Lee Stott com o Azure. O conteúdo do Azure não seria tão bom sem ele. E obrigado à Francesca Lazzeri, ao Mike McCoy e a todos os outros que contactei na Microsoft sobre o livro. Foram todos muito prestáveis.

Capítulo 1. Introdução aos MLOps

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

Desde 1986, , tive mais algumas mortes, algumas por falta de atenção, mas principalmente por ter deliberadamente ultrapassado os limites em várias direcções - arriscar no bonsai é um pouco como arriscar no amor; o melhor resultado requer uma exposição arriscada a ser ferido e nenhuma garantia de sucesso.

—Dr. Joseph Bogen

Um dos aspectos mais poderosos da ficção científica é a sua capacidade de imaginar um futuro sem restrições. Um dos programas de ficção científica mais influentes de todos os tempos é o programa de TV *Star Trek*, que foi ao ar pela primeira vez em meados da década de 1960, há aproximadamente 60 anos. O impacto cultural inspirou os criadores de tecnologias como o Palm Pilot e os telemóveis portáteis. Além disso, *Star Trek* influenciou o cofundador do computador Apple, Steve Wozniak, a criar os computadores Apple.

Nesta era de inovação na aprendizagem automática, há muitas ideias essenciais da série original relevantes para a revolução industrial MLOps (ou Operações de Aprendizagem Automática) que se aproxima. Por exemplo, os tricorders portáteis do *Star Trek* podem classificar instantaneamente objectos utilizando modelos de classificação multiclasse pré-treinados. Mas, em última análise, neste mundo futurista de ficção científica, os especialistas no domínio, como os oficiais de ciências, os oficiais médicos ou o capitão da nave, não passam meses a treinar modelos

de aprendizagem automática. Da mesma forma, a tripulação da sua nave científica, a *Enterprise*, não é chamada de cientistas de dados. Em vez disso, têm empregos em que utilizam frequentemente a ciência dos dados.

Muitos dos aspectos da aprendizagem automática deste futuro de ficção científica do *Star Trek* já não são ficção científica na década de 2020. Este capítulo guia o leitor na teoria fundamental de como tornar isto possível. Vamos começar.

Ascensão do engenheiro de aprendizagem automática e dos MLOps

A aprendizagem automática (ML), com a sua adoção generalizada a nível mundial, criou a necessidade de uma abordagem sistemática e eficiente para a construção de sistemas ML, o que levou a um aumento súbito da procura de engenheiros ML. Estes engenheiros de ML, por sua vez, estão a aplicar as melhores práticas DevOps estabelecidas às tecnologias emergentes de aprendizagem automática. Todos os principais fornecedores de Cloud têm certificações destinadas a estes profissionais. Tenho experiência de trabalho direto com a AWS, Azure e GCP como especialista em aprendizagem automática. Em alguns casos, isto inclui ajudar a criar as próprias certificações de aprendizagem automática e material de formação oficial. Além disso, ensino engenharia de aprendizagem automática e computação Cloud em alguns dos melhores programas de ciência de dados, como Duke e Northwestern. Vi em primeira mão o crescimento da engenharia de aprendizagem automática, uma vez que muitos dos meus antigos alunos se tornaram engenheiros de aprendizagem automática.

A Google tem uma [certificação de Engenheiro Profissional de Aprendizagem Automática](#). Descreve um engenheiro de ML como alguém que "concebe, constrói e produz modelos de ML para resolver desafios empresariais..." O Azure tem uma [certificação Microsoft Certified: Associado Cientista de Dados do Azure](#). Descreve este tipo de profissional como alguém que "aplica os seus conhecimentos de ciência de dados e aprendizagem automática para implementar e executar cargas de trabalho

de aprendizagem automática..." Por último, a AWS descreve um **especialista em aprendizagem automática certificado pela AWS** como alguém com "a capacidade de conceber, implementar, implantar e manter soluções de aprendizagem automática para determinados problemas empresariais".

Uma forma de olhar para a ciência dos dados versus a engenharia de aprendizagem automática é considerar a ciência versus a própria engenharia. A ciência orienta-se para a pesquisa e a engenharia orienta-se para a produção. À medida que a aprendizagem automática ultrapassa a vertente da pesquisa, as empresas estão ansiosas por um retorno do investimento em contratações relacionadas com IA e ML. De acordo com o payscale.com e o glassdoor.com, os resultados mostram que, no final de 2020, o salário médio de um cientista de dados, engenheiro de dados e engenheiro de aprendizagem automática era semelhante. De acordo com o LinkedIn, no quarto trimestre de 2020, 191 mil empregos mencionavam Cloud, havia 70 mil anúncios de emprego de engenharia de dados, 55 mil anúncios de emprego de engenharia de aprendizagem automática e 20 mil anúncios de emprego de ciência de dados, como mostra [a Figura 1-1](#).

Outra forma de ver estas tendências de emprego é que são uma parte natural do ciclo de hype da tecnologia. As organizações apercebem-se de que, para gerar um retorno do investimento (ROI), precisam de empregados com competências técnicas: computação Cloud, engenharia de dados e aprendizagem automática. Também precisam deles em muito maior quantidade do que os cientistas de dados. Como resultado, a década de 2020 pode mostrar uma aceleração no tratamento da ciência de dados como um comportamento, em vez de um título de emprego. DevOps é um comportamento, assim como a ciência de dados. Pensa nos princípios do DevOps e da ciência de dados. Em ambos os casos, o DevOps e a ciência de dados são metodologias para avaliar o mundo, não necessariamente títulos de emprego únicos.

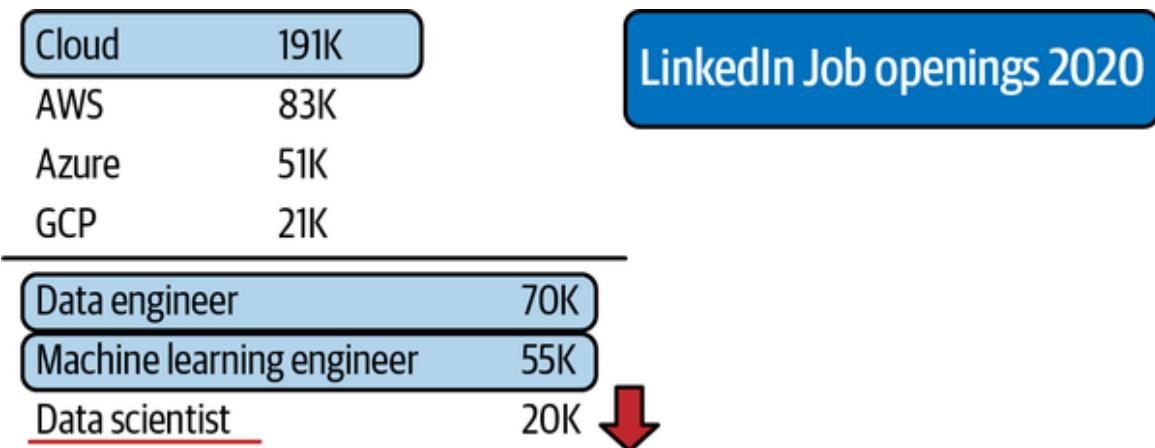


Figura 1-1. Trabalhos de aprendizagem automática

Vejamos as opções para medir o sucesso de uma iniciativa de engenharia de aprendizagem automática numa organização. Primeiro, podes contar os modelos de aprendizagem automática que entram em produção. Em segundo lugar, podes medir o impacto dos modelos de aprendizagem automática no ROI da empresa. Estas métricas culminam na eficiência operacional de um modelo. O custo, o tempo de funcionamento e o pessoal necessário para o manter são sinais que prevêem o sucesso ou o fracasso de um projeto de engenharia de aprendizagem automática.

As organizações de tecnologia avançada sabem que precisam de utilizar metodologias e ferramentas que diminuam o risco de fracasso dos projectos de aprendizagem automática. Então, quais são essas ferramentas e processos utilizados na engenharia de aprendizagem automática? Apresentamos-te uma lista parcial:

Plataformas de ML nativas da Cloud

AWS SageMaker, Azure ML Studio e GCP AI Platform

Fluxos de trabalho em contentores

Contentores em formato Docker, Kubernetes e registos de contentores públicos e privados

Tecnologia sem servidor

AWS Lambda, AWS Athena, Google Cloud Functions, Azure Functions

Hardware especializado para aprendizagem automática

GPUs, Google TPU (TensorFlow Processing Unit), Apple A14, AWS Inferentia Elastic inference

Plataformas e ferramentas de grandes volumes de dados

Databricks, Hadoop/Spark, Snowflake, Amazon EMR (Elastic Map Reduce), Google Big Query

Um padrão claro sobre a aprendizagem automática é o facto de estar profundamente ligada à computação Cloud. Isto deve-se ao facto de os ingredientes brutos da aprendizagem automática exigirem uma computação maciça, dados extensos e hardware especializado. Assim, existe uma sinergia natural com uma integração profunda com as plataformas Cloud e a engenharia da aprendizagem automática. Para além disso, as plataformas Cloud estão a criar plataformas especializadas para melhorar a operacionalização do ML. Portanto, se estás a fazer engenharia de ML, provavelmente estás a fazê-lo na Cloud. Em seguida, vamos discutir como o DevOps desempenha um papel nesse processo.

O que são MLOps?

Porque é que a aprendizagem automática não é 10 vezes mais rápida? A maior parte dos sistemas de aprendizagem automática que criam problemas envolve tudo o que rodeia a modelação da aprendizagem automática: engenharia de dados, processamento de dados, viabilidade do problema e alinhamento empresarial. Um dos problemas é a concentração no "código" e nos pormenores técnicos em vez de resolver o problema comercial com a aprendizagem automática. Há também uma falta de automatização e a questão da cultura HiPPO (Highest Paid Person's Opinions). Por último, grande parte da aprendizagem automática não é nativa da Cloud e utiliza conjuntos de dados académicos e pacotes de software académicos que não são dimensionados para problemas de grande escala.

Quanto mais rápido for o ciclo de feedback (ver Kaizen), mais tempo tens para te concentrar nos problemas empresariais, como as questões recentes da deteção rápida da Covid, da deteção de soluções de visão por computador com máscara e sem máscara implementadas no mundo real e da descoberta mais rápida de medicamentos. A tecnologia existe para resolver estes problemas, mas estas soluções não estão disponíveis no mundo real? Porque é que isso acontece?

NOTA

O que é Kaizen? Em japonês, significa melhoria. A utilização do Kaizen como filosofia de gestão de software teve origem na indústria automóvel japonesa após a Segunda Guerra Mundial. Está na base de muitas outras técnicas: Kanban, análise de causa raiz e cinco porquês, e Six Sigma. Para praticar o Kaizen, é necessária uma avaliação exacta e realista do estado do mundo e procura melhorias diárias e incrementais na busca da excelência.

A razão pela qual os modelos não estão a entrar em produção é o impulso para o surgimento do MLOps como um padrão crítico da indústria. O MLOps partilha uma linhagem com o DevOps, na medida em que o DevOps exige filosoficamente a automatização. Uma expressão comum é: *se não for automatizado, está quebrado*. Da mesma forma, com MLOps, não deve haver componentes do sistema que tenham humanos como alavancas da máquina. A história da automação mostra que os seres humanos são os menos valiosos a fazer tarefas repetitivas, mas são os mais valiosos a utilizar a tecnologia como arquitectos e profissionais. Da mesma forma, a coordenação entre programadores, modelos e operações deve ser feita através de um trabalho de equipa transparente e de uma colaboração saudável. Pensa no MLOps como o processo de automatização da aprendizagem automática utilizando metodologias DevOps.

NOTA

O que é o DevOps? Combina as melhores práticas, incluindo microserviços, integração contínua e entrega contínua, removendo as barreiras entre Operações e Desenvolvimento e Trabalho em Equipa. Podes ler mais sobre DevOps no nosso livro *Python for DevOps* (O'Reilly). Python é a linguagem predominante de scripting, DevOps e aprendizagem automática. Como resultado disso, este livro MLOps foca em Python, assim como o livro DevOps focou em Python.

Com MLOps, não só os processos de engenharia de software precisam de automação total, mas também os dados e a modelação. O treinamento e a implantação do modelo são uma nova ruga adicionada ao ciclo de vida tradicional do DevOps. Por fim, a monitorização e a instrumentação adicionais têm de ter em conta as novas coisas que podem avariar, como o desvio de dados - o delta entre as alterações nos dados desde a última vez que ocorreu o treino do modelo.

Um problema fundamental para colocar modelos de aprendizagem automática em produção é a imaturidade do sector da ciência dos dados. A indústria de software adoptou o DevOps para resolver problemas semelhantes; agora, a comunidade de aprendizagem automática adopta o MLOps. Vamos ver como fazer isso.

DevOps e MLOps

DevOps é um conjunto de práticas técnicas e de gestão que visam aumentar a velocidade de uma organização no lançamento de software de alta qualidade. Alguns dos benefícios do DevOps incluem velocidade, fiabilidade, escala e segurança. Estes benefícios ocorrem através da adesão às seguintes práticas recomendadas:

Integração contínua (CI)

CI é o processo de testar continuamente um projeto de software e melhorar a qualidade com base nos resultados desses testes. É um teste automatizado que utiliza servidores de construção de código aberto e

SaaS, como o GitHub Actions, Jenkins, Gitlab, CircleCI, ou sistemas de construção nativos da Cloud, como o AWS Code Build.

Entrega contínua (CD)

Este método fornece código para um novo ambiente sem intervenção humana. A CD é o processo de implementação automática de código, muitas vezes através da utilização de IaC.

Microsserviços

Um microsserviço é um serviço de software com uma função distinta que tem poucas ou nenhuma dependências. Uma das estruturas de microsserviço mais populares baseadas em Python é o Flask. Por exemplo, um ponto final de previsão de aprendizagem automática é uma excelente opção para um microsserviço. Estes microsserviços podem utilizar uma grande variedade de tecnologias, incluindo FaaS (função como um serviço). Um exemplo perfeito de uma função Cloud é o AWS Lambda. Um microsserviço pode estar preparado para contentores e utilizar CaaS (contentor como serviço) para implementar uma aplicação Flask com um Dockerfile num serviço como o AWS Fargate, Google Cloud Run ou Azure App Services.

Infraestrutura como código

A Infraestrutura como Código (IaC) é o processo de verificar a infraestrutura num repositório de código-fonte e "implantá-la" para enviar alterações para esse repositório. A IaC permite um comportamento idempotente e garante que a infraestrutura não necessita de humanos para a construir. Um ambiente Cloud definido puramente em código e verificado num repositório de controlo de origem é um bom exemplo de caso de utilização. As tecnologias populares incluem IaC específicas para a nuvem, como AWS Cloud Formation ou **AWS SAM (Serverless Application Model)**. As opções multicloud incluem **Pulumi** e **Terraform**.

Monitorização e instrumentação

A monitorização e a instrumentação são os processos e técnicas utilizados que permitem a uma organização tomar decisões sobre o desempenho e a fiabilidade de um sistema de software. Através do registo e de outras ferramentas, como as ferramentas de monitorização do desempenho das aplicações, como a New Relic, a Data Dog ou a Stackdriver, a monitorização e a instrumentação recolhem essencialmente dados sobre o comportamento de uma aplicação em produção ou da ciência dos dados para sistemas de software implementados. É neste processo que o Kaizen entra em ação; a organização orientada para os dados utiliza esta instrumentação para melhorar as coisas diariamente ou semanalmente.

Comunicação técnica eficaz

Esta competência envolve a capacidade de criar métodos de comunicação eficazes, repetíveis e eficientes. Um excelente exemplo de comunicação técnica eficaz pode ser a adoção da AutoML para a prototipagem inicial de um sistema. É claro que, em última instância, o modelo AutoML pode ser mantido ou descartado. No entanto, a automação pode servir como uma ferramenta informativa para evitar o trabalho num problema intratável.

Gestão eficaz de projectos técnicos

Este processo pode utilizar eficazmente soluções humanas e tecnológicas, como sistemas de bilhetes e folhas de cálculo, para gerir projectos. Além disso, a gestão adequada de projectos técnicos requer a divisão dos problemas em pequenos e discretos pedaços de trabalho, para que se verifique um progresso gradual. Um antípadrão na aprendizagem automática é muitas vezes quando uma equipa trabalha num modelo de máquina de produção que resolve um problema "perfeitamente". Em vez disso, pequenos ganhos entregues diariamente ou semanalmente são uma abordagem mais escalável e prudente para a construção de modelos.

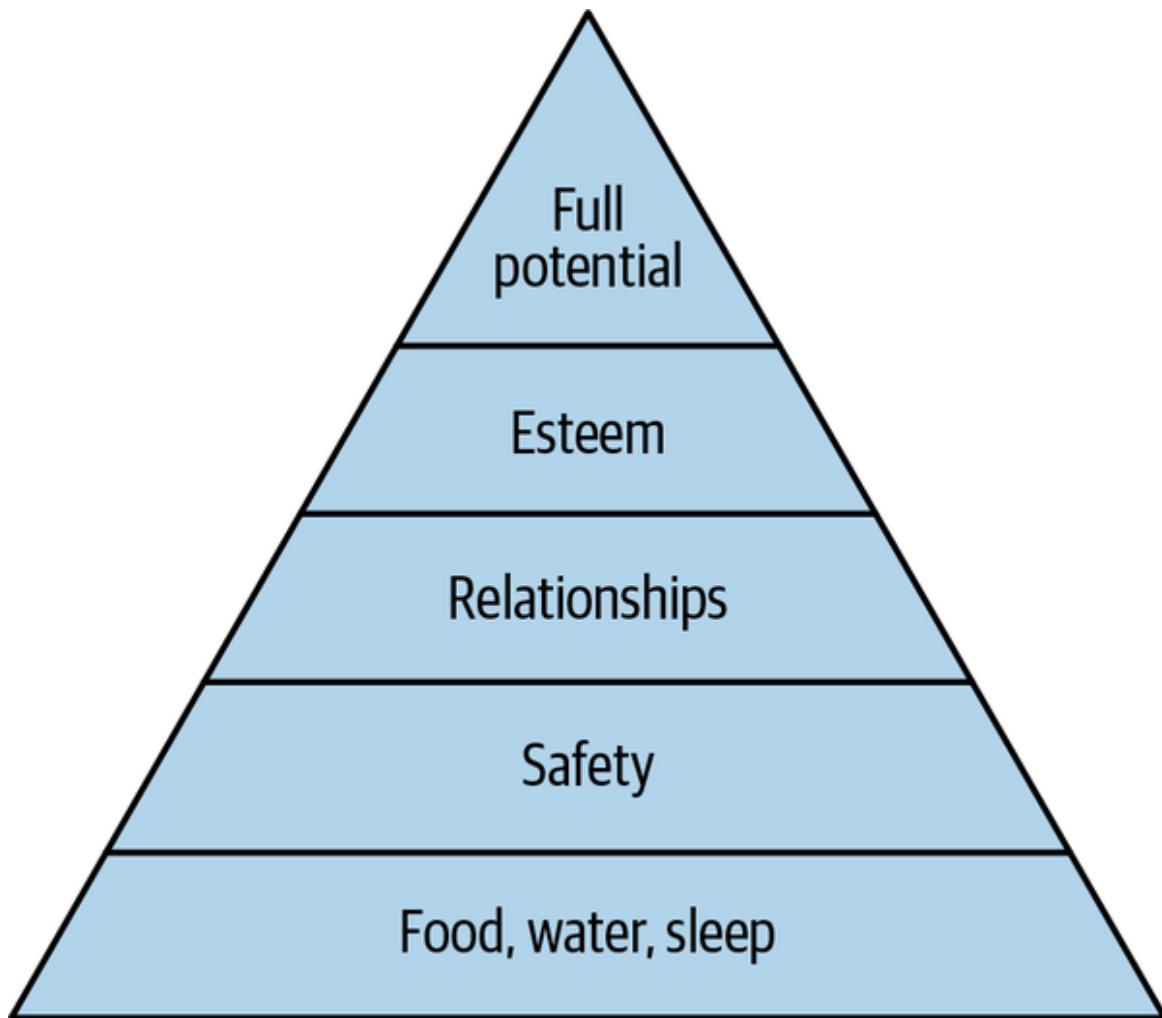
A integração contínua e a entrega contínua são dois dos pilares mais críticos do DevOps. A integração contínua envolve a fusão de código num repositório de controlo de fontes que verifica automaticamente a qualidade do código através de testes. A entrega contínua é quando as alterações ao código são automaticamente testadas e implementadas, quer num ambiente de preparação quer na produção. Ambas as técnicas são uma forma de automatização no espírito do Kaizen ou da melhoria contínua.

Uma boa pergunta a fazer é: quem na equipa deve implementar a CI/CD? Esta pergunta seria semelhante a perguntar quem contribui para os impostos numa democracia. Em uma democracia, os impostos pagam por estradas, pontes, aplicação da lei, serviços de emergência, escolas e outras infra-estruturas, então todos devem contribuir para construir uma sociedade melhor. Da mesma forma, todos os membros da equipa MLOps devem ajudar a desenvolver e manter o sistema CI/CD. Um sistema de CI/CD bem mantido é uma forma de investimento no futuro da equipa e da empresa.

Um sistema de aprendizagem automática é também um sistema de software, mas contém um componente único: um modelo de aprendizagem automática. Os mesmos benefícios do DevOps podem e aplicam-se aos sistemas de ML. A adoção da automação é a razão pela qual novas abordagens, como o Data Versioning e o AutoML, são muito promissoras na captação da mentalidade DevOps.

Uma hierarquia de necessidades MLOps

Uma forma de pensar nos sistemas de aprendizagem automática é considerar a hierarquia de necessidades de Maslow, como se mostra na [Figura 1-2](#). Os níveis mais baixos de uma pirâmide reflectem a "sobrevivência" e o verdadeiro potencial humano ocorre depois de satisfeitas as necessidades básicas de sobrevivência e emocionais.



Maslow's hierarchy of needs

Figura 1-2. Teoria da hierarquia das necessidades de Maslow

O mesmo conceito aplica-se à aprendizagem automática. Um sistema de aprendizagem automática é um sistema de software, e os sistemas de software funcionam de forma eficiente e fiável quando o DevOps e as melhores práticas de engenharia de dados estão em vigor. Por isso, como é que é possível oferecer o verdadeiro potencial da aprendizagem automática a uma organização se as regras básicas fundamentais do DevOps não existirem ou se a engenharia de dados não for totalmente automatizada? A hierarquia de necessidades de aprendizagem automática apresentada na **Figura 1-3** não é um guia definitivo, mas é um excelente ponto de partida para a discussão.

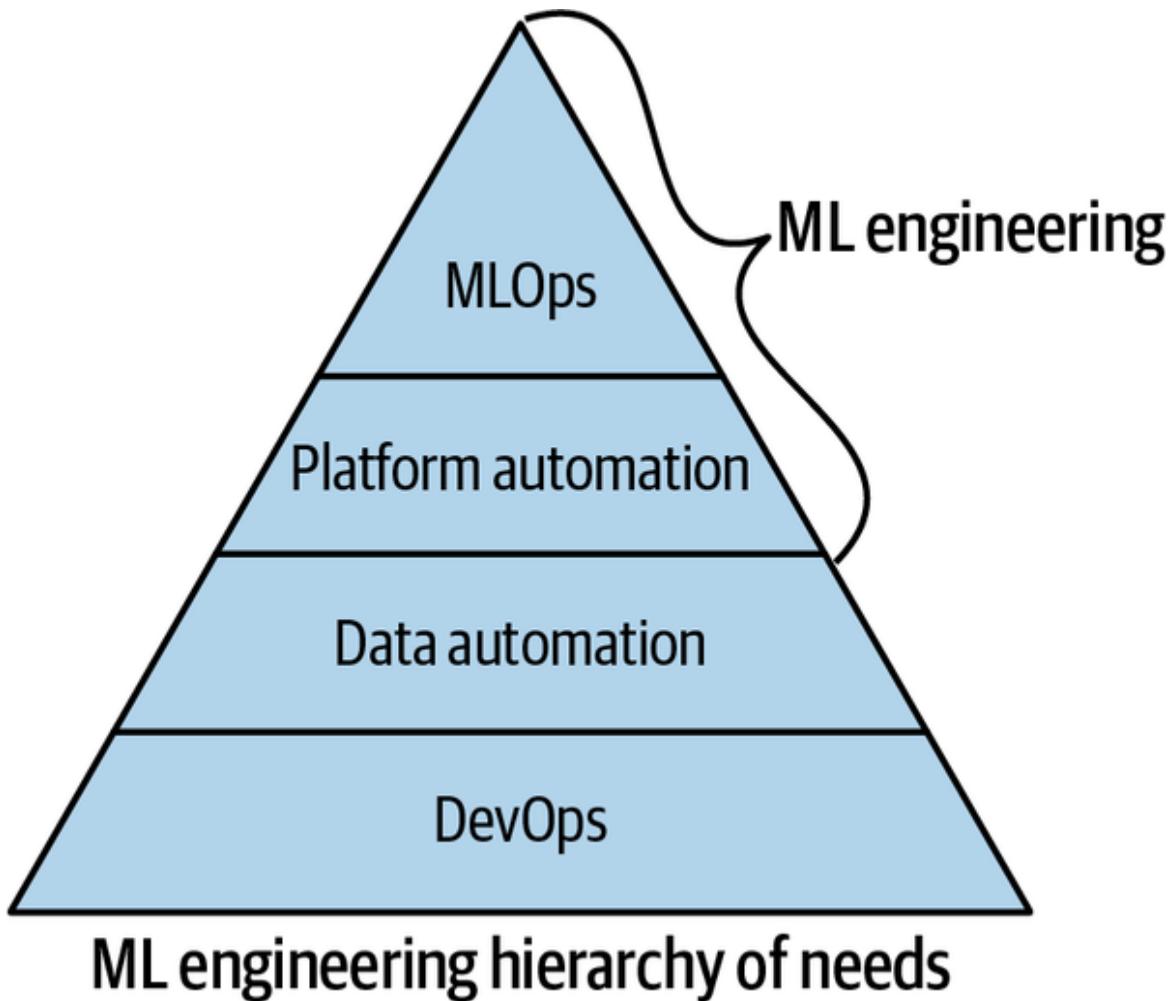


Figura 1-3. Hierarquia de necessidades da engenharia de AM

Um dos principais obstáculos aos projectos de aprendizagem automática é esta base necessária de DevOps. Depois de concluída esta base, segue-se a automatização dos dados, depois a automatização da plataforma e, finalmente, a verdadeira automatização do ML, ou MLOps. O culminar do MLOps é um sistema de aprendizagem automática que funciona. As pessoas que trabalham na operacionalização e na criação de aplicações de aprendizagem automática são engenheiros de aprendizagem automática e/ou engenheiros de dados. Vamos analisar cada passo da hierarquia do ML e certificar-nos de que tens um bom conhecimento da sua implementação, começando pelo DevOps.

Implementar o DevOps

A base do DevOps é a integração contínua. Sem testes automatizados, não há como avançar com DevOps. A integração contínua é relativamente simples para um projeto Python com as ferramentas modernas disponíveis. O primeiro passo é construir um "scaffolding" para um projeto Python, como mostrado na [Figura 1-4](#).

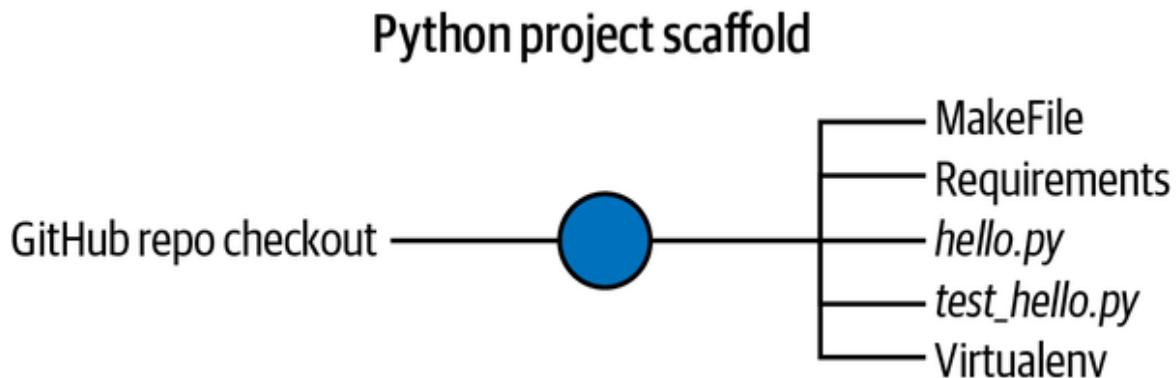


Figura 1-4. Andaime de projeto Python

É quase certo que o tempo de execução de um projeto de aprendizagem automática Python seja num sistema operativo Linux. Como resultado, a seguinte estrutura de projeto Python é simples de implementar para projectos de ML. Podes aceder ao código-fonte deste exemplo [no GitHub](#) para referência enquanto lês esta secção. Os componentes são os seguintes:

Makefile

Um *Makefile* executa "receitas" através do sistema `make`, que vem com sistemas operacionais baseados em Unix. Portanto, um *Makefile* é uma escolha ideal para simplificar as etapas envolvidas na integração contínua, como as seguintes. Observa que um *Makefile* é um bom ponto de partida para um projeto e irá frequentemente evoluir à medida que novas partes necessitam de automatização.

NOTA

Se o teu projeto usa um ambiente virtual Python, tu fazes o source dele antes de trabalhares com um Makefile, uma vez que tudo o que um Makefile faz é correr comandos. É um erro comum para um recém-chegado ao Python confundir o Makefile com um ambiente virtual. Similarmente, supõe que usas um editor como o Microsoft Visual Studio Code. Nesse caso, terás de informar o editor sobre o teu ambiente virtual Python para que ele te possa dar com precisão o realce de sintaxe, linting, e outras bibliotecas disponíveis.

Faz a instalação

Este passo instala o software através do comando `make install`

Faz cotão

Este passo verifica se existem erros de sintaxe através do comando `make lint`

Faz o teste

Esta etapa executa testes através do comando `make test`:

```
install:  
    pip install --upgrade pip &&\n        pip install -r requirements.txt  
lint:  
    pylint --disable=R,C hello.py  
  
test:  
    python -m pytest -vv --cov=hello test_hello.py
```

PORQUÊ UM MAKEFILE?

Uma reação comum ao ouvires sobre um Makefile de um principiante absoluto em Python é "Porque é que eu preciso disto?" Geralmente, é saudável ter ceticismo sobre coisas que parecem adicionar trabalho. No caso de um Makefile, no entanto, eles são na verdade menos trabalhosos porque eles controlam passos de construção complicados que são muito difíceis de lembrar e digitar corretamente.

Um ótimo exemplo é um passo de lint com a ferramenta `pylint`. Com um Makefile, só precisas de executar `make lint`, e o mesmo comando pode ser executado dentro de um servidor de integração contínua. A abordagem alternativa é digitar a diretiva completa sempre que precisar dela, como a seguinte:

```
pylint --disable=R,C *.py
```

Esta sequência é muito propensa a erros e é bastante entediante digitá-la repetidamente ao longo da vida do teu projeto. Em vez disso, é muito mais simples escreveres o seguinte:

```
make lint
```

Quando adoptas a abordagem Makefile, simplifica o teu fluxo de trabalho e facilita a integração do teu projeto num sistema de integração contínua. Há menos código para digitar, e isso é sempre uma coisa boa para a automação. Além disso, os comandos do Makefile são reconhecidos pelo auto-completar do shell, facilitando o "tab-completar" dos passos.

requisitos.txt

Um ficheiro *requirements.txt* é uma convenção usada pela ferramenta de instalação `pip`, a ferramenta de instalação predefinida para Python. Um

projeto pode conter um ou mais destes ficheiros se diferentes pacotes necessitarem de instalação para diferentes ambientes.

Código fonte e testes

A parte final do scaffolding Python é adicionar um ficheiro de código fonte e um ficheiro de teste, como mostrado aqui. Este script existe num ficheiro chamado *hello.py*:

```
def add(x, y):
    """This is an add function"""

    return x + y

print(add(1, 1))
```

A seguir, o ficheiro de teste é muito trivial de criar utilizando a estrutura `pytest`. Este script estaria em um arquivo *test_hello.py* contido na mesma pasta que *hello.py* para que o `from hello import add` funcione:

```
from hello import add

def test_add():
    assert 2 == add(1, 1)
```

Estes quatro ficheiros: *Makefile*, *requirements.txt*, *hello.py*, e *test_hello.py* são tudo o que é necessário para começar a jornada de integração contínua, exceto para criar um ambiente virtual Python local. Para fazer isso, primeiro, crie-o:

```
python3 -m venv ~/.your-repo-name
```

Além disso, tem em atenção que existem geralmente duas formas de criar um ambiente virtual. Primeiro, muitas distribuições Linux incluem

a ferramenta de linha de comando `virtualenv`, que faz a mesma coisa que `python3 -m venv`.

Em seguida, utiliza a fonte para a "ativar":

```
source ~/your-repo-name/bin/activate
```

NOTA

Porquê criar e usar um ambiente virtual Python? Esta pergunta é omnipresente para os recém-chegados ao Python, e há uma resposta direta. Porque o Python é uma linguagem interpretada, pode "agarrar" bibliotecas de qualquer parte do sistema operativo. Um ambiente virtual Python isola os pacotes de terceiros para um diretório específico. Existem outras soluções para este problema e muitas ferramentas de desenvolvimento. Elas resolvem efetivamente o mesmo problema: a biblioteca e o interpretador Python estão isolados num projeto particular.

Assim que tiveres este andaime configurado, podes realizar os seguintes passos de integração contínua local:

1. Utiliza `make install` para instalar as bibliotecas do teu projeto.

A saída será semelhante à [Figura 1-5](#) (este exemplo mostra uma execução em [GitHub Codespaces](#)):

```
$ make install
pip install --upgrade pip && \
    pip install -r requirements.txt
Collecting pip
    Using cached pip-20.2.4-py2.py3-none-any.whl (1.5 MB)
[.....more output suppressed here.....]
```

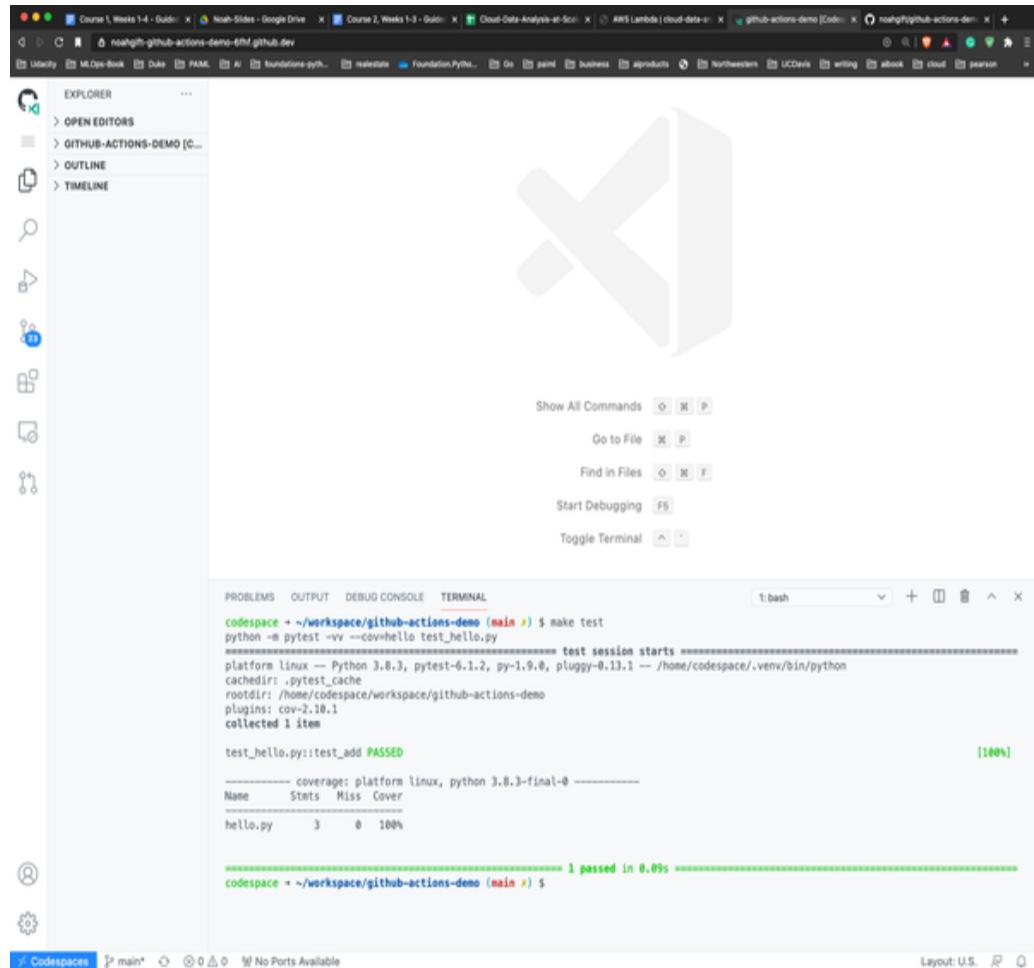


Figura 1-5. Espaços de código do GitHub

2. Executa `make lint` para colocar lint no teu projeto:

```
$ make lint  
pylint --disable=R,C hello.py
```

```
-----  
Your code has been rated at 10.00/10
```

3. Executa `make test` para testar o teu projeto:

```
$ make test  
python -m pytest -vv --cov=hello test_hello.py  
===== test session starts =====
```

```
platform linux -- Python 3.8.3, pytest-6.1.2, \
/home/codespace/.venv/bin/python
cachedir: .pytest_cache
rootdir: /home/codespace/workspace/github-actions-demo
plugins: cov-2.10.1
collected 1 item

test_hello.py::test_add PASSED
[100%]

----- coverage: platform linux, python 3.8.3-final-0 -----
Name     Stmts Miss Cover
-----
hello.py    3     0   100%
```

Uma vez que isto esteja a funcionar localmente, é simples integrar este mesmo processo com um servidor de construção SaaS remoto. As opções incluem o GitHub Actions, um servidor de compilação nativo da Cloud, como o AWS Code Build, o GCP CloudBuild, o Azure DevOps Pipelines ou um servidor de compilação auto-hospedado de código aberto, como o Jenkins.

Configurar a integração contínua com GitHub Actions

Uma das formas mais simples de implementar a integração contínua para este projeto de scaffolding Python é com as GitHub Actions. Para isso, podes selecionar "Actions" (Ações) na UI do GitHub e criar uma nova ou criar um ficheiro dentro destas diretorias que criaste, como se mostra aqui:

```
.github/workflows/<yourfilename>.yml
```

O ficheiro GitHub Actions em si é simples de criar, e o seguinte é um exemplo de um. Nota que a versão exacta do Python é definida para qualquer interpretação que o projeto exija. Neste exemplo, eu quero

verificar uma versão específica do Python que roda no Azure. As etapas de integração contínua são triviais de implementar devido ao trabalho árduo anterior na criação de um Makefile:

```
name: Azure Python 3.5
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python 3.5.10
        uses: actions/setup-python@v1
      - with:
          python-version: 3.5.10
      - name: Install dependencies
        run: |
          make install
      - name: Lint
        run: |
          make lint
      - name: Test
        run: |
          make test
```

Uma visão geral do aspecto de GitHub Actions quando executado em eventos "push" de um repositório GitHub é mostrada na [Figura 1-6](#).

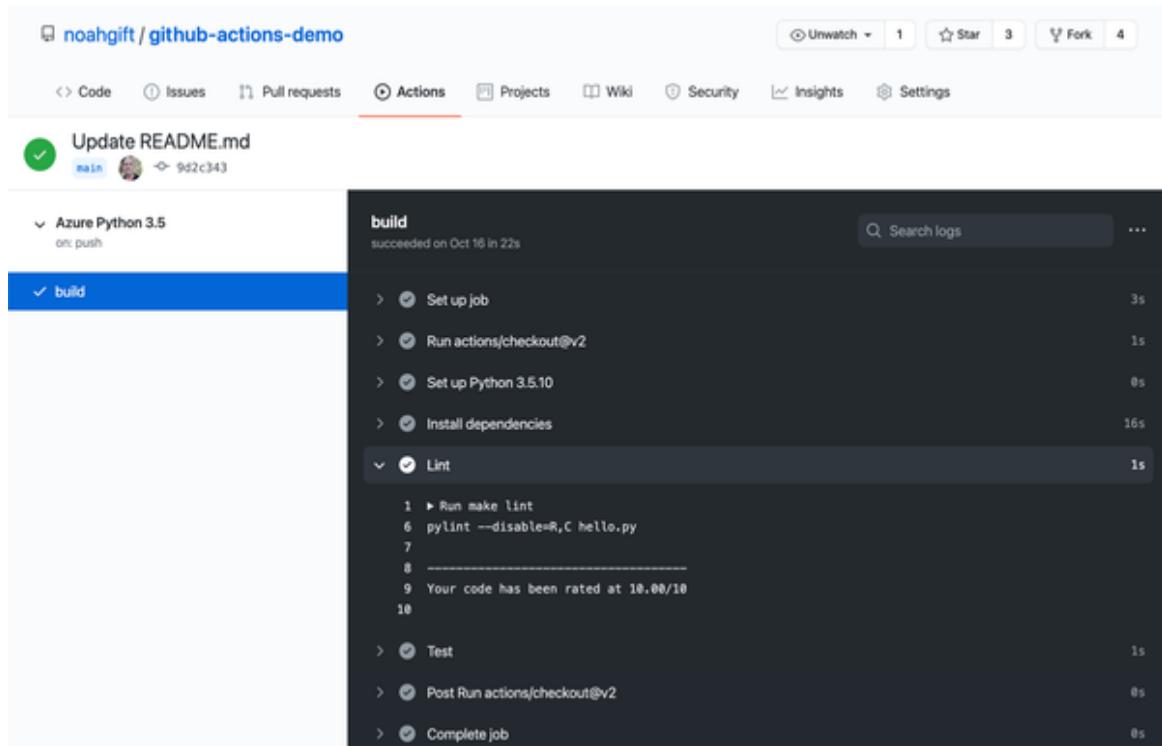


Figura 1-6. Ações do GitHub

Esta etapa completa a parte final da configuração da integração contínua. Uma implantação contínua - ou seja, colocar automaticamente o projeto de aprendizado de máquina em produção - seria a próxima etapa lógica. Essa etapa envolveria a implantação do código em um local específico usando um processo de entrega contínua e IaC (Infraestrutura como código). Esse processo é mostrado na [Figura 1-7](#).

Continuous delivery

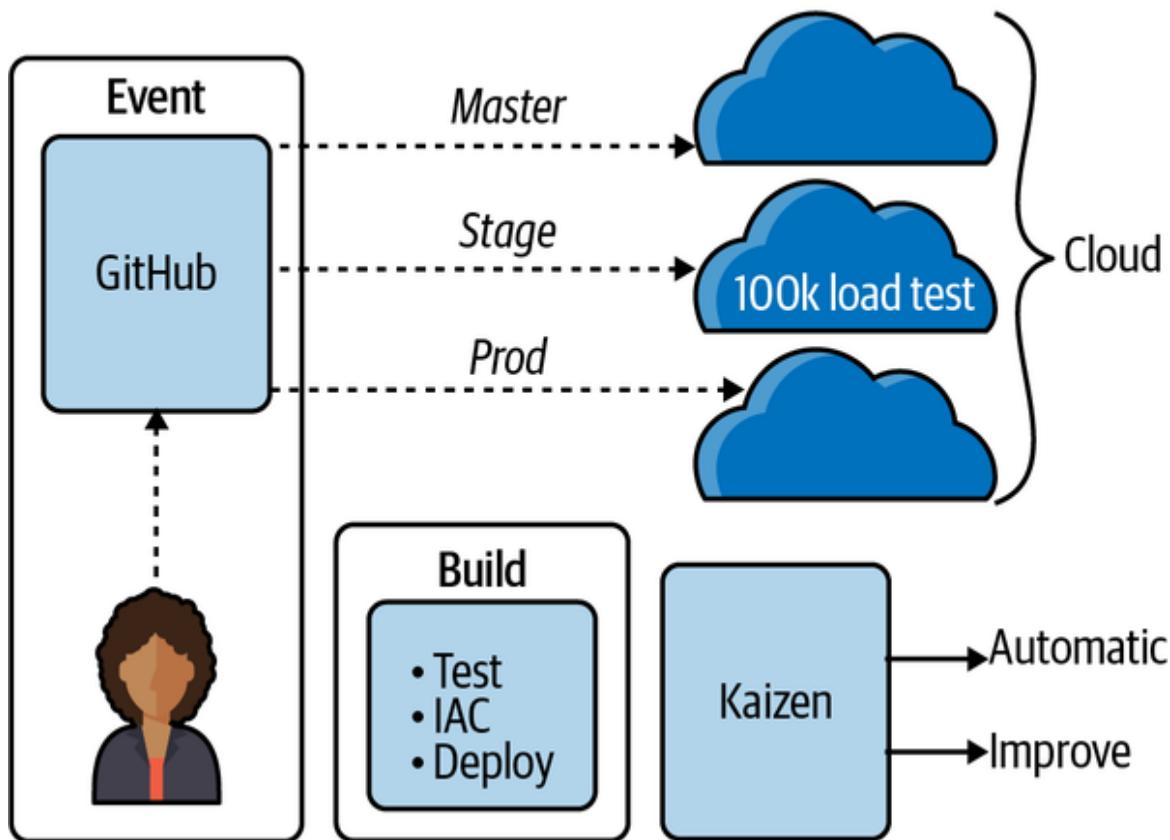


Figura 1-7. Entrega contínua

DataOps e engenharia de dados

A seguir, na hierarquia de necessidades do ML, há uma forma de automatizar o fluxo de dados. Por exemplo, imagina uma cidade com um poço como única fonte de água. A vida quotidiana é complicada devido à necessidade de organizar viagens para obter água, e as coisas que tomamos como garantidas podem não funcionar, como chuveiros quentes a pedido, lavagem de loiça a pedido ou irrigação automatizada. Da mesma forma, uma organização sem um fluxo automatizado de dados não pode fazer MLOps de forma fiável.

Muitas ferramentas comerciais estão a evoluir para fazer DataOps. Um exemplo inclui o [Apache Airflow](#), concebido pela Airbnb, e mais tarde de código aberto, para agendar e monitorizar os seus trabalhos de processamento de dados. As ferramentas da AWS incluem o AWS Data

Pipeline e o AWS Glue. AWS Glue é uma ferramenta ETL (Extrair, Carregar, Transformar) sem servidor que detecta o esquema de uma fonte de dados e, em seguida, armazena os metadados da fonte de dados. Outras ferramentas, como o AWS Athena e o AWS QuickSight, podem consultar e visualizar os dados.

Alguns itens a serem considerados aqui são o tamanho dos dados, a frequência com que as informações são alteradas e o grau de limpeza dos dados. Muitas organizações usam um lago de dados centralizado como o centro de todas as atividades em torno da engenharia de dados. A razão pela qual um lago de dados é útil para criar automação, incluindo a aprendizagem automática, é a escala "quase infinita" que proporciona em termos de E/S, juntamente com a sua elevada durabilidade e disponibilidade.

NOTA

Um lago de dados é frequentemente sinónimo de um sistema de armazenamento de objectos baseado na Cloud, como o Amazon S3. Um lago de dados permite o processamento de dados "no local", sem necessidade de os deslocar. Um lago de dados consegue-o através da capacidade quase infinita edas características de computação.

Quando trabalhei na indústria cinematográfica, em filmes como *Avatar*, os dados eram imensos e precisavam de ser movimentados por um sistema demasiado complicado. Agora, com a Cloud, este problema desaparece.

A Figura 1-8 mostra um fluxo de trabalho baseado num lago de dados Cloud. Observa a capacidade de realizar muitas tarefas, todas no local exato, sem mover os dados.

Os cargos dedicados, como o de engenheiro de dados, podem passar todo o seu tempo a criar sistemas que tratem destes diversos casos de utilização:

- Recolha periódica de dados e execução de trabalhos
- Processamento de dados de fluxo contínuo
- Dados sem servidor e orientados para eventos

- Empregos na área dos grandes dados
- Controlo de versões de dados e modelos para tarefas de engenharia ML

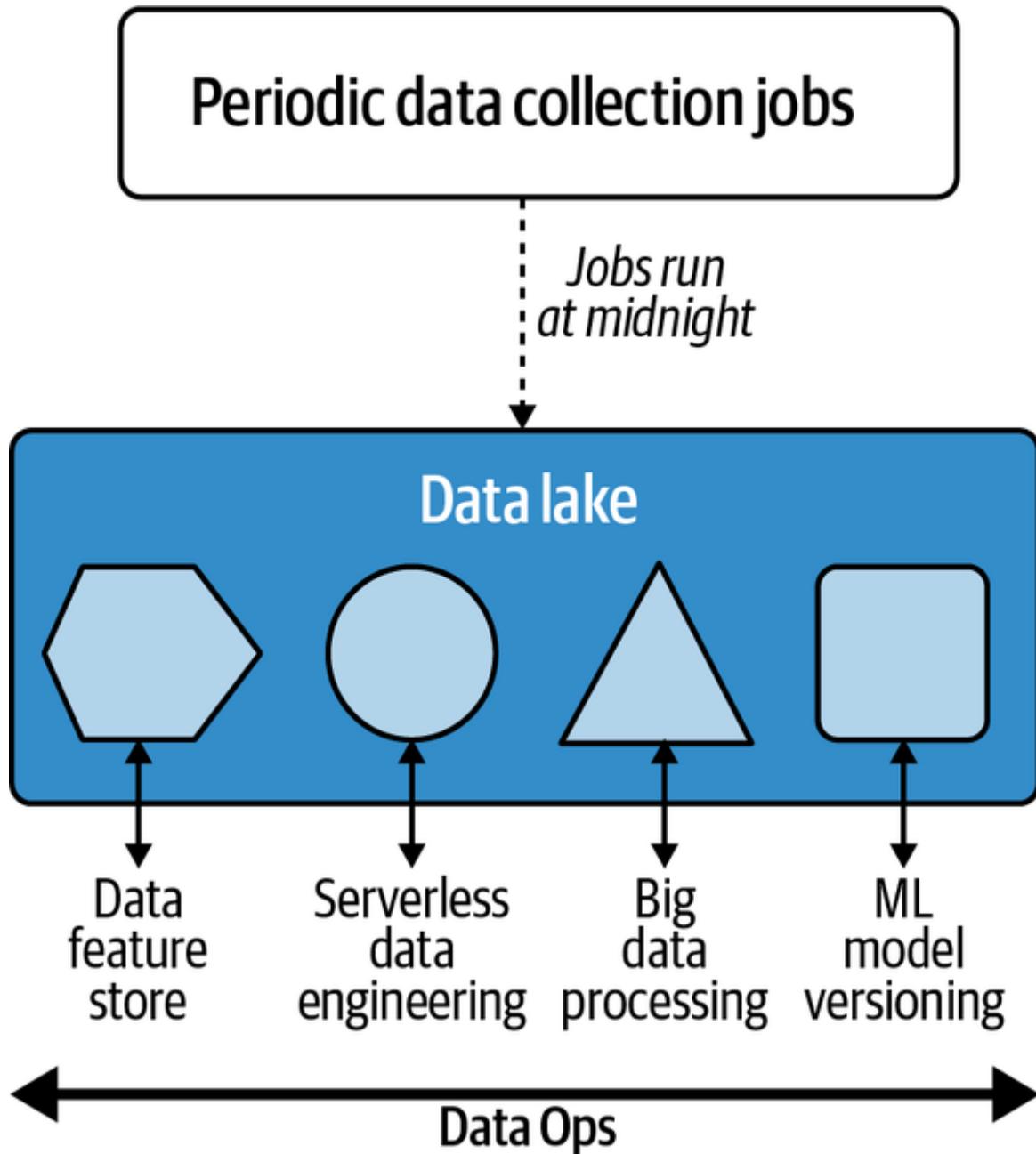


Figura 1-8. Engenharia de dados com um lago de dados Cloud

Tal como uma aldeia sem água corrente não pode utilizar uma máquina de lavar loiça automatizada, uma organização sem automatização de dados não pode utilizar métodos avançados de aprendizagem automática. Por

conseguinte, o processamento de dados necessita de automatização e operacionalização. Este passo permite operacionalizar e automatizar as tarefas de ML mais a jusante na cadeia.

Automatização da plataforma

Assim que houver um fluxo automatizado de dados, o próximo item da lista a ser avaliado é como uma organização pode usar plataformas de alto nível para criar soluções de aprendizado de máquina. Por exemplo, se uma organização já estiver a recolher dados para um lago de dados de uma plataforma Cloud, como o Amazon S3, é natural ligar os fluxos de trabalho de aprendizagem automática ao Amazon Sagemaker. Da mesma forma, se uma organização utiliza o Google, pode utilizar o Google AI Platform ou o Azure para utilizar o Azure Machine Learning Studio. Da mesma forma, o **Kubeflow** seria apropriado para uma organização que usa Kubernetes em vez de uma Cloud pública.

Um excelente exemplo de uma plataforma que resolve esses problemas aparece na [Figura 1-9](#). Observa que o AWS SageMaker orquestra uma sequência complexa de MLOps para um problema de aprendizado de máquina do mundo real, incluindo a ativação de máquinas virtuais, a leitura e a gravação no S3 e o provisionamento de pontos de extremidade de produção. Executar essas etapas de infraestrutura sem automação seria, na melhor das hipóteses, imprudente em um cenário de produção.

AWS SageMaker project architecture elasticity

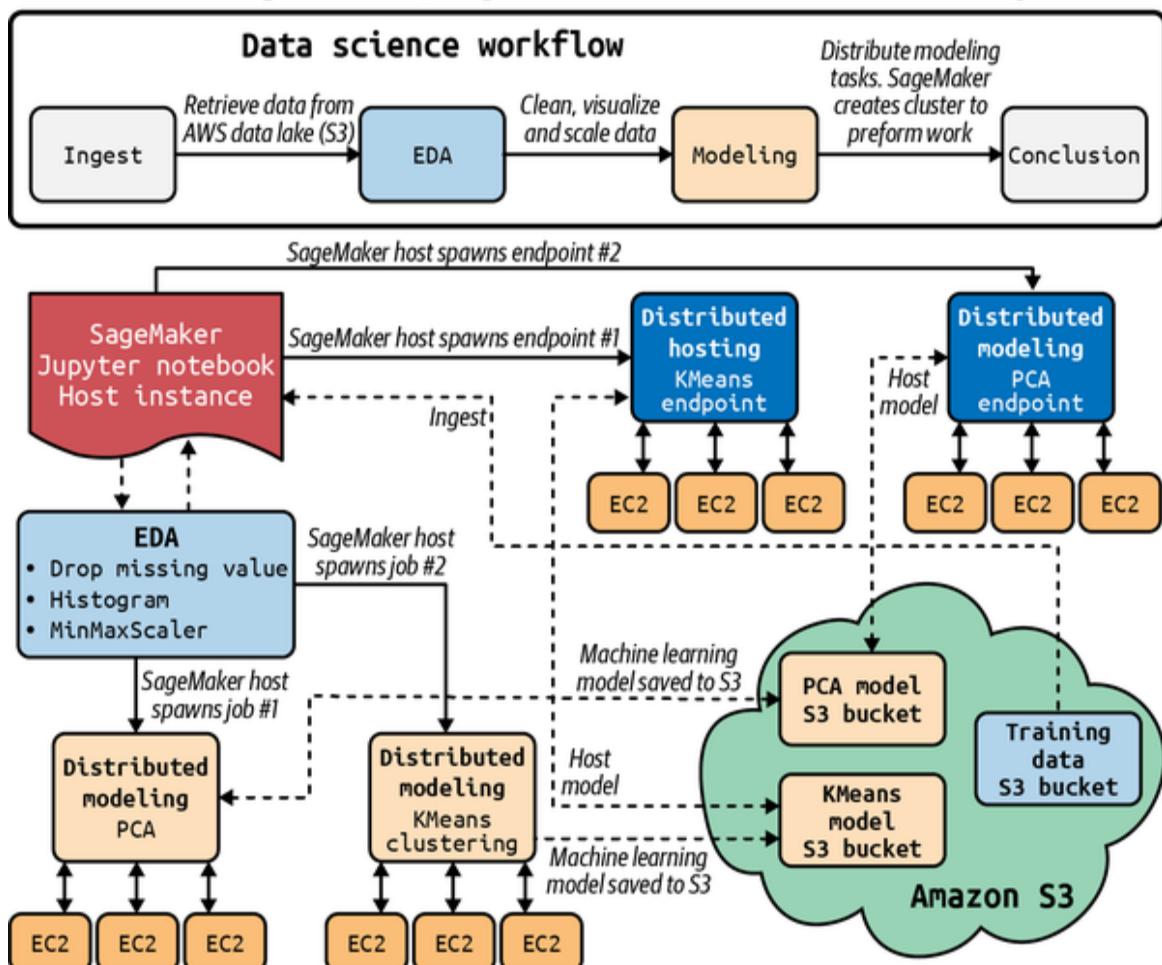


Figura 1-9. Pipeline do Sagemaker MLOps

Uma plataforma de ML resolve problemas de repetição, escala e operacionalização do mundo real.

MLOps

Assumindo que todas essas outras camadas estão completas (DevOps, Automação de dados e Automação de plataforma), o MLOps é possível. Lembra-te que o processo de automatização da aprendizagem automática utilizando metodologias DevOps é MLOps. O método de construção da aprendizagem automática é a engenharia de aprendizagem automática.

Como resultado, o MLOps é um comportamento, assim como o DevOps é um comportamento. Embora algumas pessoas trabalhem como engenheiros

de DevOps, um engenheiro de software realizará mais frequentemente tarefas utilizando as melhores práticas de DevOps. Da mesma forma, um engenheiro de aprendizagem automática deve utilizar as melhores práticas de MLOps para criar sistemas de aprendizagem automática.

Combinaste as melhores práticas de DevOps e MLOps?

Lembras-te das práticas DevOps descritas anteriormente no capítulo? O MLOps baseia-se nessas práticas e estende itens específicos para visar diretamente os sistemas de aprendizado de máquina.

Uma forma de articular estas boas práticas é considerar que elas criam modelos reproduzíveis com um robusto empacotamento, validação e implementação de modelos. Além disso, elas aumentam a capacidade de explicar e observar o desempenho do modelo. **A Figura 1-10** mostra isso com mais detalhes.

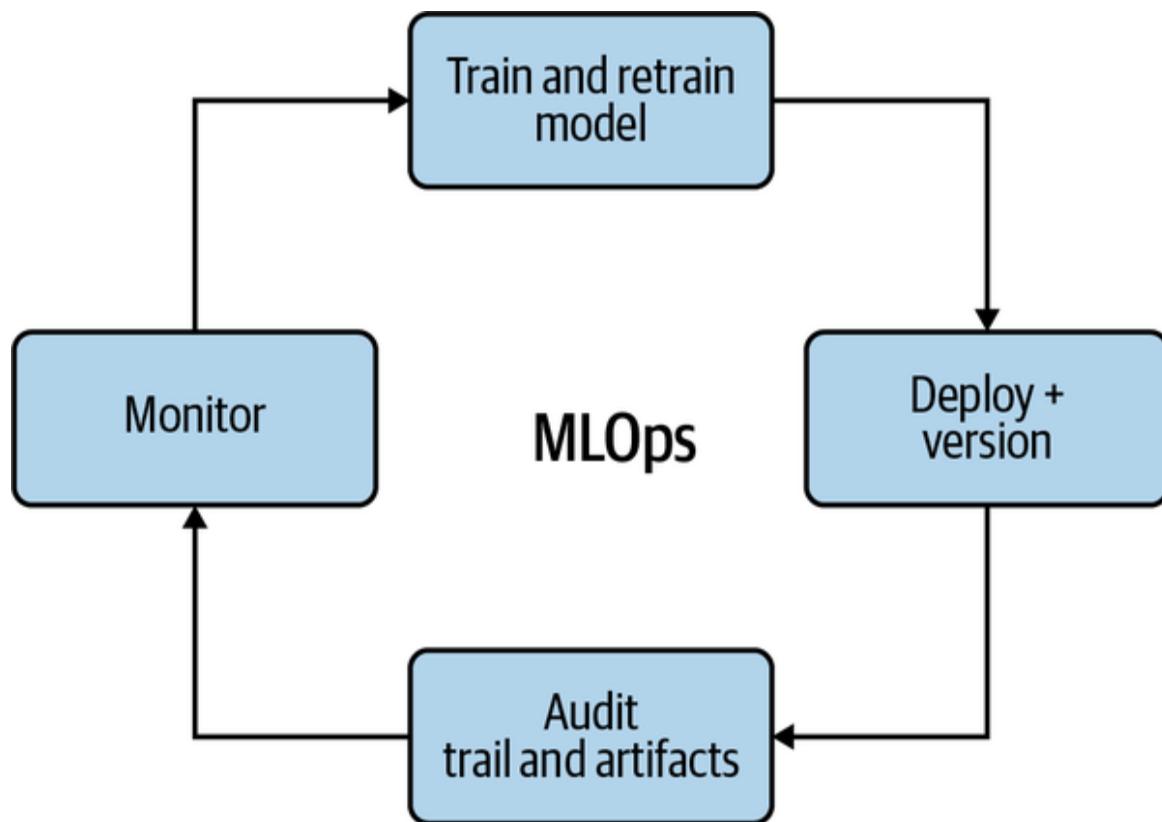


Figura 1-10. Loop de feedback do MLOps

O ciclo de feedback inclui o seguinte:

Cria e treina novamente modelos com pipelines de ML reutilizáveis

Criar um modelo apenas uma vez não é suficiente. Os dados podem mudar, os clientes podem mudar e as pessoas que criam os modelos podem mudar. A solução é ter pipelines de ML reutilizáveis que são versionados.

Entrega contínua de modelos ML

A entrega contínua de modelos de ML é semelhante à entrega contínua de software. Quando todos os passos são automatizados, incluindo a infraestrutura, utilizando a IaC, o modelo pode ser implementado em qualquer altura num novo ambiente, incluindo a produção.

Pista de auditoria para o pipeline MLOps

É fundamental ter uma auditoria para os modelos de aprendizagem automática. Não faltam problemas na aprendizagem automática, incluindo segurança, parcialidade e exatidão. Por conseguinte, ter uma pista de auditoria útil é inestimável, tal como ter um registo adequado é fundamental em projectos de engenharia de software de produção. Além disso, a pista de auditoria faz parte do ciclo de feedback em que se melhora continuamente a abordagem ao problema e o problema real.

Observa a utilização dos dados do modelo para melhorar os modelos futuros

Um dos aspectos únicos da aprendizagem automática é que os dados podem literalmente "mudar" sob o modelo. Assim, o modelo que funcionou para os clientes há dois anos provavelmente não funcionará da mesma forma hoje. Ao monitorizar a deriva dos dados, ou seja, o delta das alterações desde a última vez que ocorreu um treino do modelo, é possível evitar problemas de precisão antes de causar problemas de produção.

ONDE PODES IMPLANTAR?

Um aspecto fundamental do MLOps é criar um modelo de plataforma nativo da Cloud e, em seguida, implantá-lo em muitos alvos diferentes, conforme mostrado na [Figura 1-11](#). Essa capacidade de criar uma vez e implantar várias vezes é um recurso essencial nas operações modernas de aprendizado de máquina. Por um lado, implantar um modelo em um ponto de extremidade HTTP que pode ser escalonado elasticamente é um padrão típico, mas não é a única maneira. Um novo paradigma de aprendizagem automática de ponta utiliza processadores dedicados e especializados chamados ASICS. Exemplos disso incluem o TPU do Google e o A14 da Apple.

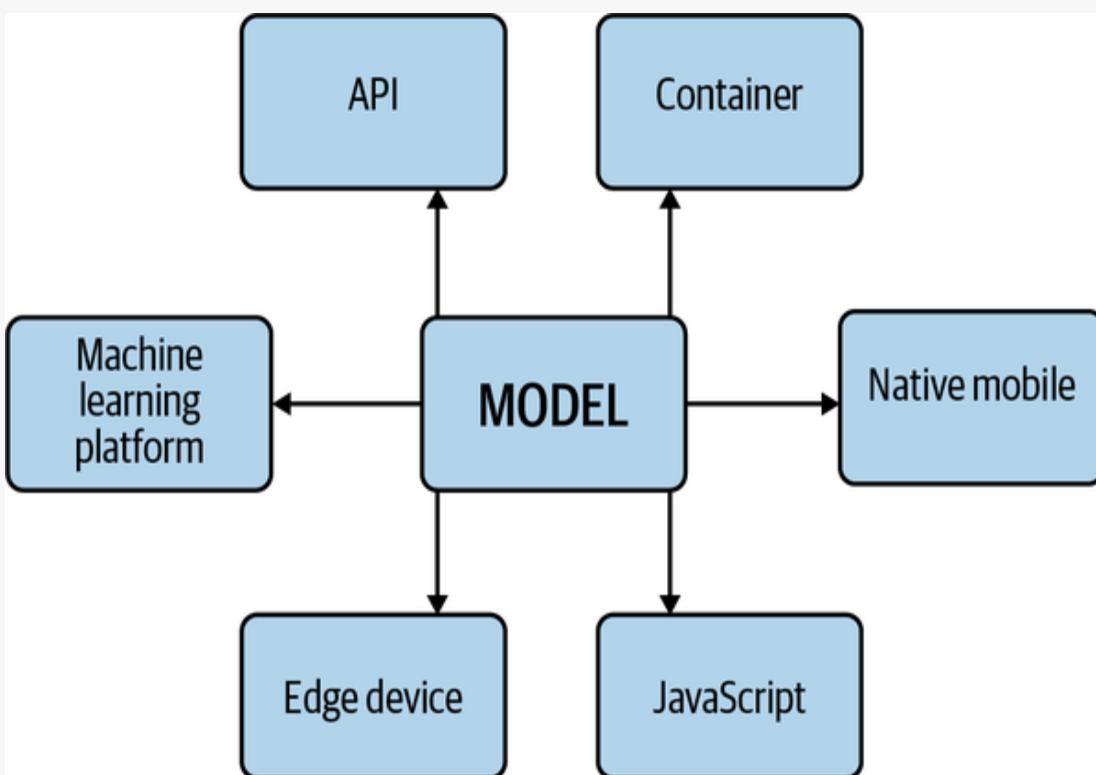


Figura 1-11. Objectivos do modelo de aprendizagem automática

Na [Figura 1-12](#), a plataforma Cloud pode utilizar o AutoML, como na visão AutoML da Google, que implementa o TensorFlow no TF Lite, TensorFlow.js, Core ML (uma estrutura ML da Apple), um Container ou Coral (hardware de ponta que utiliza uma TPU).

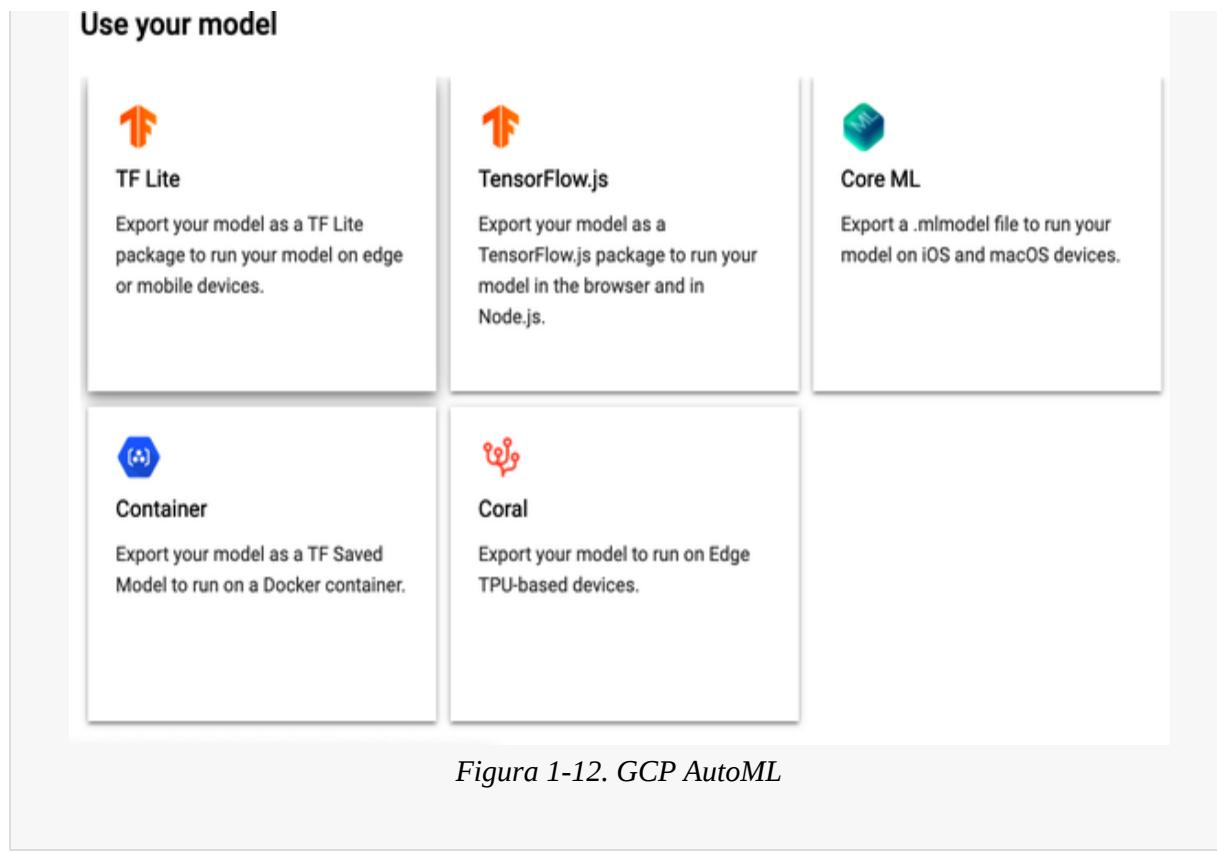


Figura 1-12. GCP AutoML

Conclusão

Este capítulo discutiu a importância da utilização dos princípios DevOps no contexto da aprendizagem automática. Para além do software, a aprendizagem automática acrescenta as novas complexidades da gestão dos dados e do modelo. A solução para esta complexidade é adotar a automação da mesma forma que a comunidade de engenharia de software o fez com o DevOps.

Construir uma estante é diferente de plantar uma árvore. Uma estante de livros requer um projeto inicial e depois uma construção única. Os sistemas de software complexos que envolvem aprendizagem automática são mais parecidos com o crescimento de uma árvore. Uma árvore que cresce com sucesso requer múltiplas entradas dinâmicas, incluindo solo, água, vento e sol.

Da mesma forma, uma maneira de pensar sobre MLOps é a regra dos 25%. Na [Figura 1-13](#), a engenharia de software, a engenharia de dados, a modelação e o problema de negócio são igualmente importantes. O aspetto multidisciplinar do MLOps é o que o torna difícil de fazer. No entanto, há muitos bons exemplos de empresas que fazem MLOps seguindo a regra dos 25%.

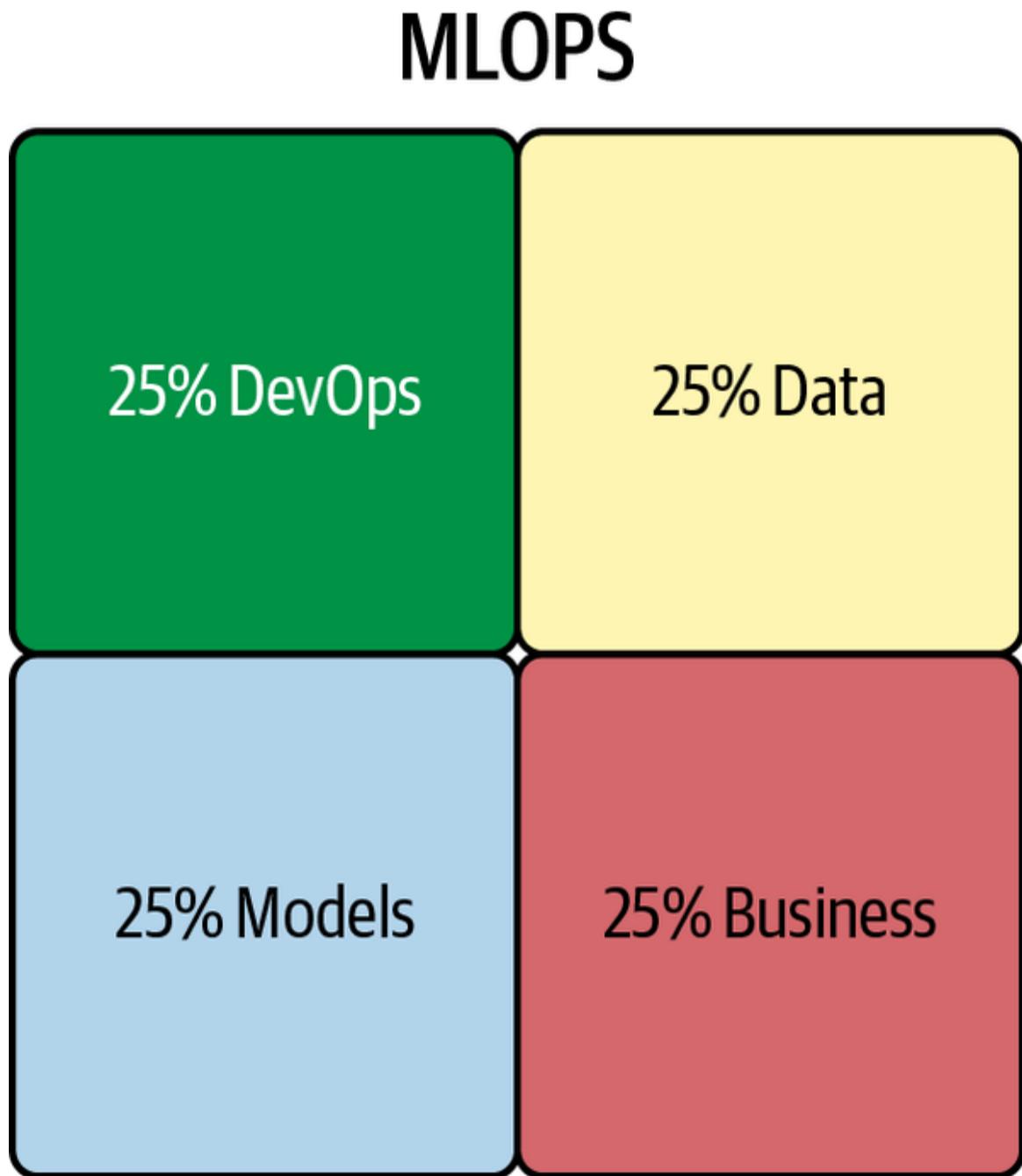


Figura 1-13. Regra dos 25%

Os carros da Tesla são um bom exemplo; fornecem aos clientes o que eles querem sob a forma de veículos semi-autónomos. Também têm excelentes práticas de engenharia de software, na medida em que fazem actualizações constantes. Ao mesmo tempo, o sistema do carro treina continuamente o modelo para melhorar com base nos novos dados que recebe. Outro exemplo de um produto que segue a regra dos 25% é o dispositivo Amazon Alexa.

As habilidades básicas necessárias para MLOps são discutidas no próximo capítulo. Inclui matemática para programadores, exemplos de projetos de ciência de dados e um processo completo de MLOps de ponta a ponta. Ao fazeres os exercícios recomendados no final deste capítulo, colcas-te numa excelente posição para absorveres o conteúdo seguinte.

Exercícios

- Cria um novo repositório GitHub com o scaffolding Python necessário utilizando um `Makefile`, linting e testes. Em seguida, executa etapas adicionais, como a formatação de código no teu `Makefile`.
- Utilizando [GitHub Actions](#), testa um projeto GitHub com duas ou mais versões Python.
- Utilizando um servidor de compilação nativo da Cloud (AWS Code Build, GCP CloudBuild ou Azure DevOps Pipelines), executa a integração contínua do teu projeto.
- Contentoriza um projeto GitHub integrando um `Dockerfile` e registando automaticamente novos contentores num Registo de Contentores.
- Cria um teste de carga simples para a tua aplicação utilizando uma estrutura de teste de carga como o `locust` ou o `loader io` e executa automaticamente este teste quando fazes push das alterações para um ramo de staging.

Questões para discussão sobre pensamento crítico

- Que problemas resolve um sistema de integração contínua (IC)?
- Porque é que um sistema de CI é uma parte essencial de um produto de software SaaS e de um sistema de ML?
- Porque é que as plataformas Cloud são o alvo ideal para as aplicações analíticas? Como é que a engenharia de dados e o DataOps ajudam na criação de aplicações analíticas baseadas na Cloud?
- Como é que a aprendizagem profunda beneficia com a Cloud? A aprendizagem profunda é viável sem a computação em Cloud?
- Explica o que é o MLOps e como pode melhorar um projeto de engenharia de aprendizagem automática.

Capítulo 2. Fundamentos do MLOps

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

A escola de medicina foi uma experiência lamentável, uma ladainha interminável de factos, cujas origens raramente eram explicadas e cuja utilidade raramente era justificada. A minha aversão à aprendizagem mecânica e a minha atitude questionadora não eram partilhadas pela maioria dos 96 alunos da turma. Isto foi particularmente evidente numa ocasião em que um professor de bioquímica afirmou estar a deduzir a equação de Nernst. A turma copiava fielmente o que ele escrevia no quadro. Tendo frequentado apenas um ano antes o curso de Pchem para licenciados em química na UCLA, pensei que ele estava a fazer bluff.

"Onde foste buscar esse valor para k?" perguntei-te.

A turma gritou-me: "Deixa-o acabar! Copia-o".

—Dr. Joseph Bogen

Ter uma base sólida sobre a qual construir é fundamental para qualquer empreendimento técnico. Neste capítulo, vários blocos de construção fundamentais estabelecem a base para o resto do livro. Ao lidar com estudantes iniciantes em ciência de dados e aprendizado de máquina, geralmente encontro concepções equivocadas sobre os itens abordados neste capítulo. Este capítulo tem como objetivo criar uma base sólida para a utilização das metodologias MLOps.

Bash e a linha de comando do Linux

A maior parte do aprendizado de máquina acontece na Cloud, e a maioria das plataformas Cloud assume que vais interagir com ele até certo ponto com o terminal. Como tal, é fundamental conhecer os conceitos básicos da linha de comando do Linux para fazer MLOps. Esta seção tem como objetivo Bootstrap com conhecimento suficiente para garantir que tenhas sucesso fazendo MLOps.

É frequente ouvires um olhar de choque e de horror quando apresento o terminal a um aluno. A reação inicial é razoável na maioria das áreas de computação modernas, devido ao poder das interfaces GUI, como o sistema operativo MacOS ou o Windows. No entanto, uma melhor forma de pensar no terminal são as "definições avançadas" do ambiente em que estás a trabalhar: a Cloud, a aprendizagem automática ou a programação. Se precisares de fazer tarefas avançadas, é a forma de as realizar. Como resultado, a competência no terminal Linux pode melhorar enormemente qualquer conjunto de habilidades. Além disso, é melhor desenvolveres num ambiente cloud shell, na maioria dos casos, o que pressupõe familiaridade com Bash e Linux.

A maioria dos servidores correm agora Linux; muitas das novas opções de implementação estão a utilizar contentores que também correm Linux. O terminal do sistema operativo MacOS é suficientemente próximo do Linux para que a maioria dos comandos sejam semelhantes, especialmente se instalares ferramentas de terceiros como o [Homebrew](#). Deves conhecer o terminal bash, e esta secção vai dar-te conhecimentos suficientes para serescompetentecom ele.

Quais são os componentes críticos e minimalistas do terminal a aprender? Estes componentes incluem a utilização de um ambiente de desenvolvimento shell baseado em Cloud, shell e comandos Bash, ficheiros e navegação, entrada/saída, configuração e escrita de um script. Então, vamos mergulhar em cada um desses tópicos.

Ambientes de desenvolvimento Cloud Shell

Quer sejas novo na computação Cloud ou tenhas décadas de experiência, vale a pena mudar de uma estação de trabalho pessoal para um ambiente de desenvolvimento baseado na Web. Uma boa analogia é um surfista que quer surfar todos os dias na praia. Em teoria, ele poderia dirigir 50 milhas para cada lado da praia todos os dias, mas isso seria muito inconveniente, ineficiente e caro. A melhor estratégia, se tivesses dinheiro para isso, seria viver na praia, acordar todas as manhãs, caminhar até à praia e surfar.

Da mesma forma, existem vários problemas que um ambiente de desenvolvimento Cloud resolve: é mais seguro, uma vez que não precisas de passar chaves de programador. Muitos problemas são intratáveis utilizando uma máquina local, uma vez que não podes transferir dados extensos para trás e para a frente a partir da Cloud. As ferramentas disponíveis no desenvolvimento baseado na Cloud incluem uma integração profunda, o que torna o trabalho mais eficiente. Ao contrário de te mudares para a praia, um ambiente de desenvolvimento na Cloud é gratuito. Todas as principais nuvens disponibilizam os seus ambientes de desenvolvimento na nuvem em níveis gratuitos. Se fores novo nestes ambientes, recomendo que comeceis com a plataforma AWS Cloud. Existem duas opções para começares a utilizar a AWS. A primeira opção é o AWS CloudShell mostrado na [Figura 2-1](#).

O AWS CloudShell é um shell Bash com conclusão de comando AWS exclusiva incorporada ao shell. Se usas regularmente o AWS CloudShell, é uma boa ideia editar o `~/.bashrc` para personalizar a tua experiência. Para fazer isso, podes usar o editor `vim` incorporado. Muitas pessoas adiam a aprendizagem do `vim`, mas têm de ser proficientes na era do CloudShell. Podes consultar a [FAQ oficial do vim](#) para saberes como fazer as coisas.

The screenshot shows the AWS CloudShell interface. At the top, there's a navigation bar with the AWS logo, 'Services ▾', a search bar ('Search for services, features, marketplace products: [Option+S]'), and a 'CloudShell' button. Below the navigation bar, the title 'AWS CloudShell' is displayed. Underneath, a section for 'us-east-1' is shown. The terminal window displays the message 'Preparing your terminal...', followed by 'A Python virtualenv is sourced'. It then shows a command prompt: '(.venv) [cloudshell-user@ip-10-1-191-33 ~]\$ Try these commands to get started: aws help or aws <command> help or aws <command> --cli-auto-prompt'. The background of the terminal window is dark grey.

Figura 2-1. AWS CloudShell

Uma segunda opção na AWS é o ambiente de desenvolvimento do AWS Cloud9. Uma diferença crítica entre o AWS CloudShell e o ambiente do AWS Cloud9 é que ele é uma maneira mais abrangente de desenvolver soluções de software. Por exemplo, podes ver um shell e um editor GUI na **Figura 2-2** para fazer realce de sintaxe para várias linguagens, incluindo Python, Go e Node.

The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a sidebar with tabs for 'Filesystem', 'AWS Cloud9', 'File', 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', 'Support', 'Preview', 'Run', 'Share', and 'Settings'. Below this is a 'Source Control' section with a tree view of the project structure, including 'Computer-Vision', '.git', 'computer-vision-api', 'workflows', 'main.yml', 'pytest_cache', '_pycache__', 'hello.py', 'Makefile', 'README.md', 'requirements.txt', 'test_hello.py', 'Marvel.jpg', and 'README.md'. The main area is a code editor with tabs for 'requirements.txt', 'Makefile', 'hello.py', 'test_hello.py', and 'main.yml'. The 'hello.py' tab is active, displaying Python code related to AWS Rekognition. At the bottom, there's a terminal window titled '(.venv) ec2-user:~/environment \$' showing a command prompt. The interface has a light blue and white color scheme.

Figura 2-2. Ambiente de desenvolvimento do AWS Cloud9

Em particular, ao desenvolver microsserviços de aprendizagem automática, o ambiente Cloud9 é ideal, uma vez que te permite fazer pedidos Web a partir da consola para serviços implementados e tem uma integração profunda com o AWS Lambda. Por outro lado, supõe que estás noutra

plataforma, como o Microsoft Azure ou o Google Cloud. Nesse caso, aplicam-se os mesmos conceitos, uma vez que os ambientes de desenvolvimento baseados na Cloud são o local ideal para criar serviços de aprendizagem automática.

NOTA

Existe um recurso de vídeo opcional que criei, chamado "Bash Essentials for Cloud Computing", que te orienta no básico. Podes vê-lo na [plataforma O'Reilly](#) ou no [canal do YouTube da Pragmatic AI Labs](#).

Shell e Comandos Bash

A shell é um ambiente interativo que contém um prompt e a capacidade de executar comandos. A maioria dos shells hoje em dia executa o Bash ou o ZSH.

Um par de coisas imediatamente valiosas a fazer num ambiente que usas normalmente para desenvolvimento é instalar configurações ZSH e vim. Para vim, uma configuração recomendada é o [awesome vim](#), e para o ZSH, existe o [ohmyzsh](#).

O que é a "shell"? Em última análise, é uma interface de utilizador que controla um computador, tal como o Finder do MacOS. Como praticantes de MLOps, vale a pena saber como utilizar a interface de utilizador mais robusta, a linha de comandos, para pessoas que trabalham com dados. Aqui estão algumas coisas que podes fazer.

Listá de ficheiros

Com a shell, podes listar ficheiros através do comando ls. A flag -l acrescenta informação adicional de listagem:

```
bash-3.2$ ls -l
total 11
drwxrwxr-x 130 root admin 4160 Jan 20 22:00 Applications
```

```
drwxr-xr-x  75 root wheel 2400 Dec 14 23:13 Library
drwxr-xr-x@  9 root wheel  288 Jan 1 2020 System
drwxr-xr-x   6 root admin  192 Jan 1 2020 Users
```

Executa comandos

Numa GUI, clica num botão ou abre uma aplicação para trabalhar. Na shell, executa um comando. Existem muitos comandos internos úteis no shell, e eles geralmente funcionam bem juntos. Por exemplo, uma excelente maneira de descobrir a localização dos executáveis do shell é usar `which`. Aqui está um exemplo:

```
bash-3.2$ which ls
/bin/ls
```

Repara que o comando `ls` está no diretório `/bin`. Essa "dica" mostra que eu posso encontrar outros executáveis nesse diretório. Aqui está uma contagem dos executáveis em `/bin/` (o operador pipe | será explicado daqui a pouco, mas, em resumo, aceita a entrada de outro comando):

```
bash-3.2$ ls -l /bin/ | wc -l
37
```

Ficheiros e navegação

Numa GUI, abres uma pasta ou um ficheiro; numa shell, utilizas comandos para fazer a mesma coisa.

`pwd` mostra o caminho completo para onde estás:

```
bash-3.2$ pwd
/Users/noahgift
```

`cd` muda para um novo diretório:

```
bash-3.2$ cd /tmp
```

Entrada/Saída

Em um exemplo anterior, a saída de `ls` redireciona para outro comando. O pipe é um exemplo de operações de entrada e saída trabalhando para realizar uma tarefa mais sofisticada. É comum usar o shell para canalizar um comando para outro.

Aqui está um exemplo que mostra um fluxo de trabalho com um redireccionamento e um pipe. Repara que primeiro, as palavras "foo bar baz" são direcionadas para um ficheiro chamado `out.txt`. A seguir, o conteúdo deste ficheiro é impresso através de `cat`, e depois é canalizado para o comando `wc`, que pode contar o número de palavras através de `-w` ou de caracteres através de `-c`:

```
bash-3.2$ cd /tmp
bash-3.2$ echo "foo bar baz" > out.txt
bash-3.2$ cat out.txt | wc -c
      12
bash-3.2$ cat out.txt | wc -w
      3
```

Aqui está outro exemplo que direciona a saída do comando `shuf` para um novo arquivo. Podes descarregar esse ficheiro do meu [repositório GitHub](#). O executável `shuf` "embaralha" um arquivo, limitando-o às linhas especificadas. Nesse caso, pega um arquivo de quase 1 GB, pega as primeiras 100.000 linhas e gera um novo arquivo usando o operador `>`:

```
bash-3.2$ time shuf -n 100000 en.openfoodfacts.org.products.tsv
>\ 
10k.sample.en.openfoodfacts.org.products.tsv
1.89s user 0.80s system 97% cpu 2.748 total
```

Utilizar uma técnica de shell como esta pode salvar o dia quando se trabalha num ficheiro CSV demasiado grande para processar bibliotecas de ciência de dados num portátil.

Configuração

Os ficheiros de configuração da shell ZSH e Bash armazenam definições que são invocadas sempre que um terminal é aberto. Como mencionei

anteriormente, a personalização do teu ambiente Bash é recomendada num ambiente de desenvolvimento Cloud. Para o ZSH, um excelente lugar para começar é `.zshrc`, e para o Bash, é `.bashrc`. Aqui está um exemplo de algo que eu armazeno na minha configuração `.zshrc` no meu laptop MacOS. O primeiro item é um alias que me permite digitar o comando `flask-azure-ml`, cd num diretório, e criar um ambiente virtual Python de uma só vez. A segunda secção é onde exporto as variáveis da ferramenta de linha de comandos da AWS para poder fazer chamadas à API:

```
## Flask ML Azure
alias flask-azure-ml="/Users/noahgift/src/flask-ml-azure-
serverless && \
source ~/.flask-ml-azure/bin/activate"

## AWS CLI
export AWS_SECRET_ACCESS_KEY=""
export AWS_ACCESS_KEY_ID=""
export AWS_DEFAULT_REGION="us-east-1"
```

Em resumo, a minha recomendação é personalizar o teu ficheiro de configuração da shell tanto para o teu portátil como para ambientes de desenvolvimento baseados na Cloud. Este pequeno investimento paga grandes dividendos à medida que constróis automação para os teus fluxos de trabalho regulares.

Escrever um guião

Pode ser um pouco assustador pensar em escrever o teu primeiro script de shell. A sintaxe é muito mais assustadora do que a do Python, com caracteres estranhos. Felizmente, em muitos aspectos, é mais fácil começares. A melhor maneira de escrever um script de shell é colocar um comando num ficheiro e depois executá-lo. Aqui tens um excelente exemplo de um script "hello world".

A primeira linha é chamada de linha "shebang" e diz ao script para usar o Bash. A segunda linha é um comando Bash, `echo`. O bom de um script Bash é que podes colar nele os comandos que quiseres. Este facto torna a

automatização de pequenas tarefas simples, mesmo com pouco ou nenhum conhecimento de programação:

```
#!/usr/bin/env bash  
  
echo "hello world"
```

Em seguida, utiliza o comando `chmod` para definir o sinalizador executável para tornar este script executável. Finalmente, executa-o anexando `./`:

```
bash-3.2$ chmod +x hello.sh  
bash-3.2$ ./hello.sh  
Hello World
```

A principal vantagem da shell é que tens de ter pelo menos algumas competências básicas para fazer MLOps. No entanto, é fácil começar e, antes que te apercebas, podes melhorar drasticamente as coisas que fazes diariamente através da automatização de scripts de shell e da utilização da linha de comandos Linux. Em seguida, vamos falar sobre como começar com uma visão geral das partes essenciais da computação em Cloud.

Fundamentos e blocos de construção da computação em Cloud

É seguro dizer que quase todas as formas de aprendizagem automática requerem a computação em Cloud de alguma forma. Um conceito-chave na computação em nuvem é a ideia de recursos quase infinitos, como descrito em "[Above the Clouds: Uma visão de Berkeley da computação em nuvem](#)". Sem a Cloud, simplesmente não é viável fazer muitos modelos de aprendizagem automática. Por exemplo, Stephen Strogatz, o autor de *Infinite Powers: How Calculus Reveals the Secrets of the Universe* (Mariner Books), defende que "ao utilizar o infinito da forma correta, o cálculo pode desvendar os segredos do universo". Durante séculos, problemas específicos como encontrar a forma de um círculo eram impossíveis sem o cálculo para lidar com números infinitos. O mesmo acontece com a

computação em Cloud; muitas questões de aprendizagem automática, especialmente a operacionalização de modelos, não são viáveis sem a Cloud. Como mostra a [Figura 2-3](#), a Cloud fornece computação e armazenamento quase infinitos e trabalha com os dados sem os mover.

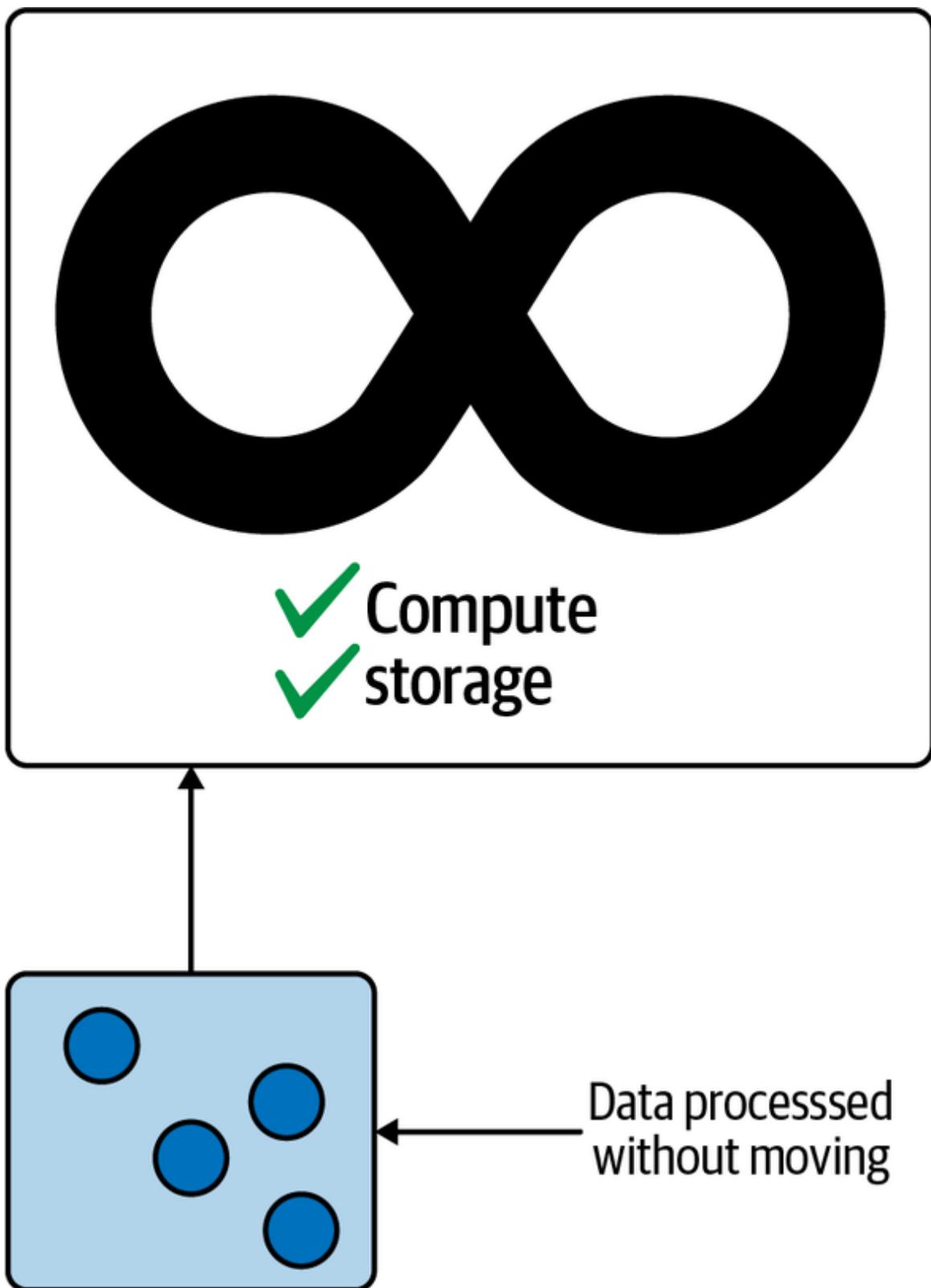


Figura 2-3. A computação em Cloud aproveita a computação e os dados quase infinitos

Acontece que a capacidade de aproveitar o poder de computação dos dados sem os mover, utilizando recursos quase infinitos através de plataformas de aprendizagem automática como o AWS SageMaker ou o Azure ML Studio, é a característica fundamental da Cloud que não pode ser replicada sem a computação em nuvem. Juntamente com esta característica é algo a que chamo a "lei do automatizador". Quando o público em geral começa a falar sobre a automatização de um sector - carros autónomos, TI, fábricas, aprendizagem automática - ela acaba por acontecer.

Este conceito não significa que apareça um unicórnio mágico que espalhe pó de fada nos projectos e estes se tornem mais fáceis de gerir; significa que os humanos são bons a detetar tendências enquanto grupo coletivo. Por exemplo, quando trabalhei na indústria televisiva em adolescente, só existia o conceito de edição "linear". Este fluxo de trabalho significava que precisavas de três cassetes diferentes para dissolver para um ecrã negro - a cassette de origem, a cassette principal de edição e uma terceira cassette que continha a filmagem negra.

Lembro-me de as pessoas falarem do trabalho que dava estar sempre a trocar as cassetes e de como seria fantástico se este fluxo de trabalho fosse automatizado. Mais tarde, tornou-se totalmente automatizado com a introdução da edição não linear. Esta tecnologia permite-te armazenar cópias digitais do material e efetuar a manipulação digital das filmagens em vez de inserir novo material numa fita linear. Estes sistemas de edição não linear custavam centenas de milhares de dólares no início da década de 1990. Agora faço edições mais complicadas no meu portátil de mil dólares com capacidade de armazenamento suficiente para guardar milhares de cassetes deste tipo.

O mesmo cenário ocorreu com a computação em Cloud no início dos anos 2000. Muitas empresas em que trabalhei utilizavam os seus próprios centros de dados com equipas de pessoas que os mantinham. Quando surgiram os primeiros componentes da computação em Cloud, muitas pessoas disseram: "Aposto que no futuro as empresas poderão controlar um centro de dados inteiro no seu portátil". Muitos técnicos especializados em centros de dados zombaram da ideia de o seu trabalho ser vítima da automatização, mas a lei

do automatizador voltou a atacar. A maioria das empresas em 2020 e nos anos seguintes tem alguma forma de computação Cloud, e os novos empregos envolvem o aproveitamento desse poder.

Da mesma forma, a automatização da aprendizagem automática através do AutoML é um avanço não trivial que permite a criação de modelos mais rapidamente, com maior precisão e melhor explicabilidade. Como resultado, os empregos no sector da ciência dos dados vão mudar, tal como mudaram os empregos dos editores e dos operadores de centros de dados.

NOTA

O AutoML é a automatização do aspeto de modelação da aprendizagem automática. Um exemplo simples e direto de AutoML é uma folha de cálculo do Excel que executa uma regressão linear. Diz ao Excel que coluna é o objetivo a prever e, em seguida, que coluna é a característica.

Os sistemas AutoML mais sofisticados funcionam de forma semelhante. O usuário seleciona o valor que deseja prever - por exemplo, classificação de imagens, tendência numérica, classificação de texto categórico ou talvez clustering. Em seguida, o sistema de software do AutoML executa muitas das mesmas técnicas que um cientista de dados executaria, incluindo o ajuste de hiperparâmetros, a seleção de algoritmos e a explicabilidade do modelo.

Todas as principais plataformas Cloud têm ferramentas AutoML incorporadas nas plataformas MLOps. Como resultado, o AutoML é uma opção para todos os projectos de aprendizagem automática baseados na Cloud e está a tornar-se cada vez mais uma melhoria de produtividade para os mesmos.

Tyler Cowen é economista, autor e colunista da Bloomberg e cresceu a jogar xadrez de competição. No seu livro *Average is Over* (Plume), Cowen menciona que o software de xadrez acabou por vencer os humanos, provando também a lei do automatismo em ação. Surpreendentemente, no entanto, no final do livro de Cowen, os humanos especialistas e o software de xadrez ganharam contra o software de xadrez sozinho. Em última análise, esta história pode acontecer com a aprendizagem automática e a ciência dos dados. A automatização pode substituir tarefas simples de aprendizagem automática e tornar os especialistas no domínio que controlam a automatização do ML muito mais eficazes.

Começa a utilizar o Cloud Computing

Uma abordagem recomendada para começar a usar a computação em nuvem é configurar um ambiente de desenvolvimento multicloud, conforme mostrado no [curso em vídeo da O'Reilly: Computação em Cloud com Python](#). Este vídeo é um excelente companheiro para esta secção, mas não é necessário para o acompanhar. A estrutura básica de um ambiente multicloud mostra que um shell de nuvem é algo que todas essas nuvens têm em comum, como mostrado na [Figura 2-4](#).

Um repositório de controle de origem usando o GitHub, ou um serviço semelhante, é o local central onde todos os três ambientes de Cloud se comunicam inicialmente. Cada uma das três nuvens - AWS, Azure e GCP - tem ambientes de desenvolvimento baseados em nuvem por meio de um shell de nuvem. No [Capítulo 1](#), um scaffolding Python necessário mostrou as vantagens de desenvolver uma estrutura repetível e testável. Um processo CI/CD (integração contínua/entrega contínua) através do GitHub Actions garante que o código Python está a funcionar e é de alta qualidade.

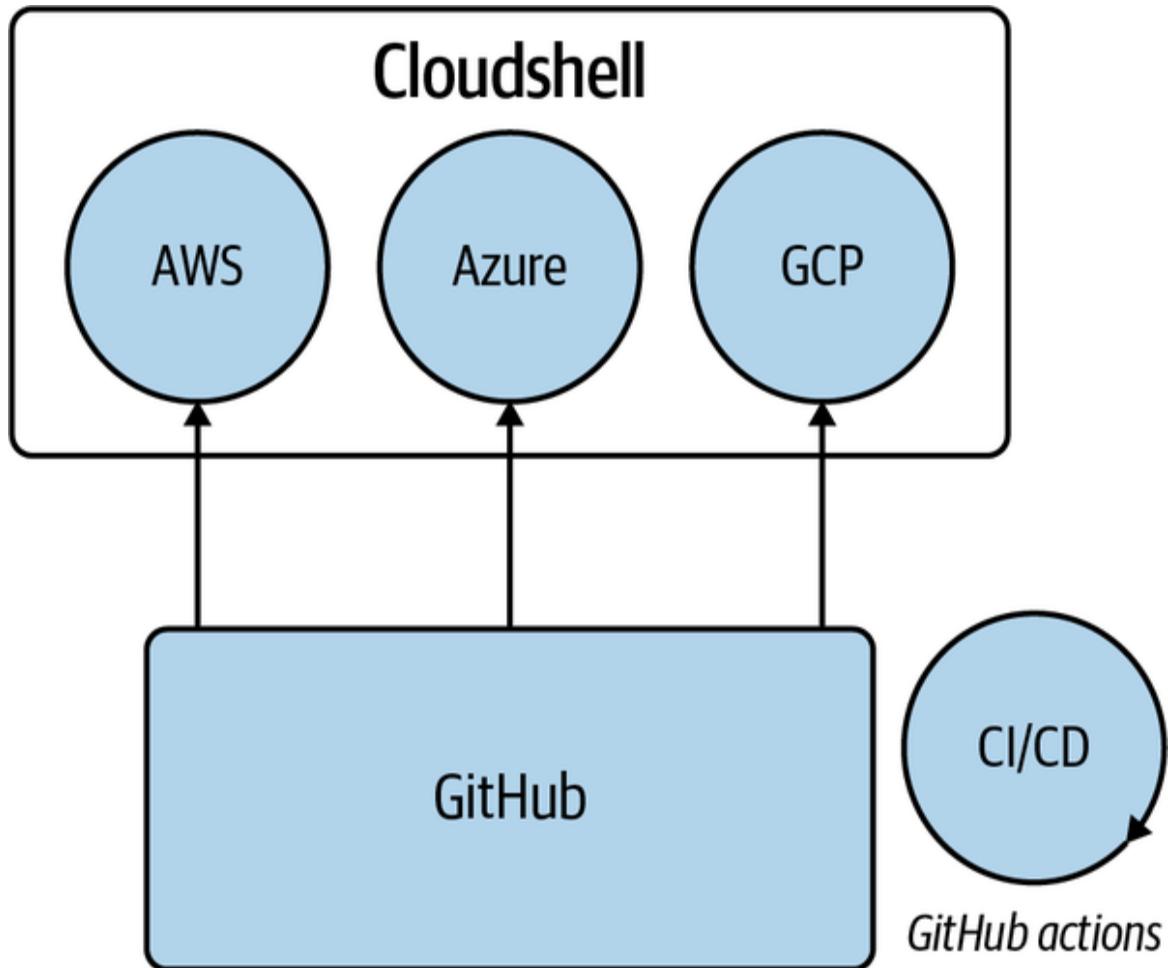


Figura 2-4. Inicia a computação em Cloud

NOTA

Testar e aplicar linting ao código Python é um processo que valida a qualidade de um projeto de software. Um programador irá executar testes e linting localmente para ajudar a manter a qualidade do software elevada. Este processo é semelhante a ter uma máquina robótica de aspiração (robovac) que ligas quando queres limpar uma divisão da tua casa. O robovac é um assistente útil que mantém o teu quarto num estado bom conhecido, e correr lints e testar o teu código mantém o teu código num estado bom conhecido.

Um pipeline de CI/CD executa estas verificações de controlo de qualidade num ambiente externo para garantir que a aplicação está a funcionar antes de ser lançada noutra produção. Este pipeline permite que a implementação de software seja repetível e fiável e é uma prática recomendada de engenharia de software moderna - outra palavra para estas práticas recomendadas de engenharia de software é DevOps.

Utilizar as três principais nuvens é uma excelente forma de te familiarizares com a computação em nuvem, se estiveres a aprender a usar a nuvem. Esse fluxo de trabalho entre nuvens ajuda porque solidifica o conhecimento. Afinal de contas, os nomes das coisas são diferentes, mas os conceitos são idênticos. Se precisar de ajuda com algumas das terminologias, consulta o [Apêndice A](#).

A seguir, vamos mergulhar no Python, que é a linguagem de facto do DevOps, da computação em Cloud, da ciência dos dados e da aprendizagem automática.

Curso rápido de Python

Uma das principais razões para o domínio do Python é que a linguagem é optimizada para o programador e não para o computador. Linguagens como C ou C++ têm excelente desempenho porque são de "nível inferior", o que significa que o programador tem de trabalhar muito mais na resolução de um problema. Por exemplo, um programador de C deve alojar memória, declarar tipos e compilar um programa. Por outro lado, um programador Python pode colocar alguns comandos e executá-los, e muitas vezes há muito menos código em Python.

No entanto, por essa conveniência, o desempenho do Python é muito mais lento do que C, C#, Java e Go. Além disso, o Python tem limitações inerentes à própria linguagem, incluindo a falta de threads verdadeiras, a falta de um compilador JIT (Just in Time) e a falta de inferência de tipos encontrada em linguagens como C# ou F#. No entanto, com a computação Cloud, o desempenho da linguagem não está ligado a muitos problemas. Portanto, seria justo dizer que o desempenho do Python teve sorte accidentalmente por causa de duas coisas: computação em nuvem e containers. Com a computação em Cloud, o design é totalmente distribuído, construindo em cima dele usando tecnologias como AWS Lambda e AWS SQS (Simple Queuing Service). Da mesma forma, a tecnologia de contêineres como Kubernetes faz o trabalho pesado de construir sistemas distribuídos, então as threads Python se tornam subitamente irrelevantes.

NOTA

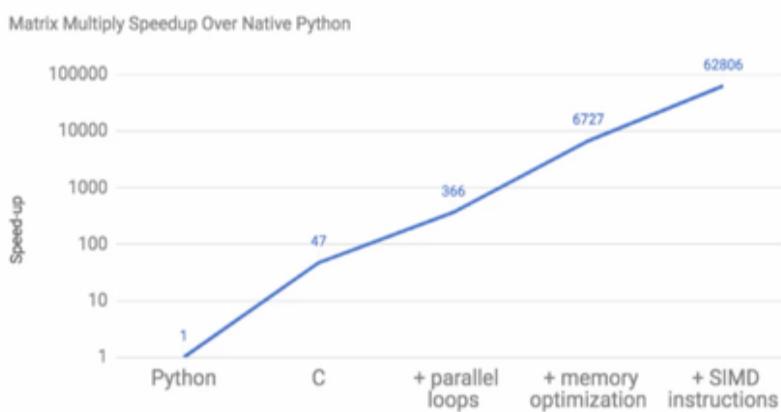
O AWS Lambda é uma tecnologia de função como serviço (FaaS) que é executada na plataforma AWS. Tem o nome FaaS porque uma função AWS Lambda pode ser apenas algumas linhas de código - literalmente uma função. Estas funções podem depois ser anexadas a eventos como um sistema de filas na Cloud, o Amazon SQS, ou uma imagem carregada para o armazenamento de objectos Amazon S3.

Uma forma de pensar sobre uma Cloud é que ela é um sistema operativo. Em meados da década de 1980, a Sun Computer usou a frase de marketing "The Network is the Computer" (A rede é o computador). Este slogan pode ter sido prematuro em 1980, mas em 2021 é muito preciso. Por exemplo, em vez de crieares threads numa única máquina, podes criar funções AWS Lambda na Cloud, que se comporta como um sistema operativo com recursos infinitamente escaláveis.

Numa palestra a que assisti na Google, o Dr. Patterson, um professor reformado de Ciências da Computação de Berkeley e cocriador da TPU (TensorFlow Processing Unit), mencionou que o Python é 64 000 vezes mais lento do que as operações matriciais equivalentes em C, mostradas na [Figura 2-5](#). Este facto é adicionalmente ao facto de não ter threads verdadeiras.

What's the Opportunity?

Matrix Multiply: relative speedup to a Python version (18 core Intel)



from: "There's Plenty of Room at the Top," Leiserson, et. al.,

17

Figura 2-5. O desempenho do Python é 64.000 vezes pior do que as operações de matriz em C

Simultaneamente, um trabalho de pesquisa, "Energy Efficiency across programming languages", mostra que muitas operações em Python requerem 50 vezes mais energia do que operações equivalentes em C. Esta pesquisa sobre eficiência energética também coloca Python como uma das linguagens com pior desempenho no que diz respeito à quantidade de energia que requer para realizar tarefas, em comparação com outras linguagens, como mostra a Figura 2-6. Como Python tem crescido em popularidade como uma das linguagens mais populares do planeta, levanta preocupações sobre se é mais como uma usina de carvão do que um sistema solar de energia verde. Em última análise, o Python pode precisar de ser novamente resgatado em termos de consumo de energia. Uma solução poderia ser um grande fornecedor de Cloud construirativamente um novo tempo de execução para Python que tire partido de técnicas modernas de informática, como um compilador JIT.

Um último obstáculo técnico que tenho visto nos recém-chegados à programação da ciência dos dados é a adoção da abordagem tradicional da informática para aprender a codificar. Por exemplo, a computação em Cloud e a aprendizagem automática são muito diferentes dos projectos convencionais de engenharia de software, como o desenvolvimento de aplicações orientadas para o utilizador através de uma GUI (interface gráfica do utilizador). Em vez disso, grande parte do mundo da Cloud e do ML envolve escrever pequenas funções. Na maioria das vezes, estas outras partes do Python não são necessárias, ou seja, código orientado para objectos, e desencorajam um recém-chegado de falar a linguagem.

	binary-trees			
	Energy	Time	Ratio	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust ↓ ₂	49.07	1263	0.039	180
(c) Fortran ↑ ₁	69.82	2112	0.033	133
(c) Ada ↓ ₁	95.02	2822	0.034	197
(c) Ocaml ↓ ₁ ↑ ₂	100.74	3525	0.029	148
(v) Java ↑ ₁ ↓ ₁₆	111.84	3306	0.034	1120
(v) Lisp ↓ ₃ ↓ ₃	149.55	10570	0.014	373
(v) Racket ↓ ₄ ↓ ₆	155.81	11261	0.014	467
(i) Hack ↑ ₂ ↓ ₉	156.71	4497	0.035	502
(v) C# ↓ ₁ ↓ ₁	189.74	10797	0.018	427
(v) F# ↓ ₃ ↓ ₁	207.13	15637	0.013	432
(c) Pascal ↓ ₃ ↑ ₅	214.64	16079	0.013	256
(c) Chapel ↑ ₅ ↑ ₄	237.29	7265	0.033	335
(v) Erlang ↑ ₅ ↑ ₁	266.14	7327	0.036	433
(c) Haskell ↑ ₂ ↓ ₂	270.15	11582	0.023	494
(i) Dart ↓ ₁ ↑ ₁	290.27	17197	0.017	475
(i) JavaScript ↓ ₂ ↓ ₄	312.14	21349	0.015	916
(i) TypeScript ↓ ₂ ↓ ₂	315.10	21686	0.015	915
(c) Go ↑ ₃ ↑ ₁₃	636.71	16292	0.039	228
(i) Jruby ↑ ₂ ↓ ₃	720.53	19276	0.037	1671
(i) Ruby ↑ ₅	855.12	26634	0.032	482
(i) PHP ↑ ₃	1,397.51	42316	0.033	786
(i) Python ↑ ₁₅	1,793.46	45003	0.040	275
(i) Lua ↓ ₁	2,452.04	209217	0.012	1961
(i) Perl ↑ ₁	3,542.20	96097	0.037	2148
(c) Swift			n.e.	

Figura 2-6. A linguagem Python está entre os piores infractores no consumo de energia

Os tópicos de ciência da computação incluem concorrência, programação orientada a objetos, metaprogramação e teoria de algoritmos. Infelizmente, o estudo desses tópicos é ortogonal ao estilo de programação necessário para a maior parte da programação em Cloud Computing e Data Science. Não é que esses tópicos não sejam valiosos; eles são benéficos para os criadores de plataformas, bibliotecas e ferramentas. Se, inicialmente, não

estás a "criar" bibliotecas e frameworks, mas sim a "usar" bibliotecas e frameworks, então podes ignorar com segurança estes tópicos avançados e ficar-te pelas funções.

Esta abordagem de curso intensivo ignora temporariamente o criador de código utilizado por outros em favor do consumidor de código e bibliotecas, ou seja, o cientista de dados ou o praticante de MLOps. Este breve curso intensivo é para o consumidor, onde a maioria das pessoas passa a sua carreira na ciência dos dados. Depois destes tópicos, se tiveres curiosidade, terás uma base sólida para avançar para tópicos mais complexos centrados nas ciências informáticas. Estes tópicos avançados não são necessários para seres produtivo imediatamente em MLOps.

Tutorial Python Minimalista

Se quisesses aprender o mínimo de Python necessário para começar, o que é que precisavas de saber? Os dois componentes mais essenciais do Python são as instruções e as funções. Por isso, vamos começar com as instruções Python. Uma declaração Python é uma instrução para um computador; ou seja, tal como dizer "olá" a uma pessoa, podes dizer a um computador para "imprimir olá". O exemplo seguinte é o interpretador Python. Repara, a "declaração" no exemplo é a frase `print("Hello World")`:

```
Python 3.9.0 (default, Nov 14 2020, 16:06:43)
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> print("Hello World")
Hello World
```

Com Python, também podes encadear duas declarações com um ponto e vírgula. Por exemplo, importo o módulo `os`, que tem uma função que quero utilizar, `os.listdir`, e depois chamo-a para listar o conteúdo de um diretório em que estou:

```
>>> import os;os.listdir(".")
['chapter11', 'README.md', 'chapter2', '.git', 'chapter3']
```

Este padrão é comum nos cadernos de ciência de dados e é tudo o que precisas de saber para começares a usar Python. Eu recomendaria que experimentasses as coisas em Python ou IPython, ou Jupyter REPL como a primeira forma de te familiarizares com Python.

O segundo item a saber é como escrever e usar funções em Python. Vamos fazer isso no exemplo seguinte. Este exemplo é uma função de duas linhas que soma dois números, x e y . O objetivo de uma função Python é servir como uma "unidade de trabalho". Por exemplo, uma torradeira na cozinha funciona como uma unidade de trabalho. Recebe o pão como entrada, aquece o pão e depois devolve a torrada. Da mesma forma, depois de escrever a função `add`, posso usá-la tantas vezes quanto possível com novas entradas:

```
>>> def add(x,y):
...     return x+y
...
>>> add(1,1)
2
>>> add(3,4)
7
>>>
```

Vamos reunir os nossos conhecimentos e construir um script Python. No início de um script Python, há uma linha shebang, tal como no Bash. De seguida, importa a biblioteca `choices`. Este módulo é mais tarde utilizado num "loop" para enviar números aleatórios para a função `add`:

```
#!/usr/bin/env python
from random import choices

def add(x,y):
    print(f"inside a function and adding {x}, {y}")
    return x+y

#Send random numbers from 1-10, ten times to the add function
numbers = range(1,10)
```

```
for num in numbers:  
    xx = choices(numbers)[0]  
    yy = choices(numbers)[0]  
    print(add(xx,yy))
```

O script tem de ser executado em `chmod +x add.py`, tal como um script Bash:

```
bash-3.2$ ./add.py  
inside a function and adding 7, 5  
12  
inside a function and adding 9, 5  
14  
inside a function and adding 3, 1  
4  
inside a function and adding 7, 2  
9  
inside a function and adding 5, 8  
13  
inside a function and adding 6, 1  
7  
inside a function and adding 5, 5  
10  
inside a function and adding 8, 6  
14  
inside a function and adding 3, 3  
6
```

Podes aprender muito mais sobre Python, mas "brincar" com exemplos como este é talvez a forma mais rápida de passar do "zero ao um". Por isso, vamos passar a outro tópico, matemática para programadores, e abordá-la também de uma forma minimalista.

Curso rápido de matemática para programadores

A matemática pode ser assustadora e irritante, mas compreender as noções básicas é essencial para trabalhar com a aprendizagem automática. Por isso, vamos abordar algumas ideias úteis e essenciais.

Estatística descritiva e distribuições normais

Muitas coisas no mundo têm uma distribuição "normal". Um bom exemplo é a altura e o peso. Se traçares a altura de cada pessoa no mundo, obterás uma distribuição em forma de sino. Esta distribuição é intuitiva, na medida em que a maioria das pessoas que encontras tem uma altura média e é invulgar ver um jogador de basquetebol com dois metros de altura. Vamos analisar um **caderno Jupyter** que contém 25 000 registos de alturas e pesos humanos de crianças de 19 anos:

In [0]:

```
import pandas as pd
```

In [7]:

```
df = pd.read_csv("https://raw.githubusercontent.com/noahgift/\nregression-concepts/master/height-weight-25k.csv")
```

Out[7]:

	Height-Inches	Weight-Pounds
01	65.78331	112.9925
12	71.51521	136.4873
23	69.39874	153.0269
34	68.21660	142.3354
45	67.78781	144.2971

Em seguida, um gráfico, apresentado na [Figura 2-7](#), mostra uma relação linear entre a altura e o peso, que a maioria de nós conhece intuitivamente. Quanto mais alto fores, mais pesas:

In [0]:

```
import seaborn as sns
import numpy as np
```

In [9]:

```
sns.lmplot("Height-Inches", "Weight-Pounds", data=df)
```

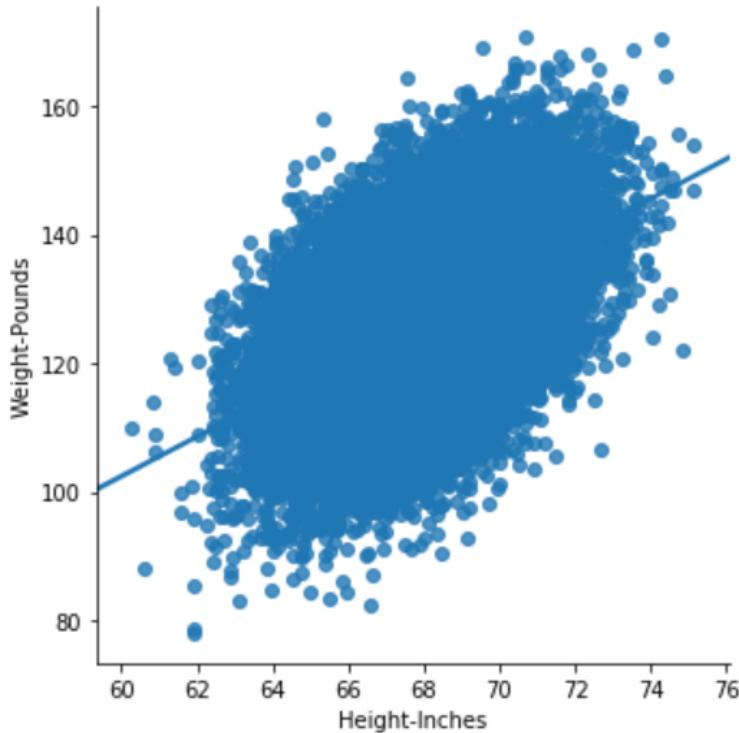


Figura 2-7. Altura e peso

O passo de visualização dos dados num conjunto de dados é designado por "Análise Exploratória de Dados". A ideia geral é "olhar à volta" utilizando uma combinação de matemática e visualização. O passo seguinte é olhar para as estatísticas descritivas desta "distribuição normal".

No Pandas, obtém estas estatísticas descritivas através da utilização de `df.describe()`. Uma forma de considerar as estatísticas descritivas é vê-las como uma forma de "ver" numericamente o que o olho vê visualmente. Por exemplo, o percentil 50, ou mediana, mostra o número que representa a altura média exacta. Este valor é cerca de 68 polegadas. A estatística máxima neste conjunto de dados é 75 polegadas. Max representa a observação mais extrema ou a pessoa mais alta medida para este conjunto de dados. A observação máxima num conjunto de dados normalmente distribuído é rara, tal como o mínimo. Pode ver esta tendência na [Figura 2-8](#). O DataFrame em Pandas vem com um método `describe`, quando chamado, fornece uma gama completa de estatísticas descritivas:

In [10]: `df.describe()`

	Index	Height-Inches	Weight-Pounds
count	25000.000000	25000.000000	25000.000000
mean	12500.500000	67.993114	127.079421
std	7217.022701	1.901679	11.660898
min	1.000000	60.278360	78.014760
25%	6250.750000	66.704397	119.308675
50%	12500.500000	67.995700	127.157750
75%	18750.250000	69.272958	134.892850
max	25000.000000	75.152800	170.924000

Figura 2-8. Estatísticas descritivas de altura/peso

Uma das melhores formas de visualizar a distribuição normal em forma de sino da altura e do peso é utilizar um gráfico de densidade de Kernel:

```
In [11]:  
sns.jointplot("Height-Inches", "Weight-Pounds", data=df,  
kind="kde");
```

Tanto o peso como a altura apresentam uma "distribuição em sino". Os valores extremos são raros e a maioria dos valores está no meio, como mostra a Figura 2-9.

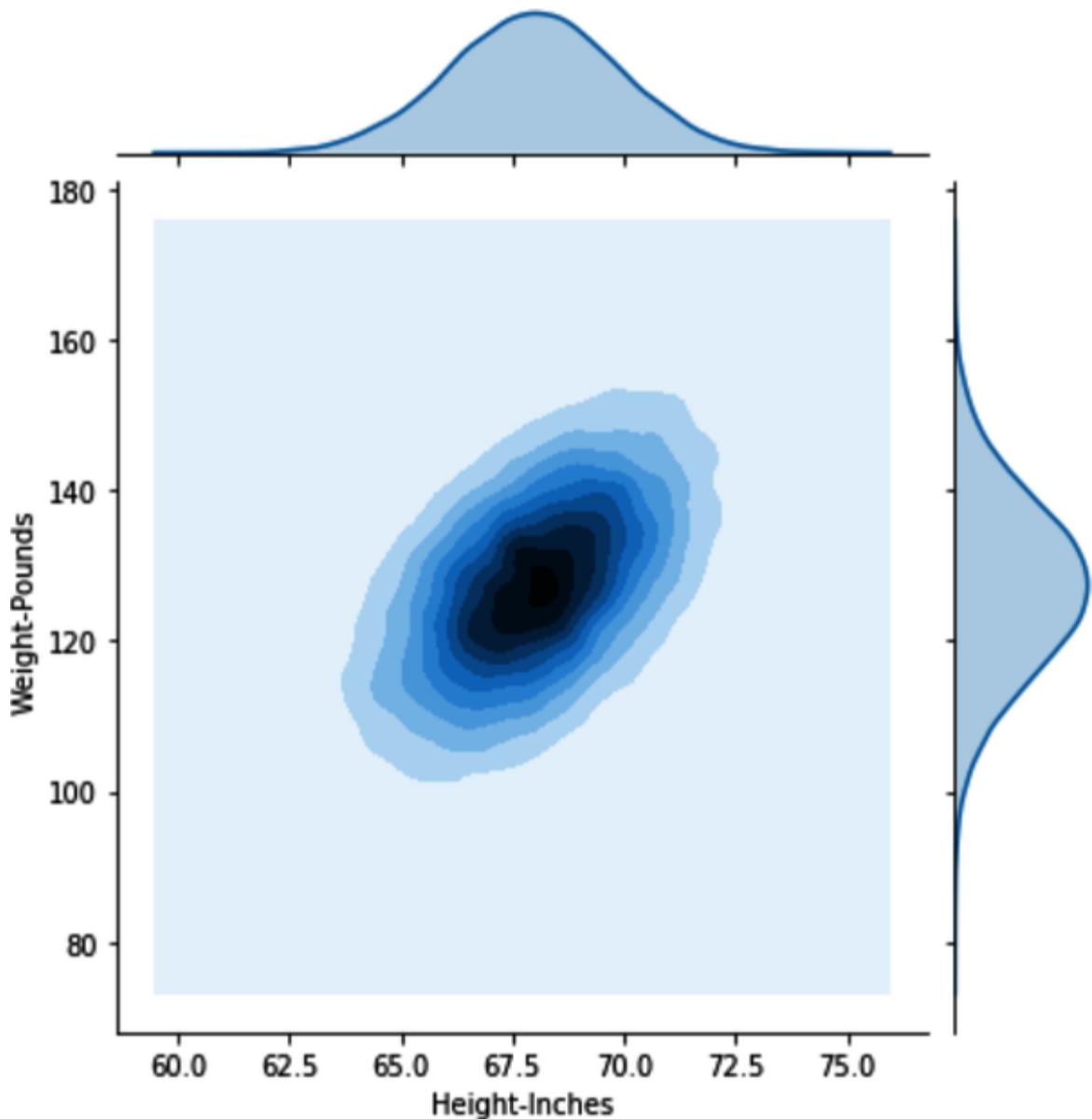


Figura 2-9. Gráfico da densidade de Kernel do peso e da altura

A aprendizagem automática baseia-se fortemente na ideia de uma distribuição normal, e ter esta intuição ajuda muito a construir e manter modelos de aprendizagem automática. No entanto, é essencial notar que outras distribuições estão para além da distribuição normal, tornando o mundo muito mais difícil de modelar. Um excelente exemplo disso é o que o autor Nassim Taleb chama de "caudas gordas", ou seja, eventos raros e difíceis de prever que afetam significativamente o mundo.

NOTA

Outro exemplo do perigo de ser demasiado confiante na modelação do mundo pode ser encontrado no livro do Dr. Steven Koonin, *Unsettled* (BenBella Books). Trabalhei com o Dr. Koonin quando ele estava na administração do Caltech e achei-o um cientista entusiasta e uma pessoa divertida com quem se pode ter uma conversa ao acaso. Aqui está uma citação do seu livro sobre modelação:

Uma vez que temos um conhecimento muito sólido das leis físicas que regem a matéria e a energia, é fácil deixarmo-nos seduzir pela ideia de que podemos simplesmente alimentar um computador com o estado atual da atmosfera e dos oceanos, fazer algumas suposições sobre futuras influências humanas e naturais e, assim, prever com precisão o clima para décadas futuras. Infelizmente, isso não passa de uma fantasia, como podes deduzir das previsões meteorológicas, que só podem ser exactas até duas semanas ou mais.

Otimização

Um problema fundamental na aprendizagem automática é o conceito de otimização. A otimização é a capacidade de encontrar a melhor solução, ou suficientemente boa, para um problema. A descida de gradiente é um algoritmo de otimização que está no centro da aprendizagem profunda. O objetivo da descida de gradiente é converter para o mínimo global, ou seja, a solução ideal, em vez de ficar preso em um mínimo local. A intuição por detrás do algoritmo é relativamente simples se imaginares que estás a descer uma montanha no escuro. A solução de mínimo global significa que sais da montanha vivo no fundo. O mínimo local significa que entraste accidentalmente num lago na encosta da montanha, a 1000 pés do fundo.

Vamos analisar um exemplo de problemas de otimização. Um excelente ponto de partida é observar os tipos de notação envolvidos na otimização. Ao criar um modelo, precisas de compreender o método Python de notação de expressões algébricas. O resumo rápido na [Figura 2-10](#) compara os termos entre uma folha de cálculo, álgebra e Python. Uma das principais conclusões é que podes fazer a mesma coisa num quadro branco, no Excel ou com um pouco de código.

Assume A_1, \dots, A_{10} are **parameters** and X_1, \dots, X_{10} are **decision variables**:

Spreadsheet	Algebra	Python
sum(X1:X6)	$\sum_{i=1}^6 X_i$	sum(range(1,6))
sum(A3:A7)	$\sum_{j=3}^7 A_j$	sum(range(3,7))
sumproduct(A1:A5,X1:X5)	$\sum_{i=1}^5 A_i X_i$	sum(reduce(operator.mul,data))
sumproduct(A3:A10,X3:X10)	$\sum_{j=3}^{10} A_j X_j$	sum(reduce(operator.mul,data))

Figura 2-10. A notação é relativa

Agora, vamos ver uma solução para fazer a alteração correta. Podes encontrar o código para esta solução [no GitHub](#). A ideia geral com este exemplo de código é escolher uma solução *gulosa* para fazer a alteração. Os algoritmos gulosos funcionam sempre escolhendo a melhor opção primeiro. Também funcionam bem se não te preocupares com a solução perfeita, ou se for impossível encontrar a solução perfeita, mas não te importas com uma solução "suficientemente boa". Neste caso, seria a moeda de valor mais elevado e farias o troco com ela, passando depois para a seguinte:

```
python change.py --full 1.34
```

```
Quarters 5: , Remainder: 9
Dimes 0: , Remainder: 0
Nickles 1: , Remainder: 4
Pennies 4:
```

Segue-se a secção principal do código que faz uma correspondência gulosa. Nota que uma função recursiva resolve cada iteração do problema, uma vez que as moedas grandes acabam por se esgotar. O algoritmo encontra a seguir moedas médias que se esgotam; finalmente, passa para as moedas mais pequenas:

```
def recursive_change(self, rem):
    """Greedy Coin Match with Recursion
    >>> c = Change(.71)
    >>> c.recursive_change(c.convert)
    2 quarters
    2 dimes
    1 pennies
    [1, 0, 2, 2]
```

```

"""
if len(self.coins) == 0:
    return []
coin = self.coins.pop()
num, new_rem = divmod(rem, coin)
self.printer(num, coin)
return self.recursive_change(new_rem) + [num]

```

Embora existam muitas formas diferentes de expressar o conceito algorítmicamente, a ideia é a mesma. Sem saberes como encontrar a solução "perfeita", a resposta adequada é escolher sempre a melhor opção quando te é apresentada. Intuitivamente, isto é como caminhar para um destino na diagonal de uma cidade e seguir em frente ou virar à direita sempre que o semáforo à tua frente fica vermelho.

Aqui tens uma série de testes para este algoritmo. Mostram o desempenho do algoritmo, o que é frequentemente uma boa ideia quando se testa uma solução que envolve otimização:

```

#!/usr/bin/env python2.5
#Noah Gift
#Greedy Coin Match Python

import unittest
import change

class TestChange(unittest.TestCase):
    def test_get_quarter(self):
        c = change.Change(.25)
        quarter, qrem, dime, drem, nickel, nrem, penny =\
            c.make_change_conditional()
        self.assertEqual(quarter, 1) #quarters
        self.assertEqual(qrem, 0) #quarter remainder
    def test_get_dime(self):
        c = change.Change(.20)
        quarter, qrem, dime, drem, nickel, nrem, penny =\
            c.make_change_conditional()
        self.assertEqual(quarter, 0) #quarters
        self.assertEqual(qrem, 20) #quarter remainder
        self.assertEqual(dime, 2) #dime
        self.assertEqual(drem, 0) #dime remainder
    def test_get_nickel(self):
        c = change.Change(.05)

```

```

quarter, qrem, dime, drem, nickel, nrem, penny =\
    c.make_change_conditional()
self.assertEqual(dime, 0)      #dime
self.assertEqual(drem, 0)      #dime remainder
self.assertEqual(nickel, 1)    #nickel
self.assertEqual(nrem, 0)      #nickel remainder
def test_get_penny(self):
    c = change.Change(.04)
    quarter, qrem, dime, drem, nickel, nrem, penny =\
        c.make_change_conditional()
    self.assertEqual(penny, 4)  #nickel
def test_small_number(self):
    c = change.Change(.0001)
    quarter, qrem, dime, drem, nickel, nrem, penny =\
        c.make_change_conditional()
    self.assertEqual(quarter, 0) #quarters
    self.assertEqual(qrem, 0)   #quarter remainder
    self.assertEqual(dime, 0)   #dime
    self.assertEqual(drem, 0)   #dime remainder
    self.assertEqual(nickel, 0) #nickel
    self.assertEqual(nrem, 0)   #nickel remainder
    self.assertEqual(penny, 0)  #penny
def test_large_number(self):
    c = change.Change(2.20)
    quarter, qrem, dime, drem, nickel, nrem, penny =\
        c.make_change_conditional()
    self.assertEqual(quarter, 8) #nickel
    self.assertEqual(qrem, 20)  #nickel
    self.assertEqual(dime, 2)   #nickel
    self.assertEqual(drem, 0)   #nickel
def test_get_quarter_dime_penny(self):
    c = change.Change(.86)
    quarter, qrem, dime, drem, nickel, nrem, penny =\
        c.make_change_conditional()
    self.assertEqual(quarter, 3) #quarters
    self.assertEqual(qrem, 11)   #quarter remainder
    self.assertEqual(dime, 1)    #dime
    self.assertEqual(drem, 1)    #dime remainder
    self.assertEqual(penny, 1)   #penny
def test_get_quarter_dime_nickel_penny(self):
    c = change.Change(.91)
    quarter, qrem, dime, drem, nickel, nrem, penny =\
        c.make_change_conditional()
    self.assertEqual(quarter, 3) #quarters
    self.assertEqual(qrem, 16)   #quarter remainder
    self.assertEqual(dime, 1)    #dime
    self.assertEqual(drem, 6)    #dime remainder
    self.assertEqual(nickel, 1)  #nickel

```

```

    self.assertEqual(nrem, 1)      #nickel remainder
    self.assertEqual(penny, 1)      #penny

if __name__ == "__main__":
    unittest.main()

```

De seguida, vamos utilizar algoritmos gulosos no seguinte problema. Um dos problemas mais estudados em otimização é o problema do caixeiro-viajante. Podes encontrar o código fonte [no GitHub](#). Este exemplo é uma lista de rotas que estão num ficheiro *routes.py*. Mostra a distância entre diferentes empresas na Bay Area.

É um excelente exemplo de uma solução perfeita que não existe, mas uma solução suficientemente boa existe. O problema geral coloca a seguinte questão: "Como podes viajar para uma lista de cidades e minimizar a distância?"

Uma forma de o fazer é utilizar um algoritmo "guloso". Escolhe a solução certa em cada escolha. Muitas vezes, isto pode levar a uma resposta suficientemente boa. Neste exemplo em particular, escolhe sempre uma cidade "aleatória" como ponto de partida. Este exemplo acrescenta a possibilidade de as simulações escolherem a distância mais baixa. Um utilizador das simulações pode simular tantas vezes quantas as que tiver tempo. O menor comprimento total é a melhor resposta. A seguir, uma amostra de como é a entrada antes de ser processada noalgoritmo TSP:

```

values = [
("AAPL", "CSCO", 14),
("AAPL", "CVX", 44),
("AAPL", "EBAY", 14),
("AAPL", "GOOG", 14),
("AAPL", "GPS", 59),
("AAPL", "HPQ", 14),
("AAPL", "INTC", 8),
("AAPL", "MCK", 60),
("AAPL", "ORCL", 26),
("AAPL", "PCG", 59),
("AAPL", "SFO", 46),
("AAPL", "SWY", 37),
("AAPL", "URS", 60),
("AAPL", "WFC", 60),
]

```

Vamos executar o script. Primeiro, repara que recebe como entrada as simulações completas a executar:

```
#!/usr/bin/env python
"""
Traveling salesman solution with random start and greedy path
selection
You can select how many iterations to run by doing the following:

python greedy_random_start.py 20 #runs 20 times

"""

import sys
from random import choice
import numpy as np
from routes import values

dt = np.dtype([("city_start", "S10"), ("city_end", "S10"),
              ("distance", int)])
data_set = np.array(values, dtype=dt)

def all_cities():
    """Finds unique cities

    array([[ "A", "A"],
           [ "A", "B"]])

    """
    cities = {}
    city_set = set(data_set["city_end"])
    for city in city_set:
        cities[city] = ""
    return cities

def randomize_city_start(cities):
    """Returns a randomized city to start trip"""

    return choice(cities)

def get_shortest_route(routes):
    """Sort the list by distance and return shortest distance
```

```

route"""

    route = sorted(routes, key=lambda dist: dist[2]).pop(0)
    return route


def greedy_path():
    """Select the next path to travel based on the shortest,
    nearest path"""

    itinerary = []
    cities = all_cities()
    starting_city = randomize_city_start(list(cities.keys()))
    # print "starting_city: %s" % starting_city
    cities_visited = {}
    # we want to iterate through all cities once
    count = 1
    while True:
        possible_routes = []
        # print "starting city: %s" % starting_city
        for path in data_set:
            if starting_city in path["city_start"]:
                # we can't go to cities we have already visited
                if path["city_end"] in cities_visited:
                    continue
                else:
                    # print "path: ", path
                    possible_routes.append(path)

        if not possible_routes:
            break
        # append this to itinerary
        route = get_shortest_route(possible_routes)
        # print "Route(%s): %s " % (count, route)
        count += 1
        itinerary.append(route)
        # add this city to the visited city list
        cities_visited[route[0]] = count
        # print "cities_visited: %s " % cities_visited
        # reset the starting_city to the next city
        starting_city = route[1]
        # print "itinerary: %s" % itinerary

    return itinerary


def get_total_distance(complete_itinerary):

```

```

        distance = sum(z for x, y, z in complete_itinerary)
        return distance

def lowest_simulation(num):

    routes = {}
    for _ in range(num):
        itinerary = greedy_path()
        distance = get_total_distance(itinerary)
        routes[distance] = itinerary
    shortest_distance = min(routes.keys())
    route = routes[shortest_distance]
    return shortest_distance, route

def main():
    """runs everything"""

    if len(sys.argv) == 2:
        iterations = int(sys.argv[1])
        print("Running simulation %s times" % iterations)
        distance, route = lowest_simulation(iterations)
        print("Shortest Distance: %s" % distance)
        print("Optimal Route: %s" % route)
    else:
        # print "All Routes: %s" % data_set
        itinerary = greedy_path()
        print("itinerary: %s" % itinerary)
        print("Distance: %s" % get_total_distance(itinerary))

if __name__ == "__main__":
    main()

```

Vamos executar este algoritmo "ganancioso" 25 vezes. Repara que encontra uma "boa" solução de 129. Esta versão pode ou não ser a solução óptima num conjunto mais extenso de coordenadas, mas é suficientemente boa para iniciar uma viagem de carro para os nossos propósitos:

```

> ./greedy-random-tsp.py 25
Running simulation 25 times
Shortest Distance: 129
Optimal Route: [(b'WFC', b'URS', 0), (b'URS', b'GPS', 1), \
(b'GPS', b'PCG', 1), (b'PCG', b'MCK', 3), (b'MCK', b'SFO', 16), \

```

```
(b'SFO', b'ORCL', 20), (b'ORCL', b'HPQ', 12), (b'HPQ', b'GOOG',
6), \
(b'GOOG', b'AAPL', 11), (b'AAPL', b'INTC', 8), (b'INTC', b'CSCO',
6), \
(b'CSCO', b'EBAY', 0), (b'EBAY', b'SWY', 32), (b'SWY', b'CVX',
13)]
```

Repara que se eu executar a simulação apenas uma vez, ela seleciona aleatoriamente uma distância piorde 143:

```
> ./greedy-random-tsp.py 1
Running simulation 1 times
Shortest Distance: 143
Optimal Route: [(b'CSCO', b'EBAY', 0), (b'EBAY', b'INTC', 6), \
(b'INTC', b'AAPL', 8), (b'AAPL', b'GOOG', 14), (b'GOOG', b'HPQ',
6), \
(b'HPQ', b'ORCL', 12), (b'ORCL', b'SFO', 20), (b'SFO', b'MCK',
16), \
(b'MCK', b'WFC', 2), (b'WFC', b'URS', 0), (b'URS', b'GPS', 1), \
(b'GPS', b'PCG', 1), (b'PCG', b'CVX', 44), (b'CVX', b'SWY', 13)]
```

Repara na [Figura 2-11](#) como eu executaria várias iterações do código num cenário do mundo real para "experimentar ideias". Se o conjunto de dados fosse enorme e eu estivesse com pressa, poderia fazer apenas algumas simulações, mas se estivesse a sair durante o dia, poderia deixá-lo correr 1.000 vezes e terminar quando voltasse de manhã no dia seguinte. Pode haver muitos mínimos locais num conjunto de dados de coordenadas geográficas - isto é, soluções para o problema que não atingem o mínimo global, ou a solução óptima.

```
or ~ noahgift@M1-Replica ~ /src/or ~ zsh ~ 117x27
(.or) ~ or git:(master) ✘ ./greedy-random-tsp.py 25
Running simulation 25 times
Shortest Distance: 129
Optimal Route: [(b'WFC', b'URS', 0), (b'URS', b'GPS', 1), (b'GPS', b'PCG', 1), (b'PCG', b'MCK', 3), (b'MCK', b'SFO',
16), (b'SFO', b'ORCL', 20), (b'ORCL', b'HPQ', 12), (b'HPQ', b'GOOG', 6), (b'GOOG', b'AAPL', 11), (b'AAPL', b'INTC',
8), (b'INTC', b'CSCO', 6), (b'CSCO', b'EBAY', 0), (b'EBAY', b'SWY', 32), (b'SWY', b'CVX', 13)]
(.or) ~ or git:(master) ✘ ./greedy-random-tsp.py 1
Running simulation 1 times
Shortest Distance: 143
Optimal Route: [(b'CSCO', b'EBAY', 0), (b'EBAY', b'INTC', 6), (b'INTC', b'AAPL', 8), (b'AAPL', b'GOOG', 14), (b'GOOG',
6), (b'HPQ', b'ORCL', 12), (b'ORCL', b'SFO', 20), (b'SFO', b'MCK', 16), (b'MCK', b'WFC', 2), (b'WFC', b'URS',
0), (b'URS', b'GPS', 1), (b'GPS', b'PCG', 1), (b'PCG', b'CVX', 44), (b'CVX', b'SWY', 13)]
```

Figura 2-11. Simulação TSP

A otimização faz parte das nossas vidas e utilizamos algoritmos gulosos para resolver problemas do dia a dia porque são intuitivos. A otimização também está no centro da forma como a aprendizagem automática funciona utilizando a descida de gradiente. Um problema de aprendizado de máquina caminha iterativamente em direção a um mínimo local ou global usando o algoritmo de descida de gradiente, como mostrado na [Figura 2-12](#).

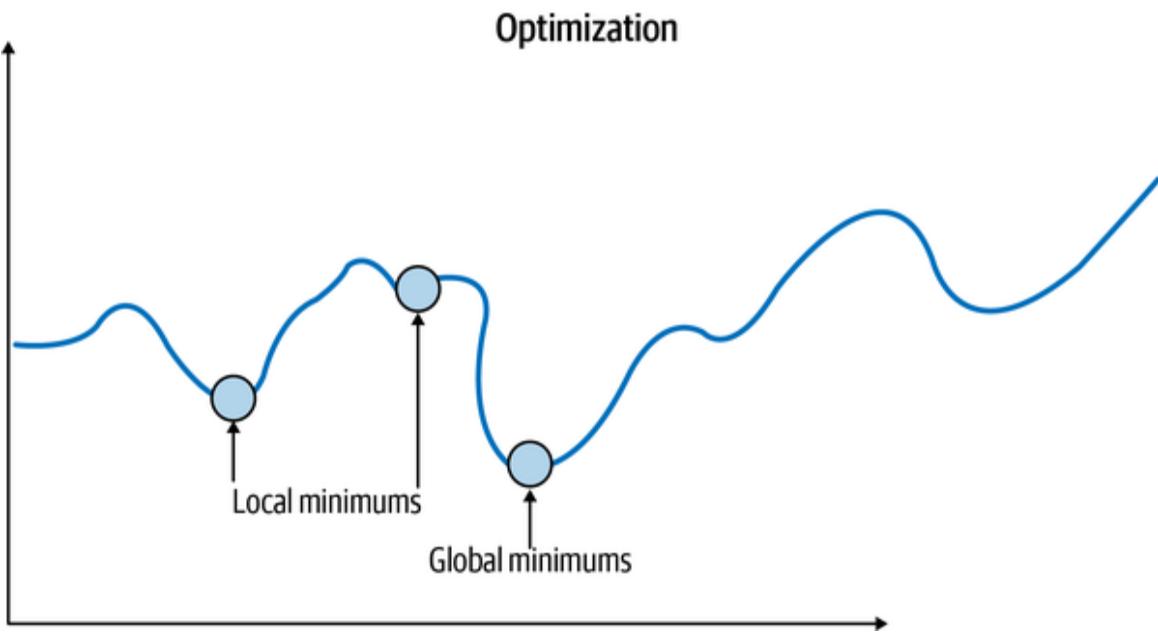


Figura 2-12. Otimização

NOTAS SOBRE A INTUIÇÃO DA DEEP LEARNING

Para os MLOps, a convergência , ou seja, a criação de um modelo que encontra uma solução que não será melhorada com a adição de mais dados, é uma questão operacional essencial. Por exemplo, será que um cluster de formação baseado em GPU permite uma convergência mais rápida? Um cluster de treinamento baseado em CPU ofereceria custos mais baixos? Os custos operacionais podem quebrar uma empresa ou um projeto no mundo real, e é essencial ter uma intuição de como a otimização funciona e testá-la na prática.

Uma ferramenta valiosa para melhorar a intuição sobre a descida de gradiente é o [Playground do TensorFlow](#). Em particular, a experimentação com a taxa de aprendizado mostra como uma taxa muito alta pode levar à oscilação, como mostrado na [Figura 2-13](#).

Observa como a perda de teste se mantém em 0,984 porque a taxa de aprendizagem é muito alta para usar o algoritmo de descida de gradiente de forma eficaz. Da mesma forma, se definir a taxa de aprendizagem demasiado baixa, pode não atingir o mínimo global ou demorar demasiado tempo a convergir para a solução correta.

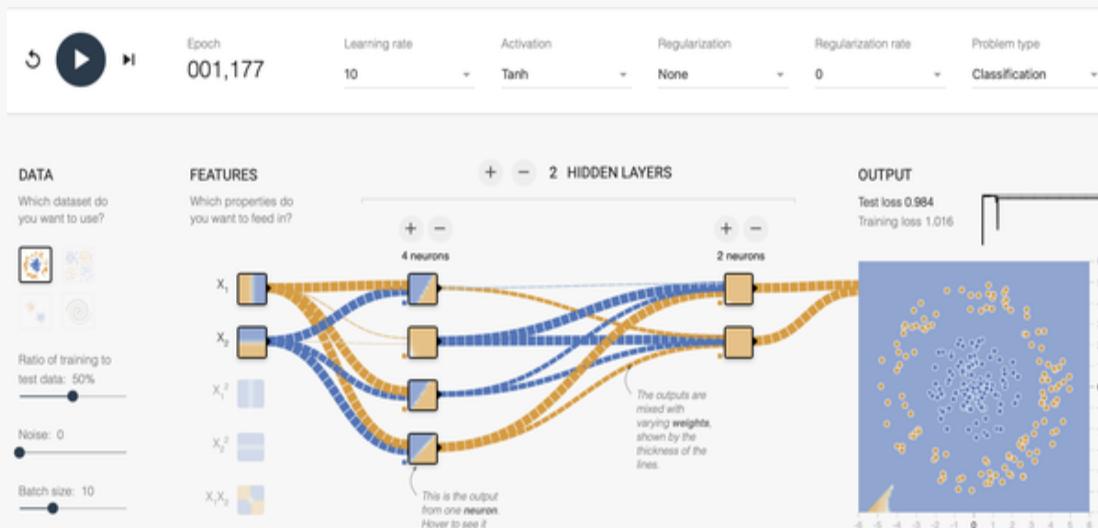


Figura 2-13. Taxa de aprendizagem demasiado elevada

Matematicamente, este conjunto de soluções de compromisso aparece na [Figura 2-14](#). A taxa de aprendizagem ideal converge para o mínimo

global, mas uma taxa muito alta leva a um colapso, como mostrado no exemplo do Playground do TensorFlow. Em alternativa, uma taxa demasiado baixa pode levar a que fiques preso num mínimo local ou a que demores demasiado tempo a convergir.

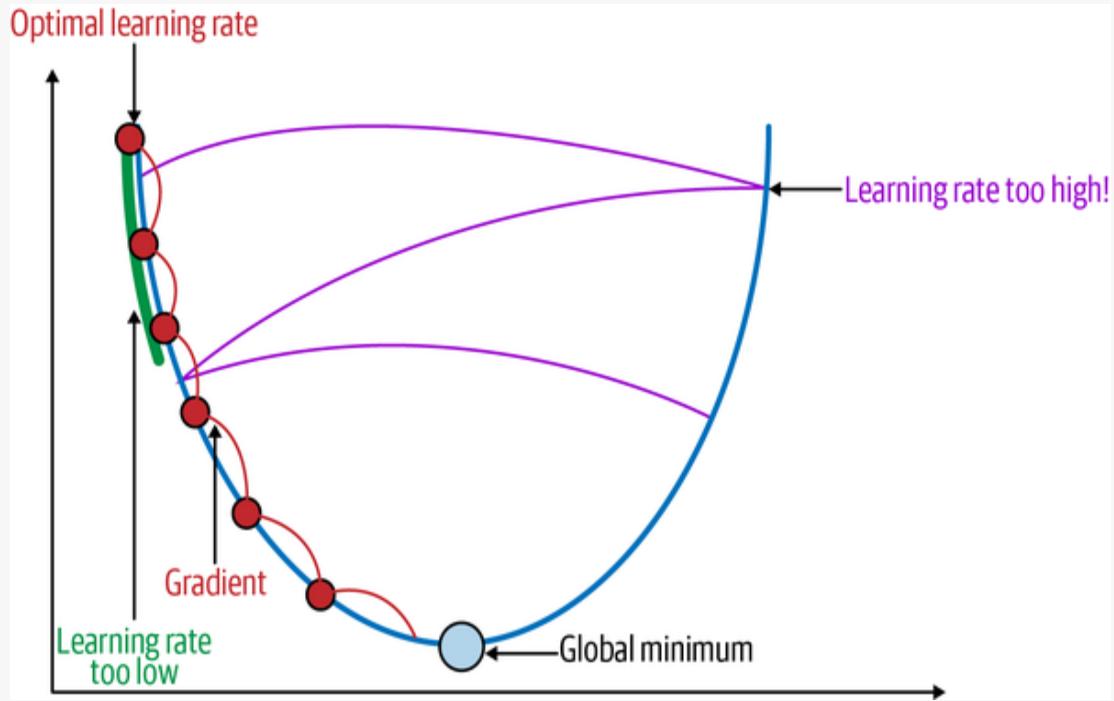


Figura 2-14. Intuição da taxa de aprendizagem

Em seguida, vamos mergulhar nos conceitos fundamentais da aprendizagem automática.

Conceitos-chave da aprendizagem automática

A aprendizagem automática é a capacidade de os computadores realizarem tarefas sem programação explícita. Fazem-no "aprendendo" com os dados. Como já foi referido, uma boa intuição seria um modelo de aprendizagem automática que pudesse prever o peso com base na altura. Poderia "aprender" com 25.000 observações e depois fazer uma previsão.

A aprendizagem automática envolve três categorias: supervisionada, não supervisionada e aprendizagem por reforço. A aprendizagem automática supervisionada ocorre quando as "etiquetas" são conhecidas e o modelo aprende a partir de dados históricos. No exemplo anterior, a altura e o peso são etiquetas. Além disso, as 25.000 observações são um exemplo de dados históricos. Repara que toda a aprendizagem automática exige que os dados estejam num formato numérico e requer escalonamento. Imagina que um amigo se gaba de ter corrido 50. O que é que ele queria dizer? O que é que ele quis dizer? Foram 50 milhas ou 50 pés? A magnitude é a razão para escalar os dados antes de processar uma previsão.

A aprendizagem automática não supervisionada funciona para "descobrir" etiquetas. Uma boa intuição de como isso funciona é considerar uma temporada da NBA. Na visualização mostrada na [Figura 2-15](#) da temporada 2015-2016 da NBA, o computador "aprendeu" como agrupar os diferentes jogadores da NBA. Cabe ao especialista no domínio, neste caso, eu, selecionar as etiquetas adequadas. O algoritmo foi capaz de agrupar grupos, um dos quais eu rotulei como os "melhores" jogadores.

NOTA

Como especialista no domínio do basquetebol, acrescentei então um rótulo chamado "melhor". No entanto, outro perito no domínio pode discordar e chamar a estes jogadores "elite bem-feita" ou qualquer outro rótulo. O agrupamento é uma arte e uma ciência. Ter um perito no domínio que compreenda as soluções de compromisso sobre o que rotular num conjunto de dados agrupados pode ser decisivo para a utilidade da previsão de aprendizagem automática não supervisionada.



Figura 2-15. Agrupamento K-means agrupado e facetado dos jogadores da NBA

O computador agrupou os dados com base numa comparação de quatro atributos: pontos, ressaltos, bloqueios e assistências. Depois, no espaço multidimensional, os jogadores com a menor distância total entre si foram agrupados para formar uma etiqueta. Esse algoritmo de agrupamento é a razão pela qual LeBron James e Kevin Durant se agrupam; eles têm métricas semelhantes. Além disso, Steph Curry e Chris Paul são parecidos, pois marcam muitos pontos e dão muitas assistências.

NOTA

Um dilema comum com o agrupamento K-means é como selecionar o número correto de agrupamentos. Essa parte do problema também é um problema de arte e ciência, pois não há necessariamente uma resposta perfeita. Uma solução é usar uma estrutura para criar gráficos de cotovelo para ti, como [Yellowbrick](#) para sklearn.

Outra solução no estilo MLOps é deixar que a plataforma MLOps, como o AWS Sagemaker, faça a atribuição do cluster K-means por meio do [ajuste automatizado de hiperparâmetros](#).

Finalmente, com a aprendizagem por reforço, um "agente" explora um ambiente para aprender a executar tarefas. Considera, por exemplo, um animal de estimação ou uma criança pequena. Eles sabem como interagir com o mundo explorando o seu ambiente. Um exemplo mais concreto é o sistema AWS DeepRacer que te permite treinar um modelo de carro para conduzir numa pista, como mostrado na [Figura 2-16](#).

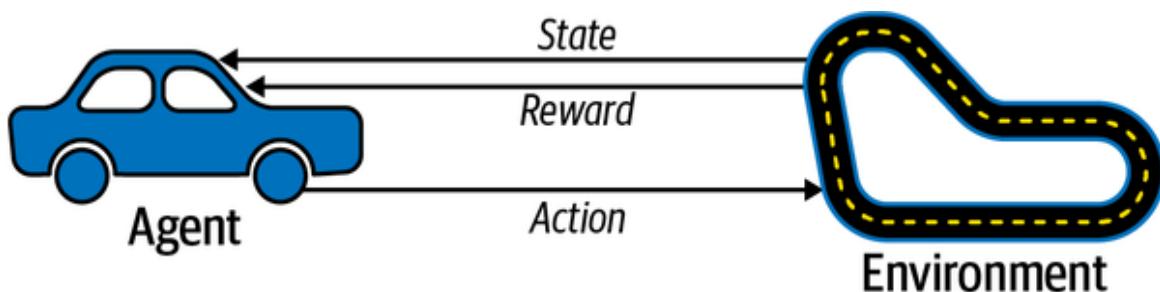


Figura 2-16. AWS DeepRacer

O agente, que é o carro, interage com a pista, que é o ambiente. O veículo move-se através de cada secção da pista e a plataforma armazena dados sobre a sua posição na pista. Uma função de recompensa decide como o agente interage em cada passagem pela pista. A aleatoriedade desempenha um papel importante no treino deste tipo de modelo, pelo que diferentes estratégias de função de recompensa podem produzir resultados diferentes.

Segue-se um exemplo de uma função de recompensa em Python para o AWS DeepRacer que recompensa seguir a linha central:

```
def reward_function(params):  
    '''
```

Example of rewarding the agent for following the centerline

```

    '''

# Read input parameters
track_width = params['track_width']
distance_from_center = params['distance_from_center']

# Calculate 3 markers that are at varying distances away from
the centerline
marker_1 = 0.1 * track_width
marker_2 = 0.25 * track_width
marker_3 = 0.5 * track_width

# Give higher reward if the car is closer to centerline and
vice versa
if distance_from_center <= marker_1:
    reward = 1.0
elif distance_from_center <= marker_2:
    reward = 0.5
elif distance_from_center <= marker_3:
    reward = 0.1
else:
    reward = 1e-3 # likely crashed/ close to off track

return float(reward)

```

Aqui tens uma função de recompensa diferente que recompensa o agente por se manter dentro dos dois limites da pista. Esta abordagem é semelhante à função de recompensa anterior, mas pode produzir resultados dramaticamente diferentes:

```

def reward_function(params):
    '''

Example of rewarding the agent for staying inside the two
borders of the
track
'''

# Read input parameters
all_wheels_on_track = params['all_wheels_on_track']
distance_from_center = params['distance_from_center']
track_width = params['track_width']

# Give a very low reward by default
reward = 1e-3

```

```

# Give a high reward if no wheels go off the track and
# the agent is somewhere in between the track borders
if all_wheels_on_track and (0.5*track_width -
distance_from_center) >= 0.05:
    reward = 1.0

# Always return a float value
return float(reward)

```

A utilização da aprendizagem automática na produção requer os conhecimentos fundamentais abordados neste capítulo, ou seja, saber qual a abordagem a utilizar. Por exemplo, a descoberta de rótulos através da aprendizagem automática não supervisionada pode ser valiosa para determinar quem são os clientes que pagam melhor. Do mesmo modo, a previsão do número de unidades que serão vendidas no próximo trimestre pode ser efectuada através de uma abordagem de aprendizagem automática supervisionada que utiliza os dados históricos e cria uma previsão. A seguir, vamos mergulhar nos conceitos básicos da ciência de dados.

Fazer ciência de dados

Outra competência fundamental a dominar é "a forma da ciência dos dados". Nas aulas que lecciono, recomendo que crie a seguinte estrutura de fórmulas num bloco de notas: *Ingest*, *EDA*, *Modelação* e *Conclusão*. Esta estrutura permite que qualquer pessoa de uma equipa passe rapidamente pelas diferentes secções do projeto para ter uma ideia do que se trata. Para além disso, para um modelo implementado em produção, é benéfico ter um bloco de notas junto com o código que implementa o modelo para servir como um *README* para o pensamento por detrás do projeto. Podes ver um exemplo disto na [Figura 2-17](#).



Figura 2-17. Bloco de notas do Colab

Podes ver um exemplo desta estrutura no [caderno Colab Data Science with Covid](#).

Esta divisão clara das partes do bloco de notas significa que cada secção pode ser um "capítulo" na escrita de um livro de ciência de dados. A secção Ingest beneficia de fontes de dados que são carregadas através de um pedido Web, ou seja, alimentam diretamente o Pandas. As fontes de dados do bloco de notas podem ser replicadas por outros usando esta abordagem, como mostra a Figura 2-18.

```

+ Code + Text
Data Science With Covid
Ingest
COVID-19 Data from New York Times Github

[1]: 1 import pandas as pd
      2 df = pd.read_csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv")
      3 df.head()

      date      state    fips  cases  deaths
0 2020-01-21 Washington  53       1       0
1 2020-01-22 Washington  53       1       0
2 2020-01-23 Washington  53       1       0
3 2020-01-24 Illinois     17       1       0
4 2020-01-24 Washington  53       1       0

```

Figura 2-18. Estrutura do bloco de notas do Colab

A secção EDA destina-se à exploração de ideias. O que é que se passa com os dados? Esta é a oportunidade para descobrires, como mostram os principais estados com Covid no gráfico seguinte, utilizando o Plotly na Figura 2-19.

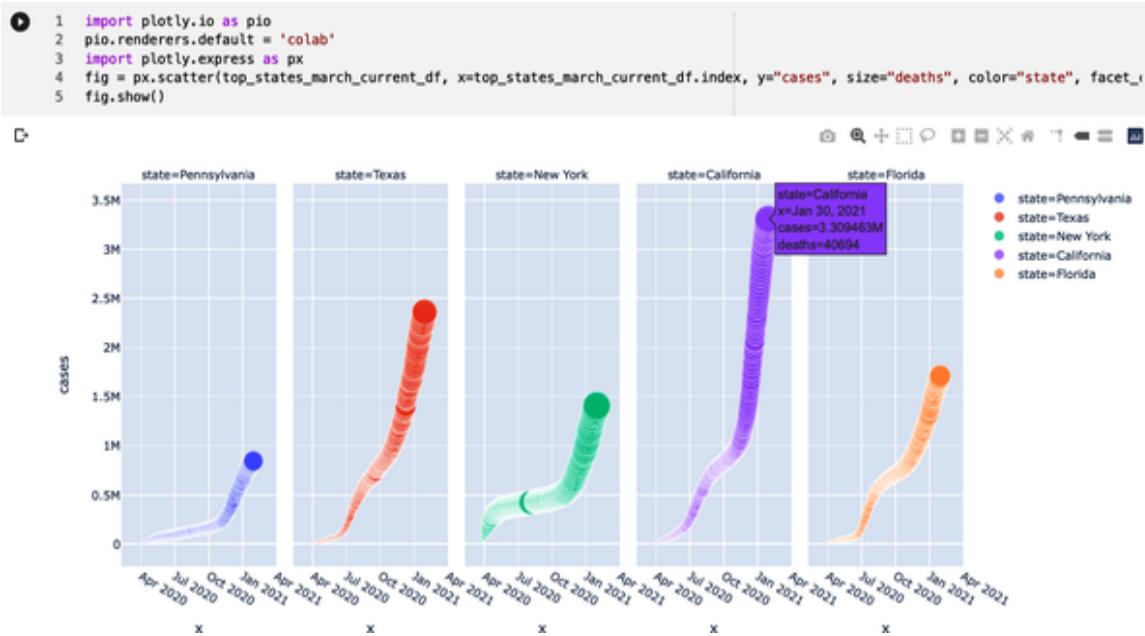


Figura 2-19. EDA do bloco de notas Colab

A secção Modeling é onde o modelo reside. Mais tarde, esta repetibilidade pode ser crítica, uma vez que o pipeline MLOps pode precisar de fazer referência à forma como a criação do modelo ocorreu. Por exemplo, podes ver um excelente exemplo de um modelo sklearn serializado neste [notebook do Boston Housing Pickle Colab](#). Observa que eu testo como esse modelo funcionará eventualmente em uma API ou em um sistema baseado em Cloud, como este [projeto de implantação do Flask ML](#).

A secção Conclusão deve ser um resumo para um líder empresarial que toma a decisão. Finalmente, verifica o teu projeto no GitHub para construir o teu portfólio MLOps. O rigor na adição desta documentação adicional compensa à medida que um projeto de ML amadurece. As equipas de operações, em particular, podem considerar muito valioso compreender o raciocínio original sobre o motivo pelo qual um modelo está em produção e decidir retirar o modelo da produção, uma vez que já não faz sentido.

Em seguida, vamos discutir a construção de um pipeline MLOps passo a passo.

Cria um pipeline de MLOps a partir do zero

Vamos juntar tudo neste capítulo e mergulhar no Implantar aplicativo de aprendizado de máquina Flask nos Serviços de aplicativos do Azure.

Observa na [Figura 2-20](#) que os eventos do GitHub acionam uma compilação do processo de compilação do Azure Pipelines, que então implanta as alterações em uma plataforma sem servidor. Os nomes são diferentes em outras plataformas Cloud, mas conceitualmente as coisas são muito semelhantes tanto na AWS quanto no GCP.

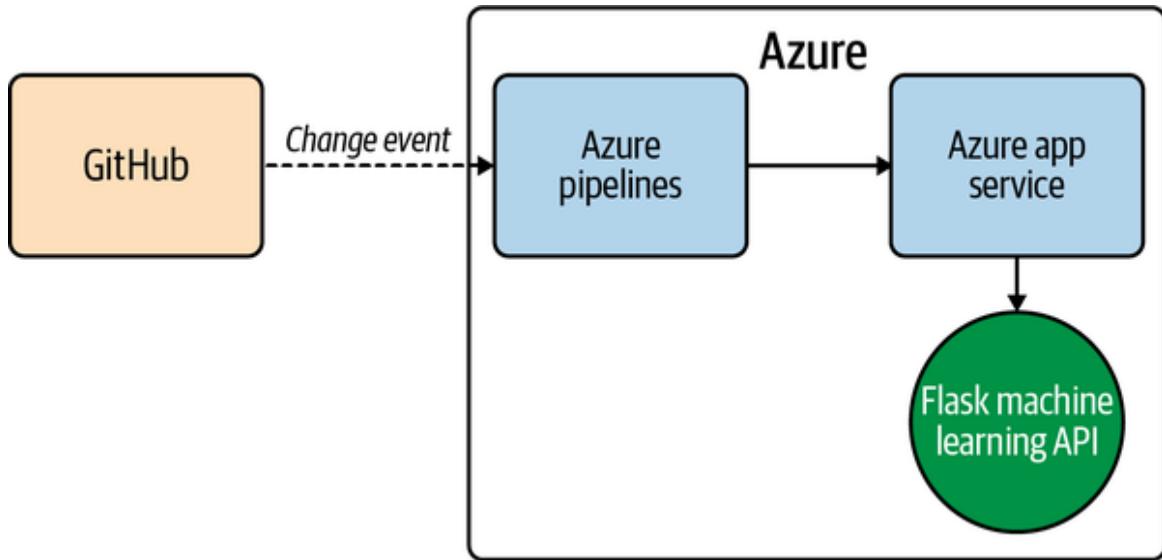


Figura 2-20. Visão geral do MLOps

Para o executar localmente, segue estes passos:

1. Cria um ambiente virtual e uma fonte:

```

python3 -m venv ~/.flask-ml-azure
source ~/.flask-ml-azure/bin/activate

```

2. Executa `make install`.

3. Executa `python app.py`.

4. Numa shell separada, executa `./make_prediction.sh`.

Para o executares no Azure Pipelines (consulta o [guia da Documentação Oficial do Azure](#)):

1. Lança o Azure Shell como mostrado na [Figura 2-21](#).



Figura 2-21. Inicia o Azure Cloud Shell

2. Cria um repositório do GitHub com o Azure Pipelines ativado (que pode ser uma bifurcação deste repositório), conforme apresentado na [Figura 2-22](#).

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

 noahgift / flask-ml-azure-serverless ✓

Great repository names are short and memorable. Need inspiration? How about [fictional-doodle](#)?

Description (optional)

Deploy Flask Machine Learning Application on Azure App Services

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: Python ▾ Add a license: None ▾ ⓘ

Grant your Marketplace apps access to this repository

You are subscribed to 2 Marketplace apps

 **Azure Pipelines**
Continuously build, test, and deploy to any platform and cloud

 **Google Cloud Build**
Build, test, and deploy in a fast, consistent, and secure manner

Create repository

Figura 2-22. Cria um repositório do GitHub com o Azure Pipelines

3. Clona o repo no Azure Cloud Shell.

NOTA

Se precisares de mais informações sobre como configurar as chaves SSH, podes seguir este guia em vídeo do YouTube sobre como [configurar as chaves SSH e configurar o ambiente do Cloud Shell](#).

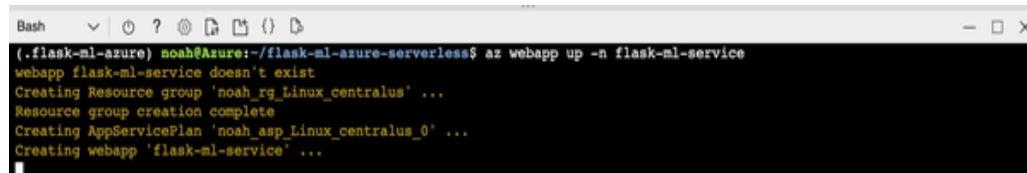
4. Cria um ambiente virtual e uma fonte:

```
python3 -m venv ~/.flask-ml-azure  
source ~/.flask-ml-azure/bin/activate
```

5. Executa `make install`.

6. Cria um serviço de aplicativo e implanta inicialmente seu aplicativo no Cloud Shell, como mostrado na [Figura 2-23](#).

```
az webapp up -n <your-appservice>
```



```
Bash .flask-ml-azure:~/.flask-ml-azure-serverless$ az webapp up -n flask-ml-service  
webapp flask-ml-service doesn't exist  
Creating Resource group 'noah_rg_Linux_centralus' ...  
Resource group creation complete  
Creating AppServicePlan 'noah_asp_Linux_centralus_0' ...  
Creating webapp 'flask-ml-service' ...
```

Figura 2-23. Serviço Flask ML

7. Verifica se a aplicação implementada funciona navegando para o url implementado: <https://<your-appservice>.azurewebsites.net/>.

Verás o resultado como mostrado na [Figura 2-24](#).



Figura 2-24. Aplicativo implantado em Flask

8. Verifica se as previsões de aprendizagem automática funcionam, como mostra a [Figura 2-25](#).

Altera a linha em `make_predict_azure_app.sh` para corresponder à previsão implementada -X POST [https://<yourappname>.azurewebsites.net:\\$PORT/predict](https://<yourappname>.azurewebsites.net:$PORT/predict).

Figura 2-25. Previsão bem sucedida

9. Cria um projeto Azure DevOps e liga-te ao Azure (como a documentação oficial descreve), como mostra a Figura 2-26.

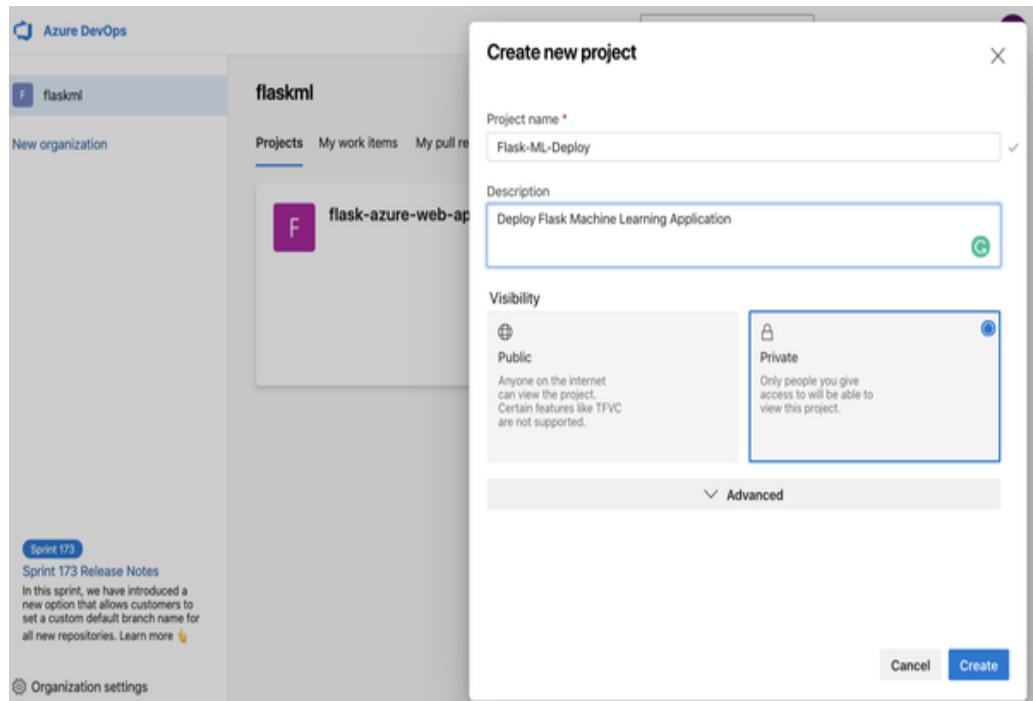


Figura 2-26. Conexão do Azure DevOps

10. Liga-te ao Azure Resource Manager, conforme ilustrado na [Figura 2-27](#).

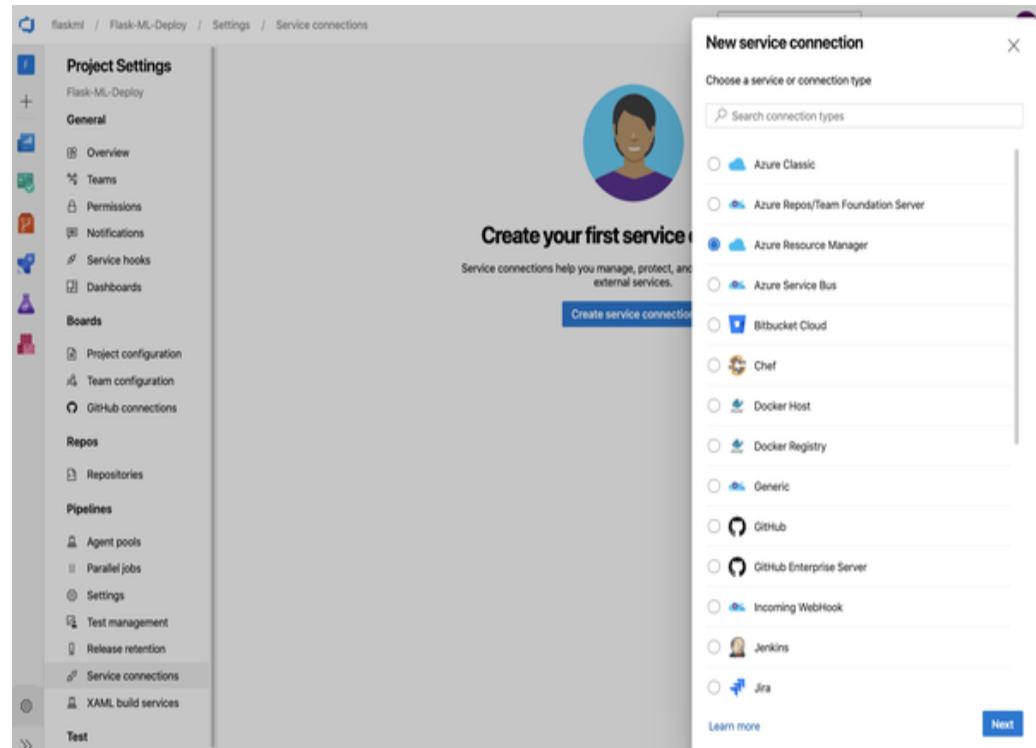


Figura 2-27. Conector de serviço

11. Configura a conexão com o grupo de recursos implantado anteriormente, conforme mostrado na **Figura 2-28**.

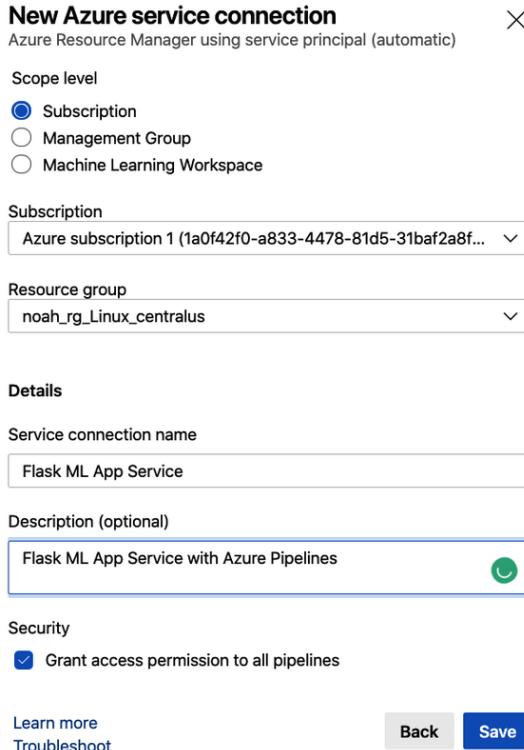


Figura 2-28. Nova ligação de serviço

12. Cria um novo pipeline Python com integração GitHub, como mostrado na **Figura 2-29**.

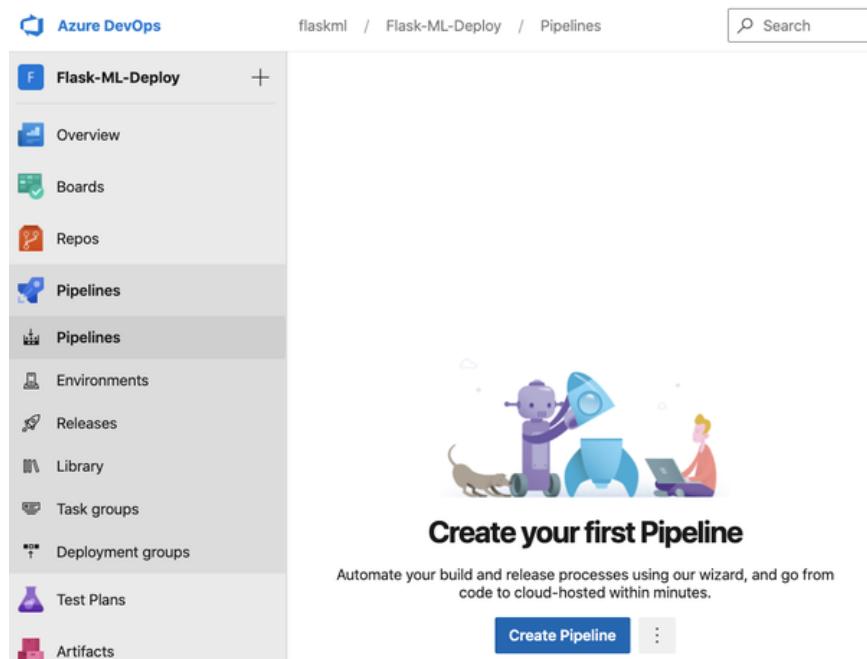


Figura 2-29. Nova tubagem

Por fim, configura a integração do GitHub como mostrado na Figura 2-30.

The screenshot shows the Azure DevOps interface for creating a new pipeline. The left sidebar lists project navigation options like Overview, Boards, Repos, Pipelines, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area is titled 'Where is your code?' and lists several integration sources: 'Azure Repos Git' (YAML), 'Bitbucket Cloud' (YAML), 'GitHub' (YAML), 'GitHub Enterprise Server' (YAML), 'Other Git', and 'Subversion'. A note at the bottom says 'Use the classic editor to create a pipeline without YAML.'

Figura 2-30. Integração com o GitHub

Esse processo criará um arquivo YAML que se parece mais ou menos com a saída YAML mostrada no código a seguir. Consulta a [documentação oficial do Azure Pipeline YAML](#) para obteres mais informações sobre o mesmo. Esta é a primeira parte do arquivo gerado pelo computador:

```
# Python to Linux Web App on Azure
# Build your Python project and deploy it to Azure as a
# Linux Web App.
# Change python version to one that's appropriate for your
# application.
#
https://docs.microsoft.com/azure/devops/pipelines/languages/python

trigger:
- master
```

```
variables:
    # Azure Resource Manager connection created during
    pipeline creation
    azureServiceConnectionId: 'df9170e4-12ed-498f-93e9-
    79c1e9b9bd59'

    # Web app name
    webAppName: 'flask-ml-service'

    # Agent VM image name
    vmImageName: 'ubuntu-latest'

    # Environment name
    environmentName: 'flask-ml-service'

    # Project root folder. Point to the folder containing
    manage.py file.
    projectRoot: $(System.DefaultWorkingDirectory)

    # Python version: 3.7
    pythonVersion: '3.7'

stages:
- stage: Build
    displayName: Build stage
    jobs:
        - job: BuildJob
            pool:
                vmImage: $(vmImageName)
            steps:
                - task: UsePythonVersion@0
                    inputs:
                        versionSpec: '$(pythonVersion)'
                    displayName: 'Use Python $(pythonVersion)'
```

```
- script: |
    python -m venv antenv
    source antenv/bin/activate
    python -m pip install --upgrade pip
    pip install setup
    pip install -r requirements.txt
workingDirectory: $(projectRoot)
```

13. Verifica a entrega contínua do Azure Pipelines alterando o *app.py*.

Podes ver este [passo-a-passo do YouTube sobre este processo](#).

14. Adiciona um passo de lint (isto protege o teu código contra falhas de sintaxe):

```
- script: |
    python -m venv antenv
    source antenv/bin/activate
    make install
    make lint
workingDirectory: $(projectRoot)
displayName: 'Run lint tests'
```

NOTA

Para obteres uma descrição completa do código, podes ver a seguinte [descrição no YouTube deste processo de implementação do MLOps](#).

Conclusão

Este capítulo tem como objetivo dar-te os conhecimentos básicos necessários para implementar a aprendizagem automática na produção, ou seja, MLOps. Um dos desafios do MLOps é a multidisciplinaridade do campo. Quando se lida com algo inherentemente complexo, uma boa

abordagem começa com algo pequeno e faz com que a solução mais básica funcione, depois repete a partir daí.

Também é essencial estar ciente das competências fundamentais para uma organização que deseja fazer MLOps. Em particular, isto significa que uma equipa deve saber o básico da computação Cloud, incluindo o terminal Linux e como navegar nele. Da mesma forma, uma compreensão firme de DevOps - ou seja, como configurar e usar CI/CD - é um componente necessário para fazer MLOps. Este exercício final é um excelente teste às tuas capacidades antes de mergulhares em tópicos mais matizados mais à frente no livro e reúne todos estes componentes fundamentais num projeto minimalista ao estilo MLOps.

No próximo capítulo, abordaremos os contentores e os dispositivos de ponta. Estes são componentes essenciais da maioria das plataformas MLOps, como o AWS SageMaker ou o Azure ML Studio, e baseiam-se nos conhecimentos abordados neste capítulo.

Exercícios

- Executa um projeto hello world Python no GitHub, verifica-o e executa os teus testes nas três Clouds: AWS, Azure e GCP.
- Cria uma nova aplicação Flask que sirva uma rota do tipo "hello world" utilizando o AWS Elastic Beanstalk que aches que seria útil para outras pessoas e coloca o código num repositório do GitHub juntamente com uma captura de ecrã do mesmo a servir um pedido no *README.md* do GitHub. Em seguida, cria um processo de entrega contínua para implantar o aplicativo Flask usando o AWS CodeBuild.
- Bifurca [este repositório](#) que contém uma aplicação de aprendizagem automática Flask e implementa-a com entrega contínua na AWS utilizando o Elastic Beanstalk e o Code Pipeline.

- Bifurca [este repositório](#) que contém uma aplicação de aprendizagem automática Flask e implementa-a com entrega contínua no GCP utilizando o Google App Engine e o Cloud Build ou o Cloud Run e o Cloud Build.
- Bifurca [este repositório](#) que contém uma aplicação de aprendizagem automática Flask e implementa-a com entrega contínua no Azure utilizando os Serviços de Aplicações do Azure e os Pipelines DevOps do Azure.
- Utiliza o exemplo de código do caixeiro-viajante e transporta-o para trabalhar com coordenadas obtidas a partir de uma API, por exemplo, todos os melhores restaurantes de uma cidade que queiras visitar. Nunca mais vais pensar nas férias da mesma forma.
- Usando o [TensorFlow Playground](#), experimenta alterar os hiperparâmetros em diferentes conjuntos de dados, bem como em tipos de problemas. Consegues identificar as configurações ideais de camadas ocultas, taxa de aprendizagem e taxa de regularização para diferentes conjuntos de dados?

Questões para discussão sobre pensamento crítico

- Uma empresa especializada em bases de dados GPU tem um membro técnico importante que defende que *deixem de* utilizar a Cloud porque seria muito mais prático comprar o seu hardware GPU, uma vez que o utilizam 24 horas por dia, 7 dias por semana. Este passo também lhes permitiria obter acesso a GPUs especializadas muito mais rapidamente do que as que estão disponíveis. Por outro lado, outro membro técnico crítico, que tem *todas as* certificações da AWS, prometeu demiti-lo se ele se atrevesse a tentar. Afirma que já investiram demasiado na AWS. Argumenta a favor ou contra esta proposta.

- Um "Engenheiro certificado pela Red Hat" construiu um dos centros de dados mais bem-sucedidos do Sudeste para uma empresa com apenas 100 funcionários. Embora a empresa seja uma empresa de comércio eletrónico e não uma empresa de Cloud, afirma que isto lhe dá uma enorme vantagem.

Por outro lado, um "Google Certified Architect" e um "Duke Data Science Masters" afirmam que a empresa está numa posição de risco ao utilizar um centro de dados de que são proprietários.

Salienta que a empresa continua a perder engenheiros de centros de dados para a Google e não tem um plano de recuperação de desastres ou tolerância a falhas. Argumenta a favor ou contra esta proposta.

- Quais são as principais diferenças técnicas entre o AWS Lambda e o AWS Elastic Beanstalk, incluindo os prós e os contras de cada solução?
- Porque é que um serviço de ficheiros gerido como o EFS on AWS ou o Google Filestore seria útil num fluxo de trabalho MLOps do mundo real na América corporativa?
- O Kaizen começa com uma pergunta simples: podemos fazer melhor? Em caso afirmativo, o que devemos fazer para melhorar esta semana ou hoje? Por fim, como podemos aplicar o Kaizen aos nossos projectos de aprendizagem automática?

Capítulo 3. MLOps para contêineres e dispositivos de borda

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

As experiências com o cérebro dividido começaram com o problema da transferência interocular. Ou seja, se aprenderes com um olho a resolver um problema, com esse olho tapado e usando o outro olho, resolvest facilmente o problema sem teres de aprender mais. A isto chama-se "transferência interocular da aprendizagem". É claro que a aprendizagem não ocorre num olho e depois é transferida para o outro olho, mas é assim que é normalmente descrita. O facto de a transferência ocorrer pode parecer óbvio, mas é no questionamento do óbvio que muitas vezes se produzem as descobertas. Neste caso, a pergunta era: Como é que a aprendizagem com um olho pode aparecer com o uso do outro? Em termos experimentais, onde é que os dois olhos estão ligados? As experiências mostraram que a transferência ocorre de facto entre os hemisférios através do corpo caloso.

—Dr. Joseph Bogen

Quando comecei a trabalhar em tecnologia, as máquinas virtuais (servidores virtualizados alojados numa máquina física) estavam bem posicionadas e generalizadas - era fácil encontrá-las em todo o lado, desde fornecedores de alojamento a empresas normais com grandes servidores na sala de TI. Muitos fornecedores de software online estavam a oferecer alojamento virtualizado. No trabalho, aperfeiçoei minhas habilidades, tentando

aprender o máximo possível sobre virtualização. A capacidade de executar máquinas virtuais dentro de outro host oferecia muita flexibilidade (bem-vinda).

Sempre que uma nova tecnologia resolve um problema (ou qualquer número de problemas, na verdade), um rastro de outros problemas vem junto para ser resolvido. Com as máquinas virtuais, um desses problemas era lidar com a sua deslocação. Se o servidor anfitrião *A* precisasse de um novo sistema operativo instalado, o administrador de sistemas teria de mover as máquinas virtuais para o servidor anfitrião *B*. As máquinas virtuais eram tão grandes como os dados quando inicialmente configuradas: uma *unidade virtual* de 50 GB significava que um ficheiro que representava a unidade virtual tinha 50 gigabytes. Mover cerca de 50 gigabytes de um servidor para o outro levaria tempo. Se estás a deslocar um serviço crítico que executa uma máquina virtual, como podes minimizar o tempo de inatividade?

A maioria desses problemas tinha suas estratégias para minimizar o tempo de inatividade e aumentar a robustez: snapshots, recuperação, backups. Projectos de software como o *Xen Project* e o *VMWare* especializaram-se em tornar estes problemas relativamente fáceis de resolver, e os fornecedores de Cloud praticamente eliminaram-nos.

Atualmente, as máquinas virtuais continuam a ter um lugar importante nas ofertas Cloud. O Google Cloud chama-lhes *Compute Engine*, por exemplo, e outros fornecedores têm uma referência semelhante. Muitas destas máquinas virtuais oferecem GPUs melhoradas para proporcionar um melhor desempenho direcionado para operações de aprendizagem automática.

Embora as máquinas virtuais tenham vindo para ficar, é cada vez mais importante compreender dois tipos de tecnologias para implantações de modelos: contêineres e dispositivos de borda. Não é razoável pensar que uma máquina virtual seria adequada para ser executada em um dispositivo de borda (como um celular) ou iterar rapidamente durante o desenvolvimento com um conjunto reproduzível de arquivos. Nem sempre

terás de tomar a decisão de utilizar uma ou outra, mas ter uma compreensão clara destas opções (e de como funcionam) fará de ti um melhor engenheiro de aprendizagem automática.

Contentores

Com todo o poder e robustez das máquinas virtuais, é fundamental entender os containers e a tecnologia de containerização em geral. Lembro-me de estar na plateia da PyCon em Santa Clara em 2013, quando o Docker foi anunciado. A sensação foi incrível! A *virtualização enxuta* demonstrada não era nova para o Linux. O que era novo e meio que revolucionário era a ferramenta. O Linux tinha o *LXC* (ou *contentores Linux*), que fornecia muitas das funcionalidades que hoje damos por garantidas com os contentores. Mas as ferramentas para LXC são péssimas, e o Docker trouxe um fator chave para se tornar um líder de sucesso: colaboração e partilha fáceis através de um registo.

Os registos permitem que qualquer programador *envie* as suas alterações para uma localização central, onde outros podem depois *obter* essas alterações e executá-las localmente. O suporte a registros com a mesma ferramenta que lida com contêineres (e tudo sem problemas) impulsionou a tecnologia a um ritmo incrível.

DICA

Para esta secção, certifica-te de que tens um tempo de execução de contentor instalado. Para os exemplos desta secção, pode ser mais fácil utilizar [o Docker](#). Após a instalação, certifica-te de que o comando `docker` mostra a saída de ajuda para verificar se a instalação foi bem-sucedida.

Uma das [descrições](#) mais significativas que vi sobre contentores em comparação com máquinas virtuais vem da Red Hat. Resumidamente, os contentores têm tudo a ver com a aplicação em si, e apenas com o que a aplicação é (como o código fonte e outros ficheiros de suporte) versus o que precisa de ser executado (como bases de dados). Tradicionalmente, os

engenheiros costumam usar máquinas virtuais como um serviço tudo-em-um em que o banco de dados, o servidor da Web e qualquer outro serviço do sistema são instalados, configurados e executados. Esses tipos de aplicativos são *monólitos*, com interdependências vinculadas em uma máquina tudo-em-um.

Um *microsserviço*, por outro lado, é uma aplicação que está totalmente dissociada dos requisitos do sistema, como bases de dados, e pode ser executada de forma independente. Embora possa usar máquinas virtuais como microsserviços, é mais comum encontrar contentores que se encaixam melhor nesse conceito.

Se já estás familiarizado com a criação e a execução de contentores, em "[Infrastructure as Code for Continuous Delivery of ML Models](#)" ([Infraestrutura como código para entrega contínua de modelos de ML](#)), aborda a forma de os construir programaticamente com modelos pré-treinados, o que pega nestes conceitos e melhora-os com automação.

Tempo de execução do contentor

Deves ter reparado que mencionei contentores, Docker e tempo de execução de contentores. Esses termos podem ser confusos, principalmente quando as pessoas os usam de forma intercambiável. Uma vez que a Docker (a empresa) desenvolveu inicialmente as ferramentas para criar, gerir e executar contentores, tornou-se comum dizer "contentor Docker". O tempo de execução - ou seja, o software necessário para executar um contentor num sistema - também foi criado pela Docker. Alguns anos após o lançamento inicial da nova tecnologia de contentores, a Red Hat (a empresa por detrás do sistema operativo RHEL) contribuiu para criar uma forma diferente de executar contentores, com um novo ambiente de tempo de execução (alternativo). Este novo ambiente também trouxe um novo conjunto de ferramentas para operar contentores, com alguma compatibilidade com as fornecidas pelo Docker. Se alguma vez ouvires falar de runtime de contentores, tens de ter em atenção que existe mais do que um.

Alguns benefícios destas novas ferramentas e tempo de execução significam que já não é necessário utilizar uma conta de superutilizador com privilégios extensos, o que faz sentido para muitos casos de utilização diferentes. Embora a Red Hat e muitos outros colaboradores de código aberto tenham feito um excelente trabalho com estas ferramentas, ainda é um pouco complicado executá-las em sistemas operativos que não sejam o Linux. Por outro lado, o Docker torna o trabalho uma experiência perfeita, independentemente de estares a usar Windows, MacOS ou Linux. Vamos começar por percorrer todos os passos necessários para criar um contentor.

Criar um contentor

O Dockerfile está no centro da criação de contentores. Sempre que estiveres a criar um contentor, tens de ter o Dockerfile presente no diretório atual. Este ficheiro especial pode ter várias secções e comandos que permitem criar uma imagem de contentor. Abre um novo ficheiro e dá-lhe o nome de *Dockerfile* e adiciona-lhe o seguinte conteúdo:

```
FROM centos:8  
RUN dnf install -y python38
```

O ficheiro tem duas secções; uma palavra-chave única delimita cada uma delas. Estas palavras-chave são conhecidas como *instruções*. O início do ficheiro utiliza a instrução **FROM**, que determina qual é a base para o contentor. A *base* (também referida como *imagem base*) é a distribuição CentOS na versão 8. A versão, neste caso, é uma tag. As tags em containers definem um ponto no tempo. Quando não há tags definidas, o padrão é a tag *mais recente*. É comum ver versões usadas como tags, como é o caso neste exemplo.

Um dos muitos aspectos úteis dos contentores é que podem ser compostos por muitas camadas, e estas camadas podem ser utilizadas ou reutilizadas noutras contentores. Este fluxo de trabalho de camadas evita que uma camada base de 10 megabytes seja descarregada todas as vezes para cada contentor que a utiliza. Na prática, descarregarás a camada de 10 megabytes

uma vez e reutilizá-la-ás muitas vezes. Isto é *muito* diferente das máquinas virtuais, onde não importa se estas máquinas virtuais têm todos os mesmos ficheiros: tens de os descarregar todos como um todo.

De seguida, a instrução RUN corre um comando de sistema. Este comando de sistema instala o Python 3, que não está incluído na imagem base do CentOS 8. Observa como o comando dnf usa o sinalizador -y, que evita um prompt para confirmação pelo instalador, acionado ao construir o contêiner. É crucial evitar qualquer prompt de comandos em execução, pois isso interromperia a construção.

Agora constrói o contentor a partir do mesmo diretório onde está o *Dockerfile*:

```
$ docker build .
[+] Building 11.2s (6/6) FINISHED
=> => transferring context: 2B
=> => transferring dockerfile: 83B
=> CACHED [1/2] FROM docker.io/library/centos:8
=> [2/2] RUN dnf install -y python38
=> exporting to image
=> => exporting layers
=> => writing
image
sha256:3ca470de8dbd5cb865da679ff805a3bf17de9b34ac6a7236dbf0c367e1
fb4610
```

A saída informa que eu já tenho a camada inicial para o CentOS 8, então não há necessidade de puxá-la novamente. Depois, instala o Python 3.8 para completar a criação da imagem. Certifica-te de que inicias uma compilação apontando para onde o *Dockerfile* está presente. Neste caso, eu estou no mesmo diretório, então eu uso um ponto para que o build saiba que o diretório atual é o diretório a partir do qual o build deve ser feito.

Esta forma de construir imagens não é muito robusta e tem alguns problemas. Primeiro, é difícil identificar esta imagem mais tarde. Tudo o que temos é o digest sha256 para referenciá-la e nada mais. Para ver algumas informações sobre a imagem que acabou de ser construída, executa novamente docker:

```
$ docker images
docker images
REPOSITORY      TAG      IMAGE ID      CREATED
SIZE
<none>          <none>    3ca470de8dbd    15 minutes ago
294MB
```

Não tem qualquer repositório ou etiqueta associada. O ID da imagem é o resumo, que é reduzido para apenas 12 caracteres. Vai ser difícil lidar com esta imagem se não tiver metadados adicionais. É uma boa prática colocar uma etiqueta ao criar a imagem. É assim que crias a mesma imagem e a etiquetas:

```
$ docker build -t localbuild:removeme .
[+] Building 0.3s (6/6) FINISHED
[...]
=> => writing
image
sha256:4c5d79f448647e0ff234923d8f542eea6938e0199440dfc75b8d7d0d10
d5ca9a
0.0s
=> => naming to docker.io/library/localbuild:removeme
```

A diferença fundamental é que agora `localbuild` tem uma etiqueta de `removeme` e aparecerá quando listares as imagens:

```
$ docker images localbuild
REPOSITORY      TAG      IMAGE ID      CREATED
SIZE
localbuild      removeme    3ca470de8dbd    22 minutes
ago            294MB
```

Como a imagem não mudou nada, o processo de construção foi rápido e, internamente, o sistema de construção marcou a imagem já construída. A nomeação e etiquetagem de imagens ajuda quando se empurra a imagem para um registo. Eu precisaria ser o dono do repositório `localbuild` para fazer push para ele. Como não possuo, o push será negado:

```
$ docker push localbuild:removeme
The push refers to repository [docker.io/library/localbuild]
```

```
denied: requested access to the resource is denied
```

No entanto, se eu remarcar o contentor para o meu repositório no registo, o push funcionará. Para voltar a marcar, primeiro tenho de fazer referência à marca original (`localbuild:removeme`) e, em seguida, utilizar a minha conta de registo e destino (`alfredodeza/removeme`):

```
$ docker tag localbuild:removeme alfredodeza/removeme
$ docker push alfredodeza/removeme
The push refers to repository [docker.io/alfredodeza/removeme]
958488a3c11e: Pushed
291f6e44771a: Pushed
latest: digest:
sha256:a022eea71ca955cafb4d38b12c28b9de59dbb3d9fcb54b size: 741
```

Go para o [registro](#) (neste caso, Docker Hub) mostra agora que a imagem recentemente enviada está disponível([Figura 3-1](#)).

Uma vez que a minha conta está aberta e o registo não está a restringir o acesso, qualquer pessoa pode "puxar" a imagem do contentor executando: `docker pull alfredodeza/removeme`. Se não estiveste exposto a contentores ou registos antes, isto deve parecer revolucionário. Como mencionei no início deste capítulo, é a base da razão pela qual os containers se tornaram virais na comunidade de desenvolvedores. A resposta para "*Como eu instalo seu software?*" agora pode ser "*apenas puxe o container*" para quase tudo.

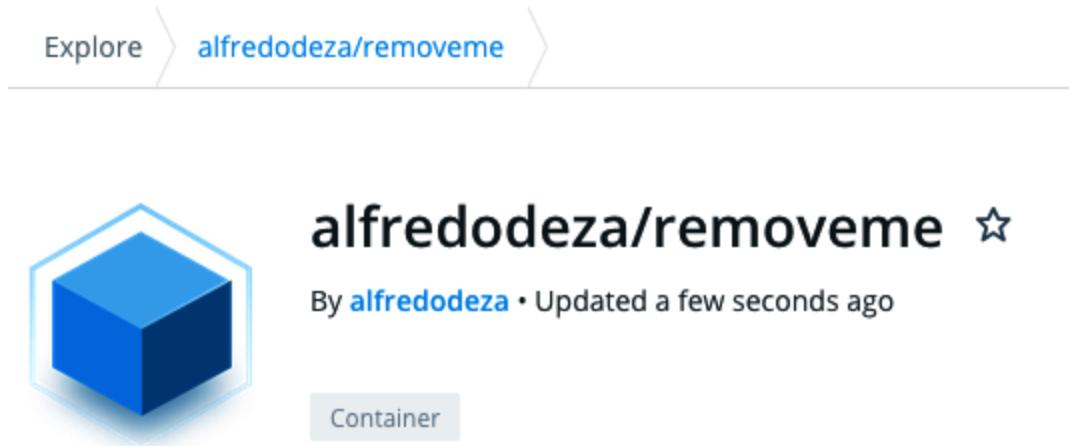


Figura 3-1. Imagem do Docker Hub

Executar um contentor

Agora que o contêiner foi construído com o *Dockerfile*, podemos executá-lo. Quando executas máquinas virtuais, é prática comum ativar o daemon SSH (também conhecido como Secure Shell) e expor uma porta para acesso remoto, e talvez até adicionar chaves SSH predefinidas para evitar receber prompts de palavra-passe. As pessoas que não estão habituadas a executar um contentor irão provavelmente pedir acesso SSH a uma instância de contentor em execução. O acesso SSH não é necessário; mesmo que possas activá-lo e fazê-lo funcionar, não é a forma de aceder a um contentor em execução.

Certifica-te de que um contentor está em execução. Neste exemplo, executo o CentOS 8:

```
$ docker run -ti -d --name centos-test --rm centos:8 /bin/bash  
1bb9cc3112ed661511663517249898bfc9524fc02dedc3ce40b5c4cb982d7bcd
```

Existem várias bandeiras novas neste comando. Usa `-ti` para alocar um TTY (emula um terminal) e anexa `stdin` a ele para interagir com ele no terminal mais tarde. A seguir, a bandeira `-d` faz com que o contentor corra em segundo plano para evitar assumir o controlo do terminal atual. Atribui um nome (`centos-test`) e depois usa `--rm` para que o Docker remova

este contentor depois de o parar. Depois de emitir o comando, um digest retorna, indicando que o contêiner foi iniciado. Agora, verifica se ele está em execução:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             NAMES
1bb9cc3112ed        centos:8           "/bin/bash"
centos-test
```

Alguns containers são criados com uma instrução **ENTRYPOINT** (e opcionalmente uma **CMD**). Essas instruções são destinadas a colocar o container em funcionamento para uma tarefa específica. No exemplo de contentor que acabámos de construir para o CentOS, o executável **/bin/bash** teve de ser especificado porque, caso contrário, o contentor não se manteria em execução. Estas instruções significam que se quiseres um contentor de longa duração, deves criá-lo com pelo menos um **ENTRYPOINT** que execute um programa. Atualiza o *Dockerfile* para que fique assim:

```
FROM centos:8
RUN dnf install -y python38
ENTRYPOINT ["/bin/bash"]
```

Agora é possível executar o contentor em segundo plano sem a necessidade de especificar o comando **/bin/bash**:

```
$ docker build -t localbuild:removeme .
$ docker run --rm -it -d localbuild:removeme
023c8e67d91f3bb3998e7ac1b9b335fe20ca13f140b6728644fd45fb6ccb9132
$ docker ps
CONTAINER ID        IMAGE               COMMAND             NAMES
023c8e67d91f        removeme           "/bin/bash"
romantic_khayyam
```

Mencionei anteriormente como é comum usar o SSH para obter acesso a uma máquina virtual e que obter acesso a um contêiner é um pouco

diferente. Embora possas ativar o SSH para um contentor (em teoria), não o recomendo. É assim que podes obter acesso a um contentor em execução utilizando o ID do contentor e o subcomando `exec`:

```
$ docker exec -it 023c8e67d91f bash  
[root@023c8e67d91f /]# whoami  
root
```

Nesse caso, tenho de usar o comando que quero executar. Uma vez que quero manipular interactivamente o contentor (como faria com uma máquina virtual), chamo o executável do programa Bash (uma shell e linguagem de comandos Unix omnipresente).

Alternativamente, podes não querer obter acesso usando um ambiente de shell interativo, e tudo o que queres fazer é executar um comando. Para isso, o comando deve ser alterado. Substitui o executável da shell utilizado no exemplo anterior pelo do comando a utilizar:

```
$ docker exec 023c8e67d91f tail /var/log/dnf.log  
  
python38-setuptools-wheel-41.6.0-  
4.module_el8.2.0+317+61fa6e7d.noarch  
  
2020-12-02T13:00:04Z INFO Complete!  
2020-12-02T13:00:04Z DDEBUG Cleaning up.
```

Uma vez que não há necessidade interactiva (não estou a enviar qualquer entrada através de uma shell), posso omitir a bandeira `-it`.

NOTA

Um aspeto comum do desenvolvimento de contentores é manter o seu tamanho o mais pequeno possível. É por isso que um container CentOS terá muito menos pacotes do que uma máquina virtual CentOS recém-instalada. Isso leva a experiências surpreendentes quando esperas que um pacote esteja presente (por exemplo, um editor de texto como o Vim) e ele não está.

Melhores práticas

A primeira coisa que faço (e recomendo vivamente) quando experimento uma nova linguagem ou ferramenta é encontrar um linter que possa ajudar a navegar por convenções e utilizações comuns com as quais posso não estar familiarizado. Existem alguns linters para criar containers com um Dockerfile. Um desses linters é `hadolint`. Ele é convenientemente empacotado como um container. Modifica o último exemplo de `Dockerfile`, para que fique parecido com isto:

```
FROM centos:8

RUN dnf install -y python38

RUN pip install pytest

ENTRYPOINT ["/bin/bash"]
```

Agora corre o linter para ver se há alguma sugestão boa:

```
$ docker run --rm -i hadolint/hadolint < Dockerfile
DL3013 Pin versions in pip.
    Instead of `pip install <package>` use `pip install <package>==<version>`
```

Esta é uma boa sugestão. Fixar pacotes é sempre uma óptima ideia porque estás a salvo de uma atualização numa dependência que seja incompatível com o código de que a tua aplicação necessita. Tem em atenção que fixar dependências e nunca passar pela tarefa de as atualizar não é uma boa ideia. Certifica-te de que voltas às dependências fixadas e vês se seria útil actualizá-las.

Uma vez que um dos objetivos das ferramentas de conteinerização é manter as tão pequenas quanto possível, podes realizar algumas coisas ao criar um Dockerfile. Sempre que há uma instrução RUN, uma nova camada é criada com essa execução. Os contentores são constituídos por camadas individuais, pelo que quanto menor for o número de camadas, menor será o tamanho do contentor. Isso significa que é preferível usar uma única linha para instalar muitas dependências em vez de uma:

```
RUN apk add --no-cache python3 && python3 -m ensurepip && pip3
install pytest
```

O uso de `&&` no final de cada comando encadeia tudo, criando uma única camada. Se o exemplo anterior tivesse uma instrução RUN separada para cada comando de instalação, o container acabaria sendo maior. Talvez para este exemplo em particular, o tamanho não faria muita diferença; no entanto, seria significativo em containers que requerem muitas dependências.

Há uma opção útil que o linting pode oferecer: a oportunidade de automatizar o linting. Fica atento às oportunidades de automatizar processos, removendo qualquer etapa manual e permitindo que te concentres nas partes essenciais do processo de envio de modelos para produção (escrevendo um bom Dockerfile, neste caso).

Outra parte crítica da construção de contentores é garantir que não existem vulnerabilidades associadas ao software instalado. Não é raro encontrar engenheiros que pensam que é improvável que a aplicação tenha vulnerabilidades porque escrevem código de alta qualidade. O problema é que um contentor vem com bibliotecas pré-instaladas. É um sistema operativo completo que, em tempo de compilação, vai puxar dependências extra para satisfazer a aplicação que estás a tentar entregar. Se vais servir um modelo treinado a partir do contentor usando uma estrutura web como o Flask, tens de estar bem ciente de que pode haver Vulnerabilidades e Exposições Comuns (CVEs) associadas ao Flask ou a uma das suas dependências.

Estas são as dependências que o Flask (na versão 1.1.2) traz:

```
click==7.1.2
itsdangerous==1.1.0
Jinja2==2.11.2
MarkupSafe==1.1.1
Werkzeug==1.0.1
```

Os CVEs podem ser relatados a qualquer momento, e os sistemas de software usados para alertar sobre vulnerabilidades garantem que eles sejam

atualizados várias vezes durante o dia para relatar com precisão quando isso acontece. Uma peça crítica da tua aplicação, como o Flask, pode não estar vulnerável hoje para a versão 1.1.2, mas pode, sem dúvida, estar amanhã de manhã, quando um novo CVE for descoberto e relatado. Muitas soluções diferentes são especializadas em verificar e relatar vulnerabilidades em containers para mitigar essas vulnerabilidades. Essas ferramentas de segurança examinam uma biblioteca que seu aplicativo instala e os pacotes do sistema operacional, fornecendo um relatório de vulnerabilidade detalhado e preciso.

Uma solução que é muito rápida e fácil de instalar é a ferramenta de linha de comando `grype` da Anchore. Para a instalar num computador Macintosh:

```
$ brew tap anchore/grype  
$ brew install grype
```

Ou em qualquer máquina Linux:

```
$ curl -sSfL \https://raw.githubusercontent.com/anchore/grype/main/install.sh | sh -s
```

Usando `curl` dessa forma, permite implantar `grype` em quase todos os sistemas de integração contínua para verificar vulnerabilidades. O método de instalação `curl` colocará o executável no caminho de trabalho atual sob um diretório `bin/`. Depois que a instalação for concluída, executa-o em um contêiner:

```
$ grype python:3.8  
✓ Vulnerability DB      [no update available]  
.. Loading image          _____ [requesting image  
from docker]  
✓ Loaded image  
✓ Parsed image  
✓ Cataloged image        [433 packages]  
✓ Scanned image          [1540 vulnerabilities]
```

Mais de mil vulnerabilidades parecem um pouco surpreendentes. A saída é demasiado longa para ser capturada aqui, por isso filtra o resultado para verificar as vulnerabilidades com uma gravidade *Alta*:

```
$ grype python:3.8 | grep High  
[...]  
python2.7      2.7.16-2+deb10u1      CVE-2020-8492      High
```

Havia algumas vulnerabilidades relatadas, então reduzi a saída para apenas uma. O **CVE** é preocupante porque pode potencialmente permitir que o sistema trave se um atacante explorar a vulnerabilidade. Como eu sei que a aplicação usa Python 3.8, então este container não é vulnerável porque Python 2.7 não é usado. Embora este seja um container Python 3.8, a imagem contém uma versão mais antiga por conveniência. A diferença crítica é que agora sabes o que é vulnerável e podes tomar uma decisão executiva para a eventual implementação do serviço em produção.

Um aprimoramento útil da automação é falhar em um nível de vulnerabilidade específico, como `high`:

```
$ grype --fail-on=high centos:8  
[...]  
discovered vulnerabilities at or above the severity threshold!
```

Esta é outra verificação que podes automatizar juntamente com o linting para uma construção robusta de contentores. Um *Dockerfile* bem escrito com relatórios constantes sobre vulnerabilidades é uma excelente maneira de aprimorar a entrega de produção de modelos em contêineres.

Servindo um modelo treinado por HTTP

Agora que alguns dos principais conceitos de criação de um container estão claros, vamos criar um container que servirá um modelo treinado através de uma API HTTP usando o framework web Flask. Como já sabes, tudo começa com o *Dockerfile*, por isso crie um, assumindo por agora que um ficheiro *requirements.txt* está presente no diretório de trabalho atual:

```
FROM python:3.7

ARG VERSION

LABEL org.label-schema.version=$VERSION

COPY ./requirements.txt /webapp/requirements.txt

WORKDIR /webapp

RUN pip install -r requirements.txt

COPY webapp/* /webapp

ENTRYPOINT [ "python" ]

CMD [ "app.py" ]
```

Há algumas coisas novas neste ficheiro que não abordei anteriormente. Primeiro, definimos um argumento chamado **VERSION** que é utilizado como uma variável para um **LABEL**. Estou a utilizar uma **convenção de esquema de etiquetas** que é útil para normalizar a forma como estas etiquetas são nomeadas. Usar uma versão é uma forma útil de adicionar metadados informativos sobre o próprio contentor. Usarei esse rótulo mais tarde quando quiser identificar a versão do modelo treinado. Imagina uma situação em que um contentor não está a produzir a precisão esperada de um modelo; adicionar uma etiqueta ajuda a identificar a versão do modelo problemático. Embora este ficheiro utilize uma etiqueta, podes imaginar que quanto mais etiquetas com dados descriptivos, melhor.

NOTA

Existe uma ligeira diferença na imagem do contentor utilizada. Esta compilação usa Python 3.7 porque, no momento da escrita, algumas das dependências ainda não funcionam com Python 3.8. Sente-te à vontade para trocar o 3.7 pelo 3.8 e verifica se agora funciona.

Em seguida, um arquivo *requirements.txt* é copiado para o contêiner. Cria o arquivo de requisitos com as seguintes dependências:

```
Flask==1.1.2
pandas==0.24.2
scikit-learn==0.20.3
```

Agora, crie um novo diretório chamado *webapp* para que os ficheiros web estejam contidos num único local e adiciona o ficheiro *app.py* para que fique com este aspeto:

```
from flask import Flask, request, jsonify

import pandas as pd
from sklearn.externals import joblib
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)

def scale(payload):
    scaler = StandardScaler().fit(payload)
    return scaler.transform(payload)

@app.route("/")
def home():
    return "<h3>Sklearn Prediction Container</h3>

@app.route("/predict", methods=['POST'])
def predict():
    """
    Input sample:

    {
        "CHAS": { "0": 0 }, "RM": { "0": 6.575 },
        "TAX": { "0": 296 }, "PTRATIO": { "0": 15.3 },
        "B": { "0": 396.9 }, "LSTAT": { "0": 4.98 }
    }

    Output sample:

    {
        "prediction": [ 20.35373177134412 ]
    }

    clf = joblib.load("boston_housing_prediction.joblib")
    inference_payload = pd.DataFrame(request.json)
    scaled_payload = scale(inference_payload)
    prediction = list(clf.predict(scaled_payload))
    return jsonify({'prediction': prediction})
```

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

O último arquivo necessário é o modelo treinado. Se estiveres a treinar o conjunto de dados de previsão de habitação de Boston, certifica-te de que o colocas no diretório *webapp* juntamente com o ficheiro *app.py* e lhe dás o nome de *boston_housing_prediction.joblib*. Também podes encontrar uma versão treinada do modelo neste [repositório do GitHub](#).

A estrutura final do projeto deve ter o seguinte aspeto:

```
.
├── Dockerfile
└── webapp
    ├── app.py
    └── boston_housing_prediction.joblib
```

1 directory, 3 files

Agora constrói o contentor. No exemplo, usarei a ID de execução que o Azure me forneceu quando treinei o modelo como a versão para facilitar a identificação de onde o modelo veio. Podes utilizar uma versão diferente (ou nenhuma versão, se não precisares de uma):

```
$ docker build --build-arg VERSION=AutoML_287f444c -t flask-
predict .
[+] Building 27.1s (10/10) FINISHED
=> => transferring dockerfile: 284B
=> [1/5] FROM docker.io/library/python:3.7
=> => resolve docker.io/library/python:3.7
=> [internal] load build context
=> => transferring context: 635B
=> [2/5] COPY ./requirements.txt /webapp/requirements.txt
=> [3/5] WORKDIR /webapp
=> [4/5] RUN pip install -r requirements.txt
=> [5/5] COPY webapp/* /webapp
=> exporting to image
=> => writing image
sha256:5487a63442aae56d9ea30fa79b0c7eed1195824aad7ff4ab42b
=> => naming to docker.io/library/flask-predict
```

Verifica novamente se a imagem está agora disponível após a construção:

```
$ docker images flask-predict
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
flask-predict    latest   5487a63442aa  6 minutes ago  1.15GB
```

Agora executa o contentor em segundo plano, expondo a porta **5000**, e verifica se está funcionar:

```
$ docker run -p 5000:5000 -d --name flask-predict flask-predict
d95ab6581429ea79495150bea507f009203f7bb117906b25ffd9489319219281
$ docker ps
CONTAINER ID IMAGE      COMMAND      STATUS      PORTS
d95ab6581429 flask-predict "python app.py" Up 2 seconds
0.0.0.0:5000->5000/tcp
```

No browser, abre *http://localhost:5000* e o HTML da função `home()` deve dar as boas-vindas à aplicação Sklearn Prediction. Outra maneira de verificar se isso está conectado corretamente é usar `curl`:

```
$ curl 192.168.0.200:5000
<h3>Sklearn Prediction Container</h3>
```

Podes utilizar qualquer ferramenta que possa enviar informações por HTTP e processar uma resposta. Este exemplo utiliza algumas linhas de Python com a biblioteca `requests` (certifica-te de que a instalas antes de a executares) para enviar um pedido POST com os dados JSON de amostra:

```
import requests
import json

url = "http://localhost:5000/predict"

data = {
    "CHAS": {"0": 0},
    "RM": {"0": 6.575},
    "TAX": {"0": 296.0},
    "PTRATIO": {"0": 15.3},
    "B": {"0": 396.9},
    "LSTAT": {"0": 4.98},
}
# Convert to JSON string
```

```
input_data = json.dumps(data)

# Set the content type
headers = {"Content-Type": "application/json"}

# Make the request and display the response
resp = requests.post(url, input_data, headers=headers)
print(resp.text)
```

Escreve o código Python num ficheiro e chama-lhe *predict.py*. Executa o script para obteres algumas previsões no terminal:

```
$ python predict.py
{
    "prediction": [
        20.35373177134412
    ]
}
```

A implementação de contentores é uma excelente forma de criar dados portáteis que podem ser experimentados por outros. Ao compartilhar um contêiner, o atrito da configuração do ambiente é bastante reduzido, garantindo um sistema repetível com o qual interagir. Agora que já sabes como criar, executar, depurar e implementar contentores para ML, podes tirar partido disto para começar a automatizar ambientes não contentorizados para acelerar as implementações de produção e melhorar a robustez de todo o processo. Para além dos contentores, há um impulso para aproximar os serviços do utilizador, e é isso que irei abordar a seguir com dispositivos e implementações de ponta.

Dispositivos de borda

O custo computacional da inferência (rápida) era astronómico há alguns anos. Algumas das capacidades mais avançadas da aprendizagem automática que estão atualmente disponíveis tinham custos proibitivos há pouco tempo. Não só os custos baixaram, como estão a ser produzidos chips mais potentes. Alguns destes chips são explicitamente concebidos para tarefas de aprendizagem automática. A combinação certa das

características necessárias para estes chips permite a inferência em dispositivos como os telemóveis: rápidos, pequenos e feitos para tarefas de aprendizagem automática. Quando se fala de "implantação no extremo" em tecnologia, refere-se a dispositivos de computação que não estão num centro de dados juntamente com milhares de outros servidores. Telemóveis, Raspberry Pi e dispositivos domésticos inteligentes são alguns exemplos que se enquadram na descrição de um "dispositivo de ponta". Nos últimos anos, as grandes empresas de telecomunicações têm vindo a apostar na computação periférica. A maioria destas implantações de ponta pretende obter um feedback mais rápido para os utilizadores em vez de encaminhar pedidos de computação dispendiosos para um centro de dados remoto.

A ideia geral é que quanto mais próximos os recursos de computação estiverem do utilizador, mais rápida será a experiência do utilizador. Há uma linha ténue que divide o que pode chegar ao limite e o que deve ir até ao centro de dados e voltar. Mas, como já mencionei, os chips especializados estão a ficar mais pequenos, mais rápidos e mais eficazes; faz sentido prever que o futuro significa mais ML no edge. E a borda, neste caso, significará cada vez mais dispositivos que antes não pensávamos que poderiam lidar com tarefas de ML.

A maior parte das pessoas que vivem em países com muitos centros de dados que alojam dados de aplicações não sofrem de grande atraso. Para os países que não têm, o problema é exacerbado. Por exemplo, o Peru tem vários cabos submarinos que o ligam a outros países da América do Sul, mas não tem ligação direta aos EUA. Isto significa que se estiveres a carregar uma imagem do Peru para um serviço que aloja a sua aplicação num centro de dados nos EUA, demorará exponencialmente mais tempo do que num país como o Panamá, com vários cabos que ligam à América do Norte. Este exemplo de carregamento de uma fotografia é trivial, mas torna-se ainda pior quando operações computacionais, como previsões de ML, são efectuadas nos dados enviados. Esta secção explora algumas formas diferentes de como os dispositivos periféricos podem ajudar, realizando inferências rápidas o mais próximo possível do utilizador. Se as longas distâncias são um problema, imagina o que acontece quando não há

conetividade (ou esta é muito limitada), como numa quinta remota. Se precisares de uma inferência rápida num local remoto, as opções são limitadas, e é aqui que a *implementação no extremo* tem uma vantagem sobre qualquer centro de dados.

Lembra-te, os utilizadores não se importam muito com a colher: estão interessados em ter uma forma simples de provar a deliciosa sopa.

Coral

O Projeto Coral é uma plataforma que ajuda a criar inferências locais (no dispositivo) que captam a essência das implementações de ponta: rápidas, próximas do utilizador e offline. Nesta secção, abordarei o **Acelerador USB**, que é um dispositivo de ponta que suporta todos os principais sistemas operativos e funciona bem com os modelos TensorFlow Lite. Podes compilar a maioria dos modelos TensorFlow Lite para serem executados nesta TPU (Tensor Processing Unit) de ponta. Alguns aspectos da operacionalização do ML significam estar ciente do suporte do dispositivo, dos métodos de instalação e da compatibilidade. Estes três aspectos são verdadeiros em relação à TPU Coral Edge: funciona na maioria dos sistemas operativos, com modelos TensorFlow Lite, desde que possam ser compilados para serem executados na TPU.

Se tiveres a tarefa de implementar uma solução de inferência rápida no edge numa localização remota, tens de garantir que todas as peças necessárias para essa implementação funcionam corretamente. Este conceito central do DevOps é abordado ao longo deste livro: métodos de implantação repetíveis que criam ambientes reproduzíveis são essenciais. Para garantir que esse seja o caso, tens de estar ciente da compatibilidade.

Primeiro, começa por instalar o tempo de execução TPU. Para a minha máquina, isto significa descarregar e descomprimir um ficheiro para executar o script de instalação:

```
$ curl -O  
https://dl.google.com/coral/edgetpu_api/edgetpu_runtime_20201204.  
zip
```

```
[...]
$ unzip edgetpu_runtime_20201204.zip
Archive: edgetpu_runtime_20201204.zip
  creating: edgetpu_runtime/
    inflating: edgetpu_runtime/install.sh
[...]
$ cd edgetpu_runtime
$ sudo bash install.sh
Password:
[...]
Installing Edge TPU runtime library [/usr/local/lib]...
Installing Edge TPU runtime library symlink [/usr/local/lib]...
```

NOTA

Estes exemplos de configuração utilizam um computador Macintosh, pelo que alguns dos métodos de instalação e dependências podem variar em relação a outros sistemas operativos. [Consulta o guia de iniciação](#) se necessitares de suporte para um computador diferente.

Agora que as dependências de tempo de execução estão instaladas no sistema, estamos prontos para experimentar o edge TPU. A equipa Coral tem um repositório útil com código Python3 que ajuda a executar a classificação de imagens com um único comando. Cria um diretório para clonar o conteúdo do repositório para configurar o espaço de trabalho para a classificação de imagens:

```
$ mkdir google-corral && cd google-corral
$ git clone https://github.com/google-corral/tflite --depth 1
[...]
Resolving deltas: 100% (4/4), done.
$ cd tflite/python/examples/classification
```

NOTA

O comando `git` usa uma flag `--depth 1`, que executa um clone superficial. Um clone superficial é desejável quando o conteúdo completo do repositório não é necessário. Como este exemplo está usando as últimas alterações do repositório, não há necessidade de executar um clone completo que contenha um histórico completo do repositório.

Para este exemplo, não executes o script *install_requirements.sh*. Primeiro, certifica-te que tens o Python3 disponível e instalado no teu sistema e usa-o para criar um novo ambiente virtual; certifica-te que após a ativação, o interpretador Python aponta para o ambiente virtual e não para o Python do sistema:

```
$ python3 -m venv venv
$ source venv/bin/activate
$ which python
~/google-
coral/tflite/python/examples/classification/venv/bin/python
```

Agora que o *virtualenv* está ativo, instala as duas dependências de biblioteca e o suporte de tempo de execução do TensorFlow Lite:

```
$ pip install numpy Pillow
$ pip install https://github.com/google-
coral/pycoral/releases/download/\
release-frogfish/tflite_runtime-2.5.0-cp38-cp38-
macosx_10_15_x86_64.whl
```

Tanto o *numpy* como o *Pillow* são fáceis de instalar na maioria dos sistemas. O problema é a ligação muito longa que se segue. Esta ligação é crucial, e tem de corresponder à tua plataforma e arquitetura. Sem essa biblioteca, não é possível interagir com o dispositivo Coral. [O guia de instalação Python para o TensorFlow Lite](#) é a fonte certa para verificares qual o link que precisas de usar para a tuaplataforma.

Agora que tens tudo instalado e pronto para realizar a classificação da imagem, corre o script *classify_image.py* para obteres o menu de ajuda. Trazer de volta o menu de ajuda, neste caso, é uma excelente maneira de verificar se todas as dependências foram instaladas e se o script funciona corretamente:

```
usage:
  classify_image.py [-h] -m MODEL -i INPUT [-l LABELS] [-k TOP_K]
  [-c COUNT]
  classify_image.py:
```

```
error: the following arguments are required: -m/--model, -i/--input
```

Como não defini nenhum sinalizador quando chamei o script, retornou um erro, mencionando que preciso passar alguns sinalizadores. Antes de começarmos a usar as outras bandeiras, precisamos de obter um modelo TensorFlow para trabalhar com uma imagem para o testar.

O [site Coral AI](#) tem uma secção de modelos onde podes ver alguns dos modelos especializados pré-treinados que tem para fazer alguma classificação de imagens. Encontra o modelo *iNat insects*, que reconhece mais de mil tipos diferentes de insectos. Descarrega o modelo *tflite* e as etiquetas.

Para este exemplo, transfere uma imagem de amostra de uma mosca comum. [A fonte original da imagem está no Pixabay](#), mas também está convenientemente acessível no [repositório do GitHub para este livro](#).

Cria diretórios para o modelo, as etiquetas e a imagem. Coloca os ficheiros necessários, respetivamente, nas respectivas diretórias. Não é necessário seguir esta ordem, mas é útil para começares a adicionar mais modelos de classificação, etiquetas e imagens mais tarde, para brincares mais com o dispositivo TPU.

É assim que a estrutura de diretórios deve ficar agora:

```
.
├── README.md
├── classify.py
├── classify_image.py
├── images
│   └── macro-1802322_640.jpg
├── install_requirements.sh
└── labels
    └── inat_insect_labels.txt
└── models
    └── mobilenet_v2_1.0_224_inat_insect_quant_edgetpu.tflite
```

3 directories, 7 files

Finalmente, podes tentar fazer operações de classificação com o dispositivo Coral. Certifica-te de que o dispositivo está ligado com o cabo USB, caso contrário obterás um longo traceback (que infelizmente não explica bem qual é o problema):

```
Traceback (most recent call last):
  File "classify_image.py", line 122, in <module>
    main()
  File "classify_image.py", line 99, in main
    interpreter = make_interpreter(args.model)
  File "classify_image.py", line 72, in make_interpreter
    tflite.load_delegate(EDGETPU_SHARED_LIB,
  File "~/lib/python3.8/site-
packages/tflite_runtime/interpreter.py",
  line 154, in load_delegate
    raise ValueError('Failed to load delegate from
{}{}'.format(
ValueError: Failed to load delegate from libedgetpu.1.dylib
```

Esse erro significa que o dispositivo está desligado. Liga-o e executa o comando de classificação:

```
$ python3 classify_image.py \
--model
models/mobilenet_v2_1.0_224_inat_insect_quant_edgetpu.tflite \
--labels labels/inat_insect_labels.txt \
--input images/macro-1802322_640.jpg
-----INFERENCE TIME-----
Note: The first inference on Edge TPU is slow because it includes
loading
the model into Edge TPU memory.
11.9ms
2.6ms
2.5ms
2.5ms
2.4ms
-----RESULTS-----
Lucilia sericata (Common Green Bottle Fly): 0.43359
```

A imagem foi classificada corretamente e a mosca comum foi detectada! Encontra outras imagens de insectos e executa novamente o comando para verificar o desempenho do modelo com diferentes entradas.

Perceção Azure

Quando este livro estava a ser escrito, a Microsoft anunciou o lançamento de uma plataforma e hardware chamados Azure Percept. Embora não tenha tido tempo suficiente para obter exemplos práticos de como tirar partido das suas características, penso que vale a pena mencionar algumas das suas funcionalidades.

Os mesmos conceitos que se aplicam ao dispositivo Coral na secção anterior e à periferia, em geral, aplicam-se aos dispositivos do Percept: permitem operações de aprendizagem automática sem descontinuidades na periferia.

Em primeiro lugar, é importante sublinhar que, embora os produtos Percept sejam maioritariamente anunciados como peças de hardware, o Azure Percept é uma plataforma completa para fazer computação periférica, desde os próprios dispositivos até à implementação, formação e gestão no Azure. Há também suporte para as principais plataformas de IA, como ONNX e TensorFlow, facilitando a experimentação com modelos pré-construídos.

Uma desvantagem do hardware do Azure Percept em comparação com os dispositivos Coral é que é muito mais caro, tornando mais difícil comprar um dos seus pacotes para experimentar a nova tecnologia. Como sempre, a Microsoft fez um excelente trabalho **ao documentar e adicionar uma boa quantidade de contexto e exemplos** que vale a pena explorar se estiveres interessado.

TFHub

Um excelente recurso para encontrar modelos do TensorFlow é o **TensorFlow Hub**. O hub é um repositório de milhares de modelos pré-treinados prontos para serem usados. No entanto, para a TPU do Coral Edge, nem todos os modelos funcionarão. Como a TPU tem instruções separadas específicas para o dispositivo, um modelo precisa de ser explicitamente compilado para ele.

Agora que pode executar classificações com o dispositivo USB Coral, pode utilizar o TFHub para encontrar outros modelos pré-treinados para trabalhar. No hub, está disponível um formato de modelo Coral; clica nele para aceder aos modelos prontos a utilizar para a TPU, como mostra a [Figura 3-2](#).

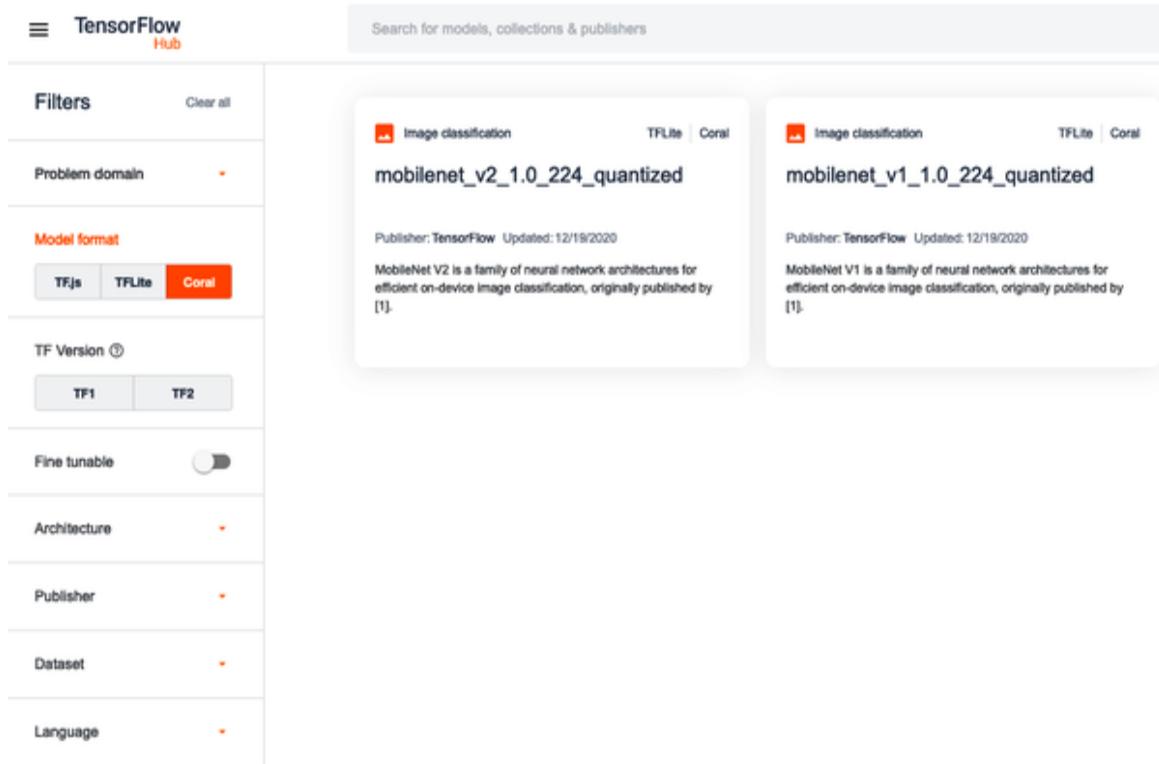


Figura 3-2. Modelos do TFHub Coral

Seleciona o modelo *MobileNet Quantized V2* para transferência. Este modelo pode detetar mais de mil objectos a partir de imagens. Os exemplos anteriores que utilizam o Coral requerem as etiquetas e o modelo, por isso certifica-te de que também os descarregas.

NOTA

Quando estes modelos são apresentados no site TFHub, estão disponíveis vários formatos diferentes. Certifica-te de que verificas o formato do modelo que estás a receber e que (neste caso) é compatível com o dispositivo Coral.

Transposição de modelos não TPU

Podes descobrir que o modelo de que precisas está disponível em algumas situações, mas não está compilado para o dispositivo TPU que tens. O TPU Coral Edge tem um compilador disponível, mas não pode ser instalado em todas as plataformas, como acontece com as dependências de tempo de execução. Quando estas situações surgem, tens de ser criativo nas soluções e tentar sempre encontrar a automação dentro de quaisquer soluções possíveis. A documentação do compilador requer uma distribuição Debian ou Ubuntu Linux, e as instruções sobre como configurar tudo para o compilador estão ligadas a essa distribuição em particular.

No meu caso, estou a trabalhar a partir de um computador Apple e não tenho outros computadores a correr Linux. O que eu *tenho* é um tempo de execução de container instalado localmente no qual eu posso executar qualquer imagem de qualquer distro com alguns comandos. Já falamos sobre como começar a usar containers, como executá-los e como criá-los. E este é o caso de uso perfeito para criar um novo container baseado no Debian com tudo instalado para o compilador resolver este problema.

Agora que entendemos o problema e temos uma solução em mente com containers, crie um novo *Dockerfile* para construir uma imagem de container para o compilador:

```
FROM debian:stable

RUN apt-get update && apt install -yq curl build-essential gnupg

RUN curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | \
\
    apt-key add -

RUN \
    echo "deb https://packages.cloud.google.com/apt coral-edgetpu-\
stable main" | \
    tee /etc/apt/sources.list.d/coral-edgetpu.list

RUN apt-get update && apt-get install -yq edgetpu-compiler

CMD ["/bin/bash"]
```

Com o *Dockerfile* recém-criado, cria uma nova imagem para executar o compilador:

```
$ docker build -t tpu-compiler .
[+] Building 15.5s (10/10) FINISHED
=> => transferring dockerfile: 408B
[...]
=> [5/5] RUN apt update && apt install -yq edgetpu-compiler
=> exporting to image
=> => exporting layers
=> => writing image

sha256:08078f8d7f7dd9002bd5a1377f24ad0d9dbf8f7b45c961232cf2cbf8f9
f946e4
=> => naming to docker.io/library/tpu-compiler
```

Identifiquei **um modelo** que quero utilizar com o compilador TPU, mas não vem compilado para ele.

NOTA

Apenas os modelos pré-compilados para o TensorFlow Lite e quantizados funcionarão com o compilador. Certifica-te de que os modelos são *tflite* e *quantizados* antes de os transferires para os converteres com o compilador.

Descarrega o modelo localmente. Neste caso, utilizo a linha de comando para o guardar no diretório de trabalho atual:

```
$ wget -O mobilenet_v1_50_160_quantized.tflite \
https://tfhub.dev/tensorflow/lite-model/\
mobilenet_v1_0.50_160_quantized/1/default/1?lite-format=tflite

Resolving tfhub.dev (tfhub.dev)... 108.177.122.101,
108.177.122.113, ...
Connecting to tfhub.dev (tfhub.dev)|108.177.122.101|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 1364512 (1.3M) [application/octet-stream]
Saving to: 'mobilenet_v1_50_160_quantized.tflite'

(31.8 MB/s) - 'mobilenet_v1_50_160_quantized.tflite' saved
```

[1364512/1364512]

```
$ ls  
mobilenet_v1_50_160_quantized.tflite
```

Embora eu tenha utilizado a linha de comandos, também podes descarregar o modelo indo [ao modelo no sítio Web](#). Certifica-te de que transferes o ficheiro para o diretório de trabalho atual para os passos seguintes.

Precisamos de colocar o modelo descarregado no contentor e depois copiar os ficheiros localmente. O Docker torna essa tarefa um pouco mais gerenciável usando um *bind mount*. Esta operação de montagem ligará um caminho da minha máquina ao contentor, partilhando efetivamente qualquer coisa que eu tenha no contentor. Isso também funciona muito bem para arquivos criados no contêiner, e eu preciso deles de volta na máquina local. Esses ficheiros criados no contentor aparecerão automaticamente no meu ambiente local.

Inicia o contentor com o bind mount:

```
$ docker run -it -v ${PWD}:/models tpu-compiler  
root@5125dcd1da4b:/# cd models  
root@5125dcd1da4b:/models# ls  
mobilenet_v1_50_160_quantized.tflite
```

Há um par de coisas a acontecer com o comando anterior. Primeiro, estou a usar **PWD** para indicar que o diretório de trabalho atual, onde existe o ficheiro *mobilenet_v1_50_160_quantized.tflite*, é o que quero no contentor. O caminho de destino dentro do contentor é */models*. E, por último, estou a utilizar o contentor construído com a etiqueta **tpu-compiler** para especificar o contentor de que necessito. Se usaste uma etiqueta diferente ao construir a imagem, terás de atualizar essa parte do comando. Depois de iniciar o contêiner, mudo os diretórios para */models*, listo o conteúdo do diretório e encontro o modelo baixado na minha máquina local. O ambiente agora está pronto para usar o compilador.

Verifica se o compilador funciona chamando o seu menu de ajuda:

```
$ edgetpu_compiler --help
Edge TPU Compiler version 15.0.340273435
```

```
Usage:
edgetpu_compiler [options] model...
```

Em seguida, executa o compilador contra o modelo quantizado:

```
$ edgetpu_compiler mobilenet_v1_50_160_quantized.tflite
Edge TPU Compiler version 15.0.340273435

Model compiled successfully in 787 ms.

Input model: mobilenet_v1_50_160_quantized.tflite
Input size: 1.30MiB
Output model: mobilenet_v1_50_160_quantized_edgetpu.tflite
Output size: 1.54MiB
Number of Edge TPU subgraphs: 1
Total number of operations: 31
Operation log: mobilenet_v1_50_160_quantized_edgetpu.log
See the operation log file for individual operation details.
```

A operação demorou menos de um segundo a ser executada e produziu alguns ficheiros, incluindo o modelo recentemente compilado(*mobilenet_v1_50_160_quantized_edgetpu.tflite*) que podes agora utilizar com o dispositivo de borda.

Finalmente, sai do contentor, volta à máquina local e lista o conteúdo do diretório:

```
$ ls
mobilenet_v1_50_160_quantized.tflite
mobilenet_v1_50_160_quantized_edgetpu.log
mobilenet_v1_50_160_quantized_edgetpu.tflite
```

Esta é uma solução prática para a necessidade de um sistema operativo para uma ferramenta. Agora que esse contêiner pode compilar modelos para o dispositivo de borda, ele pode ser automatizado ainda mais, portando os modelos necessários com algumas linhas em um script. Lembra-te de que há algumas suposições feitas no processo e que tens de garantir que todas elas são precisas no momento da compilação. Caso contrário, receberás

erros do compilador. Este processo é um exemplo de tentativa de usar um modelo não quantizado com o compilador:

```
$ edgetpu_compiler vision_classifier_fungi_mobile_v1.tflite
Edge TPU Compiler version 15.0.340273435
Invalid model: vision_classifier_fungi_mobile_v1.tflite
Model not quantized
```

Contentores para sistemas ML geridos

No centro dos fluxos de trabalho avançados de MLOps da próxima geração estão os sistemas de ML geridos, como o AWS SageMaker, o Azure ML Studio e o Vertex AI da Google. Todos estes sistemas são construídos em cima de contentores. Os contentores são um ingrediente secreto para os MLOps. Sem a contentorização, é muito mais difícil desenvolver e utilizar tecnologias como o AWS SageMaker. Na [Figura 3-3](#), repara que o Registo de Contentores EC2 é o local onde se encontram a imagem do código de inferência e o código de treino.

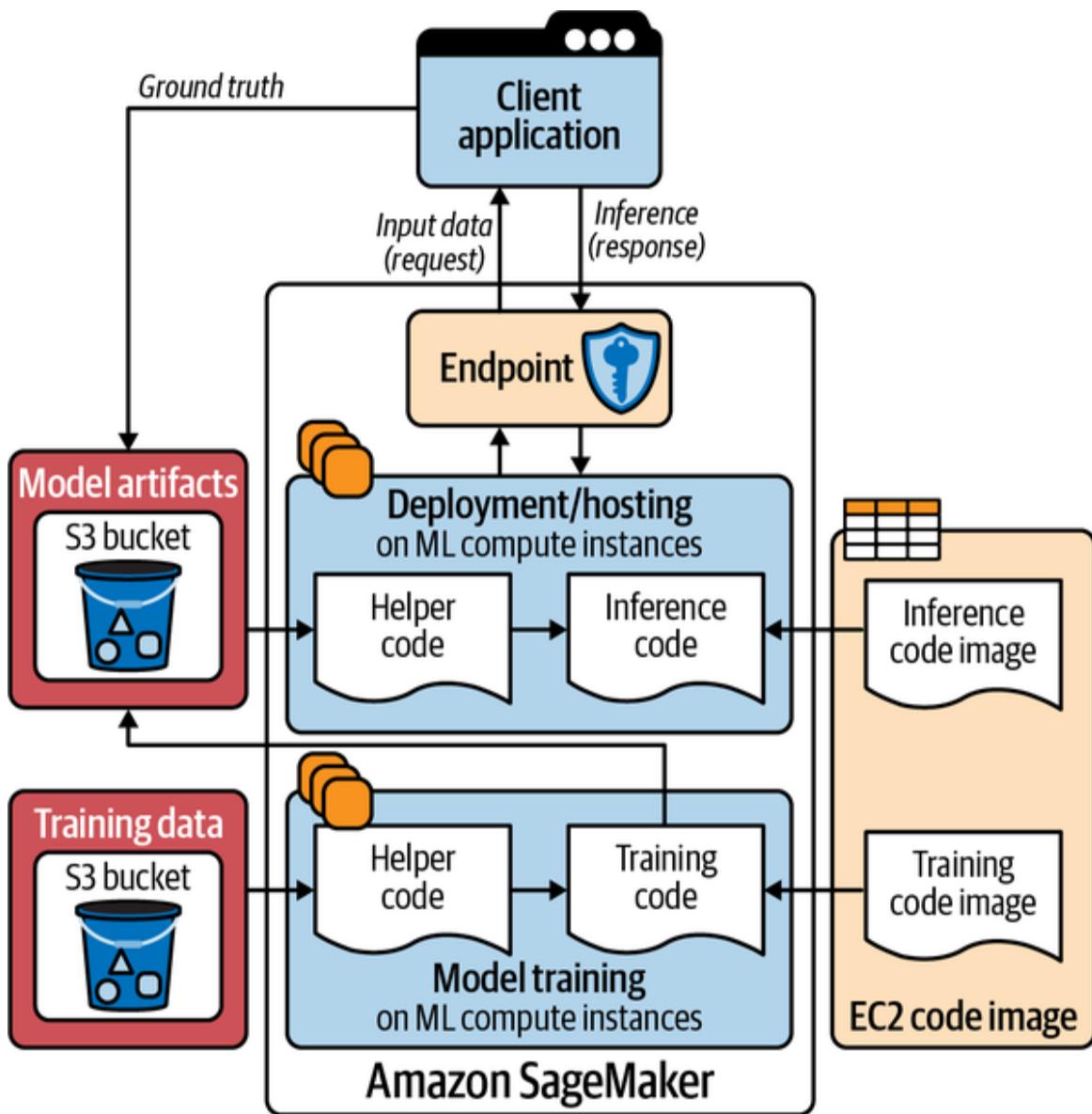


Figura 3-3. Contentores SageMaker

Esse processo é extremamente importante porque permite que as práticas recomendadas do DevOps sejam incorporadas à criação dessas imagens - entre elas, as mais importantes são as integrações contínuas e a entrega contínua. Os contentores aumentam a qualidade de toda a arquitetura de ML reduzindo a complexidade, uma vez que as imagens já estão "cozinhadas". A capacidade intelectual pode ser transferida para outros problemas, como desvio de dados, análise do armazenamento de recursos para candidatos adequados a um modelo mais recente ou avaliação se o novo modelo atende às necessidades do cliente.

Contentores na monetização de MLOps

Monetizar os MLOps é outro problema crucial tanto para startups como para grandes empresas. Os contentores desempenham mais uma vez um papel importante! No caso do SageMaker, usa um algoritmo ou um modelo vendido no AWS Marketplace, como mostrado na [Figura 3-4](#). Eles são o modo de entrega do produto vendido.

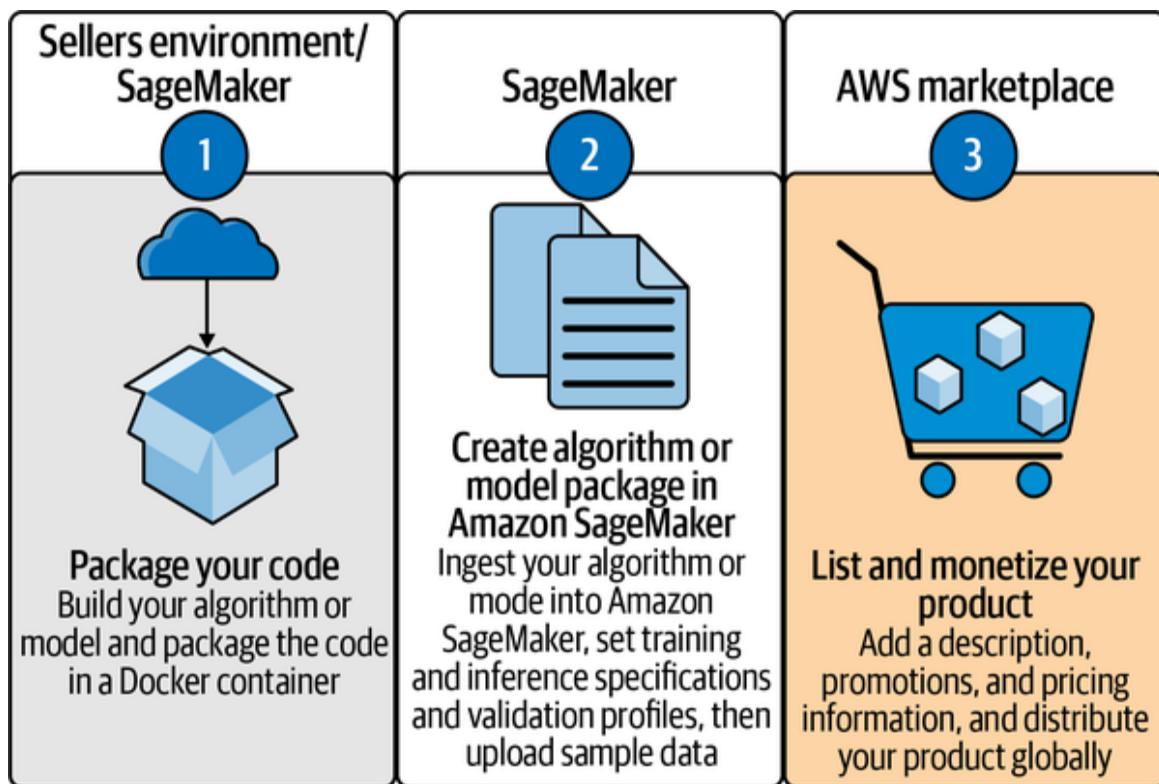


Figura 3-4. Fluxo de trabalho do vendedor do SageMaker

A vantagem de um contentor como produto é que é vendido da mesma forma que outros produtos vendidos numa loja física, como manteiga de amendoim, farinha ou leite. No cenário em que uma empresa decide produzir manteiga de amendoim orgânica de alta qualidade, pode querer concentrar-se estritamente no fabrico da manteiga de amendoim e não na construção de uma rede de lojas para vender a manteiga de amendoim.

Da mesma forma, nas empresas que procuram rentabilizar a aprendizagem automática, o contentor é um pacote ideal para fornecer modelos e algoritmos aos clientes. Em seguida, vamos ver como podes construir uma vez e executar muitas com contentores.

Constrói uma vez, executa muitos fluxos de trabalho MLOps

Em última análise, um processo de contentor para MLOps culmina em muitas opções ricas tanto para o produto como para a engenharia. Na [Figura 3-5](#), vê que um contentor é um pacote ideal para rentabilizar a propriedade intelectual de uma perspetiva de produto. Da mesma forma, do ponto de vista da engenharia, um contentor pode servir previsões, fazer formação ou implementar num dispositivo de ponta como um TPU Coral ou um iPhone da Apple.

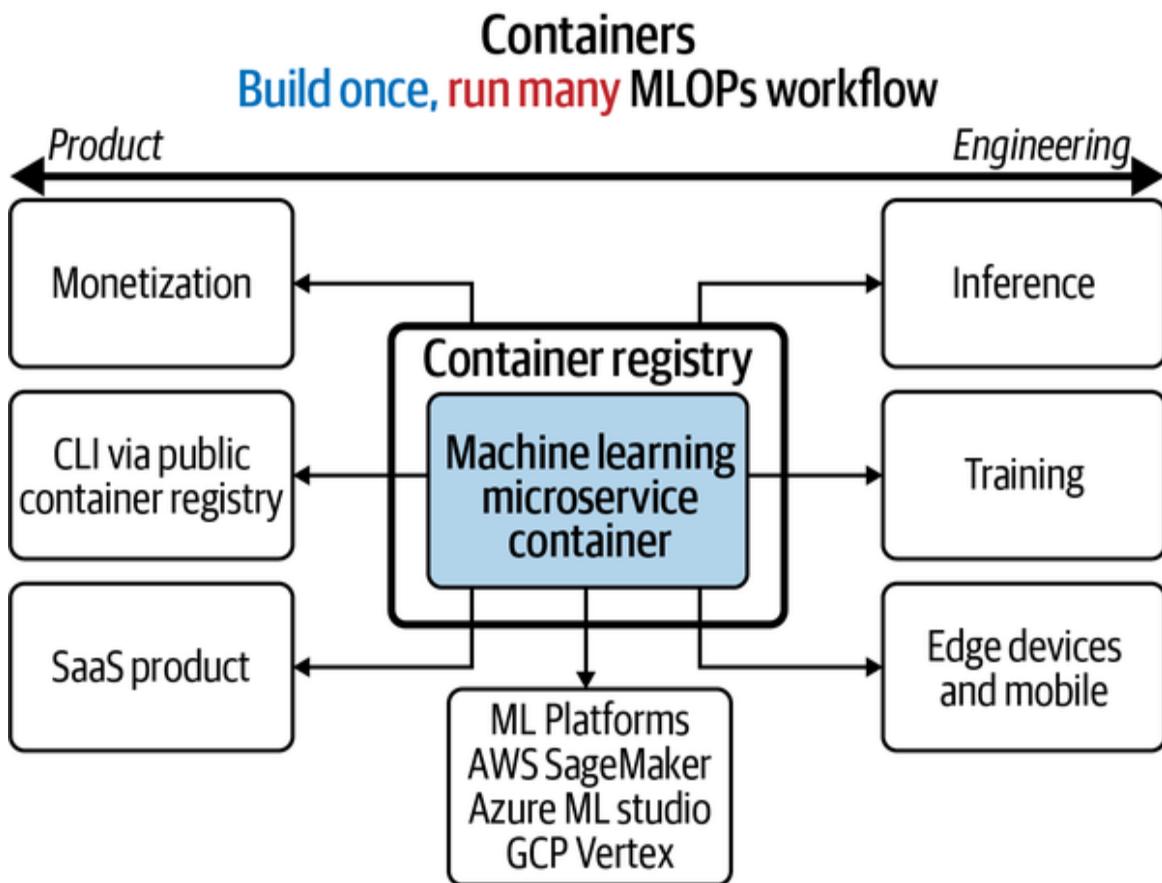


Figura 3-5. Constrói uma vez, executa muitos MLOps

Os MLOps e a tecnologia de contentores são complementares na medida em que os contentores ajudam a fornecer valor comercial. As metodologias MLOps são construídas diretamente sobre essa tecnologia para otimizar a produtividade e agregar valor. A seguir, vamos encerrar o capítulo e resumir os aspectos essenciais dos contêineres para MLOps.

Conclusão

Ao operacionalizar os modelos de ML, depara frequentemente com muitas possibilidades diferentes de implementação. Está a tornar-se bastante comum ver modelos a serem implementados em telemóveis e outros (pequenos) dispositivos que podem ser ligados a qualquer computador com uma porta USB. Os problemas que a inferência de borda fornece (como acesso offline, remoto e rápido) podem ser transformacionais, especificamente para regiões remotas sem acesso a uma fonte confiável de energia e rede. Semelhante aos dispositivos de ponta, a contentorização permite uma reprodução mais rápida e fiável dos ambientes. A reprodução de ambientes de máquinas era um problema difícil de resolver há apenas alguns anos. A contentorização é excepcionalmente relevante nesse caso. O dimensionamento rápido de recursos e a transição de ambientes de implementação de fornecedores de Cloud, ou mesmo a transferência de cargas de trabalho do local para a Cloud, é muito mais fácil de realizar com contentores.

Com isso coberto, nosso próximo capítulo mergulha no processo de entrega contínua para modelos de aprendizado de máquina.

Exercícios

- Recompila um modelo para funcionar com a TPU Coral Edge da TFHub.
- Utiliza o modelo MobileNet V2 para realizar inferências sobre outros objectos e obter resultados precisos.
- Cria uma nova imagem de contêiner, com base no exemplo do Flask, que serve um modelo e fornece exemplos em uma solicitação GET para interagir com o modelo. Cria outro ponto de extremidade que forneça metadados úteis sobre o modelo.
- Publica a imagem recém-criada num registo de contentores como o Docker Hub.

Questões para discussão sobre pensamento crítico

- Seria possível utilizar um contentor para efetuar previsões online com um dispositivo TPU de ponta como o Coral? Como? ou Por que não?
- O que é um tempo de execução de contentor e qual é a sua relação com o Docker?
- Cita três boas práticas ao criar um Dockerfile.
- Quais são os dois conceitos críticos de DevOps mencionados neste capítulo? Porque é que são úteis?
- Cria uma definição, nas tuas próprias palavras, do que é a "borda". Dá alguns exemplos de ML que possam ser aplicados.

Capítulo 4. Entrega contínua paramodelos de aprendizado de máquina

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

Será que é realmente uma triste verdade que a filosofia natural (aquilo a que agora chamamos ciência) se afastou tanto das suas origens que só deixou atrás de si papirólogos - pessoas que recebem papel, põem papel fora e, enquanto lêem e escrevem assiduamente, evitam seriamente o tangível? Será que consideram que o contacto direto com os dados tem um valor negativo? Será que, tal como um saloio do romance Tobacco Road, se orgulham da sua ignorância?

—Dr. Joseph Bogen

Como atleta profissional, lidava frequentemente com lesões. As lesões têm todo o tipo de níveis de gravidade. Por vezes, era algo menor, como uma ligeira contratura no meu tendão esquerdo depois de intensos treinos de obstáculos. Outras vezes, são mais graves, como uma dor lombar insuportável. Os atletas de alto rendimento não se podem dar ao luxo de ter dias de folga a meio da época. Se o plano é treinar sete dias por semana, é fundamental que cumbras esses sete dias. Falhar um dia tem repercussões graves que podem diminuir (ou anular completamente) os treinos até esse momento. Os treinos são como empurrar um carrinho de mão para cima, e faltar a um treino significa dar um passo para o lado, deixando o carrinho de mão descer a encosta. A repercussão dessa ação é que terás de voltar atrás e

pegar no carrinho de mão para o empurrar novamente para cima. Não podes faltar aos treinos.

Se estiveres lesionado e não puderes fazer exercício, recuperar a forma o mais rapidamente possível é uma prioridade *tão importante como encontrar exercícios alternativos*. Isto significa que, se te dói o tendão e não podes correr, vê se podes ir à piscina e manter o plano de cardio. As repetições de colina não são possíveis amanhã porque partiste um dedo do pé? Então experimenta subir para a bicicleta e fazer essas mesmas subidas. As lesões requerem uma estratégia de guerra; desistir e desistir não é uma opção, mas se tiveres de recuar, então considera primeiro recuar o *menospossível*. Se não podemos disparar canhões, trazemos a cavalaria. Há sempre uma opção, e a criatividade é tão importante como tentar recuperar totalmente.

A recuperação também requer estratégia, mas mais do que estratégia, requer uma avaliação constante. Uma vez que continuas a fazer o máximo de exercício possível com uma lesão, é essencial avaliar se a lesão está a piorar. Se te metes na bicicleta para compensar o facto de não poderes correr, tens de estar muito atento se a bicicleta está a agravar a lesão. A avaliação constante das lesões é um algoritmo bastante simplista:

1. A primeira coisa que fazes todos os dias é avaliar se a lesão está igual, pior ou melhor do que no dia anterior.
2. Se for pior, então faz mudanças para evitar os treinos anteriores ou alterá-los. Estes podem estar a prejudicar a recuperação.
3. Se for igual, compara a lesão com a da semana passada ou mesmo do mês passado. Faz a seguinte pergunta : "*Sinto-me igual, pior ou melhor do que na semana passada?*"
4. Por fim, se te sentires melhor, isso reforça fortemente que a estratégia atual está a funcionar e que deves continuar até estares totalmente recuperado.

Com algumas lesões, tive de avaliar com maior frequência (em vez de esperar até à manhã seguinte). O resultado de uma avaliação constante foi a chave para a recuperação. Em alguns casos, tive de avaliar se uma ação

específica me estava a prejudicar. Uma vez, parti um dedo do pé (bati com ele no canto de uma estante) e pensei imediatamente numa estratégia: posso andar? Sinto dores se correr? A resposta a todas estas perguntas foi um retumbante "sim". Tentei nadar nessa tarde. Durante as semanas seguintes, verifiquei constantemente se era possível andar sem dores. A dor não é um inimigo. É o indicador que te ajuda a decidir se deves continuar a fazer o que estás a fazer ou parar e repensar a estratégia atual.

A avaliação constante, a realização de alterações e a adaptação ao feedback, bem como a aplicação de novas estratégias para alcançar o sucesso, são exatamente o objetivo da integração contínua (CI) e da entrega contínua (CD). Mesmo hoje em dia, onde a informação sobre estratégias de implementação robustas está facilmente disponível, é frequente encontrar empresas sem testes ou com uma estratégia de testes deficiente para garantir que um produto está pronto para um novo lançamento ou mesmo lançamentos que demoram semanas (e meses!). Lembro-me de tentar cortar uma nova versão de um grande projeto de código aberto, e houve alturas em que demorou quase uma semana. Pior ainda, o líder de Garantia de Qualidade (QA) enviava e-mails a todos os líderes de equipa e perguntava-lhes se se sentiam prontos para um lançamento ou se queriam mais alterações.

Enviar e-mails e esperar por diferentes respostas não é uma forma simples de lançar software. É propenso a erros e altamente inconsistente. O ciclo de feedback que as plataformas e os passos de CI/CD proporcionam a ti e à tua equipa é inestimável. Se encontrares um problema, deves automatizá-lo e fazer com que não seja um problema para a próxima versão. A avaliação constante, tal como as lesões dos atletas de alto desempenho, é um pilar fundamental do DevOps e absolutamente crítico para uma operacionalização bem sucedida da aprendizagem automática.

Gosto da descrição de contínuo como persistência ou recorrência de um processo. CI/CD são geralmente mencionados juntos quando se fala sobre o sistema que constrói, verifica e implanta artefatos. Neste capítulo, detalharei o aspeto de um processo robusto e como podes ativar várias

estratégias para implementar (ou melhorar) um pipeline para enviar modelos para produção.

Embalagem para modelos ML

Não faz muito tempo que ouvi falar sobre empacotamento de modelos ML pela primeira vez. Se nunca ouviste falar em empacotar modelos antes, não faz mal - isto é tudo bastante recente, e *empacotar* aqui não significa um tipo especial de pacote de sistema operativo como um ficheiro RPM (Red Hat Package Manager) ou DEB (Debian Package) com diretivas especiais para empacotamento e distribuição. Tudo isso significa colocar um modelo em um contêiner para aproveitar os processos em contêineres para ajudar a compartilhar, distribuir e facilitar a implantação. Já descrevi a contentorização em pormenor em "[Contentores](#)" e porque faz sentido utilizá-los para operacionalizar a aprendizagem automática em vez de utilizar outras estratégias como as máquinas virtuais, mas vale a pena reiterar que a capacidade de experimentar rapidamente um modelo a partir de um contentor, independentemente do sistema operativo, é um cenário de sonho tornado realidade.

Há três características do acondicionamento de modelos ML em contentores que é importante analisar:

- Desde que o tempo de execução do contentor esteja instalado, é fácil executar um contentor localmente.
- Existem muitas opções para implementar um contentor na Cloud, com a capacidade de aumentar ou diminuir a escala conforme necessário.
- Outros podem experimentá-lo rapidamente com facilidade e interagir com o contentor.

As vantagens destas características são que a manutenção se torna menos complicada e a depuração de um modelo não funcional localmente (ou mesmo numa oferta Cloud) pode ser tão simples como alguns comandos

num terminal. Quanto mais complicada for a estratégia de implantação, mais difícil será solucionar problemas e investigar possíveis problemas.

Para esta secção, vou utilizar um modelo ONNX e empacotá-lo num contentor que serve uma aplicação Flask que executa a previsão. Vou usar o modelo ONNX **RoBERTa-SequenceClassification**, que está muito bem documentado. Depois de criares um novo repositório Git, o primeiro passo é descobrir as dependências necessárias. Depois de criar o repositório Git, começa por adicionar o seguinte ficheiro *requirements.txt*:

```
simpletransformers==0.4.0
tensorboardX==1.9
transformers==2.1.0
flask==1.1.2
torch==1.7.1
onnxruntime==1.6.0
```

Em seguida, cria um Dockerfile que instala tudo no contentor:

```
FROM python:3.8

COPY ./requirements.txt /webapp/requirements.txt

WORKDIR /webapp

RUN pip install -r requirements.txt

COPY webapp/* /webapp

ENTRYPOINT [ "python" ]

CMD [ "app.py" ]
```

O Dockerfile copia o arquivo de requisitos, cria um diretório *webapp* e copia o código do aplicativo em um único arquivo *app.py*. Cria o arquivo *webapp/app.py* para executar a análise de sentimentos. Começa por adicionar as importações e tudo o que é necessário para criar uma sessão de tempo de execução ONNX:

```
from flask import Flask, request, jsonify
import torch
```

```

import numpy as np
from transformers import RobertaTokenizer
import onnxruntime

app = Flask(__name__)
tokenizer = RobertaTokenizer.from_pretrained("roberta-base")
session = onnxruntime.InferenceSession(
    "roberta-sequence-classification-9.onnx")

```

Esta primeira parte do arquivo cria o aplicativo Flask, define o tokenizador a ser usado com o modelo e, finalmente, inicializa uma sessão de tempo de execução ONNX que requer a passagem de um caminho para o modelo. Há algumas importações que ainda não são usadas. Vais utilizá-las a seguir quando adicionares a rota Flask para ativar a inferência em tempo real:

```

@app.route("/predict", methods=["POST"])
def predict():
    input_ids = torch.tensor(
        tokenizer.encode(request.json[0],
add_special_tokens=True)
    ).unsqueeze(0)

    if input_ids.requires_grad:
        numpy_func = input_ids.detach().cpu().numpy()
    else:
        numpy_func = input_ids.cpu().numpy()

    inputs = {session.get_inputs()[0].name:
numpy_func(input_ids)}
    out = session.run(None, inputs)

    result = np.argmax(out)

    return jsonify({"positive": bool(result)})

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)

```

A função `predict()` é uma rota Flask que ativa o URL `/predict` quando a aplicação está em execução. A função só permite os métodos HTTP POST. Ainda não há descrição das entradas e saídas de exemplo porque falta uma parte crítica da aplicação: o modelo ONNX ainda não existe. Descarrega

localmente o modelo ONNX **RoBERTa-SequenceClassification** e coloca-o na raiz do projeto. A estrutura final do projeto deve ter este aspeto:

```
.  
└── Dockerfile  
└── requirements.txt  
└── roberta-sequence-classification-9.onnx  
└── webapp  
    └── app.py
```

1 directory, 4 files

Uma última coisa que falta antes de construir o contentor é que não há instruções para copiar o modelo para o contentor. O arquivo *app.py* querer que o modelo *roberta-sequence-classification-9.onnx* exista no diretório */webapp*. Atualiza o *Dockerfile* para refletir isso:

```
COPY roberta-sequence-classification-9.onnx /webapp
```

Agora o projeto tem tudo o que é necessário para que possas construir o contentor e executar a aplicação. Antes de construir o contêiner, vamos verificar se tudo está funcionando. Cria um novo ambiente virtual, ativa-o e instala todas as dependências:

```
$ python3 -m venv venv  
$ source venv/bin/activate  
$ pip install -r requirements.txt
```

O modelo ONNX existe na raiz do projeto, mas a aplicação querer-o na diretoria/*webapp*, por isso move-o para dentro dessa diretoria para que a aplicação Flask não se queixe (este passo extra não é necessário quando o contentor é executado):

```
$ mv roberta-sequence-classification-9.onnx webapp/
```

Agora executa a aplicação localmente invocando o ficheiro *app.py* com Python:

```
$ cd webapp
$ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Em seguida, a aplicação está pronta para consumir pedidos HTTP. Até agora, não mostrei quais são as entradas esperadas. Estes vão ser pedidos formatados em JSON com respostas JSON. Usa o programa *curl* para enviar um payload de amostra para detetar sentimentos:

```
$ curl -X POST -H "Content-Type: application/JSON" \
--data '[{"Containers are more or less interesting"}' \
http://0.0.0.0:5000/predict

{
  "positive": false
}

$ curl -X POST -H "Content-Type: application/json" \
--data '[{"MLOps is critical for robustness"}' \
http://0.0.0.0:5000/predict

{
  "positive": true
}
```

O pedido JSON é uma matriz com uma única cadeia de caracteres, e a resposta é um objeto JSON com uma chave "positiva" que indica o sentimento da frase. Agora que verificaste que a aplicação funciona e que a previsão em tempo real está a funcionar corretamente, é altura de criar o contentor localmente para verificar se tudo funciona. Cria o contentor e marca-o com algo significativo:

```
$ docker build -t alfredodeza/roberta .
[+] Building 185.3s (11/11) FINISHED
=> [internal] load metadata for docker.io/library/python:3.8
=> CACHED [1/6] FROM docker.io/library/python:3.8
=> [2/6] COPY ./requirements.txt /webapp/requirements.txt
```

```
=> [3/6] WORKDIR /webapp
=> [4/6] RUN pip install -r requirements.txt
=> [5/6] COPY webapp/* /webapp
=> [6/6] COPY roberta-sequence-classification-9.onnx /webapp
=> exporting to image
=> => naming to docker.io/alfredodeza/roberta
```

Agora executa o contentor localmente para interagir com ele da mesma forma que quando executas a aplicação diretamente com Python. Lembra-te de mapear as portas do contentor para o localhost:

```
$ docker run -it -p 5000:5000 --rm alfredodeza/roberta
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Envia um pedido HTTP da mesma forma que antes. Podes voltar a utilizar o programa *curl*:

```
$ curl -X POST -H "Content-Type: application/json" \
--data '[{"espresso is too strong"}' \
http://0.0.0.0:5000/predict

{
  "positive": false
}
```

Passámos por muitas etapas para empacotar um modelo e colocá-lo dentro de um contentor. Algumas dessas etapas podem parecer complicadas, mas processos desafiadores são uma oportunidade perfeita para automatizar e aproveitar os padrões de entrega contínua. Na próxima secção, vou automatizar tudo isto utilizando a entrega contínua e publicando este contentor num registo de contentores que qualquer pessoa pode consumir.

Infraestrutura como código para entrega contínua de modelos de ML

Recentemente, no trabalho, vi que existiam algumas imagens de contentores de teste num repositório público, que eram amplamente utilizadas pela infraestrutura de testes. Ter imagens alojadas num registo de contentores (como o Docker Hub) já é um grande passo na direção certa para compilações repetíveis e testes fiáveis. Encontrei um problema com uma das bibliotecas utilizadas num contentor que precisava de uma atualização, por isso procurei os ficheiros utilizados para criar estes contentores de teste. Não os encontrei em lado nenhum. Em algum momento, um engenheiro criou-os localmente e carregou as imagens para o registo. Isto representou um grande problema porque eu não podia fazer uma simples alteração à imagem, uma vez que os ficheiros necessários para construir a imagem estavam perdidos.

Os desenvolvedores de containers experientes podem encontrar uma maneira de obter a maioria dos arquivos (se não todos) para reconstruir o container, mas isso não vem ao caso. Um passo adiante nessa situação problemática é criar uma automação que possa construir automaticamente esses contêineres a partir de arquivos de origem conhecidos, incluindo o *Dockerfile*. Reconstruir ou resolver o problema para atualizar o contentor e voltar a carregar para o registo é como encontrar velas e lanternas num apagão, em vez de ter um gerador que arranca automaticamente assim que a energia acaba. Sê altamente analítico quando acontecerem situações como a que acabei de descrever. Em vez de apontares o dedo e culpar os outros, usa-as como uma oportunidade para melhorares o processo com automação.

O mesmo problema acontece na aprendizagem automática. Temos tendência para nos habituarmos facilmente a que as coisas sejam manuais (e complexas!), mas há sempre uma oportunidade para automatizar. Esta secção não irá rever todos os passos necessários para a contentorização (já abordados em "[Contentores](#)"), mas irei abordar os detalhes necessários para automatizar tudo. Vamos supor que estamos numa situação semelhante à que acabei de descrever e que alguém criou um contentor com um modelo

que vive no Docker Hub. Ninguém sabe como o modelo treinado foi parar ao contentor; não existem documentos e são necessárias actualizações. Vamos acrescentar uma ligeira complexidade: o modelo não está em nenhum repositório para ser encontrado, mas vive no Azure como um modelo registado. Vamos fazer alguma automação para resolver este problema.

AVISO

Pode ser tentador adicionar modelos a um repositório do GitHub. Embora isso seja certamente possível, o GitHub tem (no momento em que este texto foi escrito) um limite de arquivo rígido de 100 MB. Se o modelo que estás a tentar empacotar estiver perto desse tamanho, talvez não consigas adicioná-lo ao repositório. Além disso, o Git (o sistema de controlo de versões) não foi concebido para lidar com o controlo de versões de ficheiros binários e tem o efeito secundário de criar repositórios enormes por causa disso.

No cenário do problema atual, o modelo está disponível na plataforma de ML do Azure e previamente registado. Como ainda não tinha um, registei rapidamente o [RoBERTa-SequenceClassification](#) utilizando o Azure ML Studio. Clica na secção Models (Modelos) e, em seguida, em "Register model" (Registrar modelo), conforme apresentado na [Figura 4-1](#).

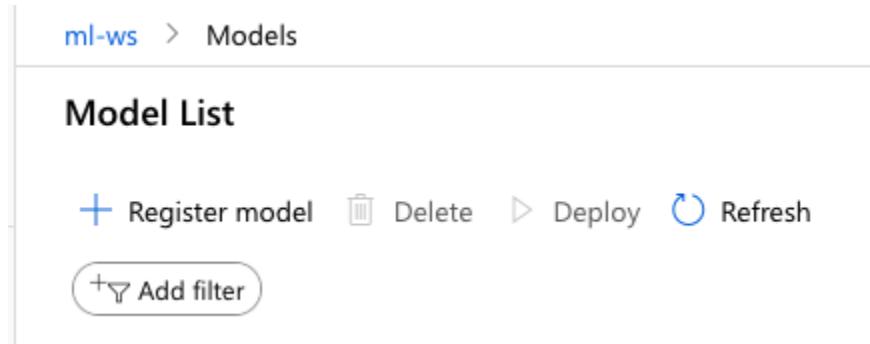


Figura 4-1. Menu de registo do modelo Azure

Preenche o formulário mostrado na [Figura 4-2](#) com os detalhes necessários. No meu caso, descarreguei o modelo localmente e preciso de o carregar utilizando o campo "Upload file".

Register a model X

Name * eye icon

Description

Model framework *
 dropdown arrow *

Framework version *

Model file or folder *
 Upload file Upload folder
 * Browse

Add tags
 : Add tag

Add properties
 : Add property

Figura 4-2. Formulário de registo de modelo do Azure

NOTA

Se quiseres saber mais sobre como registar um modelo no Azure, explico como o fazer com o Python SDK em "["Registrar Modelos"](#)".

Agora que o modelo pré-treinado está no Azure, vamos reutilizar o mesmo projeto do "["Packaging for ML Models"](#)". Todo o trabalho pesado para

realizar a inferência ao vivo (local) está feito, portanto, cria um novo repositório do GitHub e adiciona o conteúdo do projeto , *exceto* o modelo ONNX. Lembra-te de que existe um limite de tamanho para os ficheiros no GitHub, pelo que não é possível adicionar o modelo ONNX ao repositório do GitHub. Cria um ficheiro `.gitignore` para ignorar o modelo e evitar adicioná-lo por engano:

`*onnx`

Depois de fazeres o push do conteúdo do repositório Git sem o modelo ONNX, estamos prontos para começar a automatizar a criação e a entrega do modelo. Para isso, vamos usar o GitHub Actions, que nos permite criar um fluxo de trabalho de entrega contínua num ficheiro YAML que é acionado quando são cumpridas condições configuráveis. A ideia é que sempre que o repositório tenha uma alteração no ramo principal, a plataforma puxe o modelo registado do Azure, crie o contentor e, por fim, faça push para um registo de contentores. Começa por criar um diretório `.github/workflows/` na raiz do teu projeto e, em seguida, adiciona um `main.yml` com o seguinte aspeto:

```
name: Build and package RoBERTa-sequencing to Dockerhub

on:
  # Triggers the workflow on push or pull request events for the
  main branch
  push:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
workflow_dispatch:
```

A configuração até agora não faz mais nada para além de definir a ação. Podes definir qualquer número de tarefas e, neste caso, definimos uma tarefa *de construção* que irá juntar tudo. Acrescenta o seguinte ao ficheiro `main.yml` que criaste anteriormente:

```
jobs:
  build:
```

```

runs-on: ubuntu-latest
steps:
  - uses: actions/checkout@v2

  - name: Authenticate with Azure
    uses: azure/login@v1
    with:
      creds: ${{secrets.AZURE_CREDENTIALS}}

  - name: set auto-install of extensions
    run: az config set
        extension.use_dynamic_install=yes_without_prompt

  - name: attach workspace
    run: az ml folder attach -w "ml-ws" -g "practical-mlops"

  - name: retrieve the model
    run: az ml model download -t "." --model-id "roberta-
sequence:1"

  - name: build flask-app container
    uses: docker/build-push-action@v2
    with:
      context: ../
      file: ./Dockerfile
      push: false
      tags: alfredodeza/flask-roberta:latest

```

O trabalho de compilação tem muitas etapas. Nesse caso, cada etapa tem uma tarefa distinta, o que é uma excelente maneira de separar os domínios de falha. Se tudo estivesse em um único script, seria mais difícil identificar possíveis problemas. O primeiro passo é verificar o repositório quando a ação é desencadeada. Em seguida, uma vez que o modelo ONNX não existe localmente, temos de o recuperar a partir do Azure, pelo que temos de nos autenticar utilizando a ação do Azure. Após a autenticação, a ferramenta *az* é disponibilizada e tens de anexar a pasta para o teu espaço de trabalho e grupo. Finalmente, o trabalho pode recuperar o modelo pelo seu ID.

NOTA

Alguns passos no ficheiro YAML têm uma diretiva `uses`, que identifica que ação externa (por exemplo, `actions/checkout`) e em que versão. As versões podem ser ramos ou etiquetas publicadas de um repositório. No caso de `checkout`, é a etiqueta `v2`.

Quando todas estas etapas estiverem concluídas, o modelo RoBERTa-Sequence deverá estar na raiz do projeto, permitindo que as etapas seguintes construam o contentor corretamente.

O ficheiro do fluxo de trabalho está a utilizar `AZURE_CREDENTIALS`. Estas são utilizadas com uma sintaxe especial que permite ao fluxo de trabalho obter segredos configurados para o repositório. Essas credenciais são as informações da entidade de serviço. Se não estiveres familiarizado com uma entidade de serviço, isto é abordado na secção "[Autenticação](#)". Precisarás da configuração da entidade de serviço que tem acesso aos recursos no espaço de trabalho e no grupo onde o modelo reside. Adiciona o segredo no teu repositório GitHub indo a Definições, depois Segredos e, por fim, clicando na ligação "Novo segredo de repositório". [A Figura 4-3](#) mostra o formulário que te será apresentado quando adicionares um novo segredo.

[Actions secrets / New secret](#)

Name

AZURE_CREDENTIALS

Value

```
{
  ...
  "clientId": "xxxxxxxx-3af0-4065-8e14-xxxxxxxxxx",
  ...
  "sqlManagementEndpointUrl": "https://management.core.windows.net:8443/",
  "galleryEndpointUrl": "https://gallery.azure.com/",
  "managementEndpointUrl": "https://management.core.windows.net/"
}
```

Add secret

Figura 4-3. Adicionar segredo

Confirma e envia as alterações para o repositório e, em seguida, vai para o separador Ações. Uma nova execução é imediatamente agendada e deve começar a ser executada em alguns segundos. Depois de alguns minutos, tudo deve estar concluído. No meu caso, a Figura 4-4 mostra que leva cerca de quatro minutos.



Figura 4-4. Sucesso da ação do GitHub

Existem agora algumas partes móveis para realizar uma execução de trabalho bem sucedida. Ao conceber um novo conjunto de etapas (ou pipelines, como abordarei na próxima secção), uma boa ideia é enumerar as etapas e identificar as etapas ávidas. Estas etapas ávidas são etapas que estão a tentar fazer demasiado e têm muita responsabilidade. À primeira vista, é difícil identificar qualquer passo que possa ser problemático. O processo de manutenção de um trabalho de CI/CD inclui o refinamento das responsabilidades dos passos e a sua adaptação em conformidade.

Uma vez identificadas as etapas, podes dividi-las em etapas mais pequenas, o que te ajudará a compreender mais rapidamente a responsabilidade de cada parte. Uma compreensão mais rápida significa uma depuração mais fácil e, embora não seja imediatamente aparente, beneficiarás se fizeres disto um hábito.

Estas são as etapas que utilizamos para acondicionar o modelo RoBERTa-Sequence:

1. Verifica o ramo atual do repositório.
2. Autentica-te na Cloud do Azure.
3. Configura a instalação automática das extensões do Azure CLI.
4. Anexa a pasta para interagir com o espaço de trabalho.
5. Descarrega o modelo ONNX.
6. Constrói o container para o repo atual.

No entanto, há um item final em falta, que é publicar o contentor após a sua construção. Diferentes registos de contentores irão requerer diferentes opções aqui, mas a maioria suporta GitHub Actions, o que é refrescante. O Docker Hub é simples, e tudo o que requer é criar um token e depois guardá-lo como um segredo de projeto do GitHub, juntamente com o teu nome de utilizador do Docker Hub. Uma vez que isso esteja no lugar, adapta o arquivo de fluxo de trabalho para incluir a etapa de autenticação antes da construção:

```
- name: Authenticate to Docker hub
  uses: docker/login-action@v1
  with:
    username: ${{ secrets.DOCKER_HUB_USERNAME }}
    password: ${{ secrets.DOCKER_HUB_ACCESS_TOKEN }}
```

Por fim, actualiza o passo de construção para utilizar `push: true`.

Recentemente, o GitHub também lançou uma oferta de registo de contentores, e a sua integração com o GitHub Actions é simples. Os mesmos passos do Docker podem ser utilizados com pequenas alterações e criando um PAT (Personal Access Token). Começa por criar um PAT indo às definições da tua conta GitHub, clicando em Developer Settings e, finalmente, em tokens de "Acesso pessoal". Quando a página carregar, clica em "Gerar novo token". Dá-lhe uma descrição significativa na secção Nota e certifica-te de que o token tem permissões para escrever pacotes, como eu faço na [Figura 4-5](#).

Note

Github Registry

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry

Figura 4-5. Token de acesso pessoal do GitHub

Quando terminares, é apresentada uma nova página com o token real. Esta é a única vez que verás o token em texto simples, por isso certifica-te de que o copias agora. Em seguida, vai ao repositório onde está o código do contentor e cria um novo segredo de repositório, tal como fizeste com as credenciais principais do serviço Azure. Dá o nome de *GH_REGISTRY* ao novo segredo e cola o conteúdo do PAT criado no passo anterior. Agora estás pronto para atualizar os passos do Docker para publicar o pacote usando o novo token e o registo de contentores do GitHub:

```
- name: Login to GitHub Container Registry
  uses: docker/login-action@v1
  with:
    registry: ghcr.io
    username: ${{ github.repository_owner }}
    password: ${{ secrets.GH_REGISTRY }}

- name: build flask-app and push to registry
  uses: docker/build-push-action@v2
  with:
    context: .
    tags: ghcr.io/alfredodeza/flask-roberta:latest
    push: true
```

No meu caso, *alfredodeza* é a minha conta do GitHub, por isso posso fazer a etiqueta com ela juntamente com o nome *Flask-roberta* do repositório. Estes terão de corresponder de acordo com a tua conta e o teu repositório. Depois de fazeres push das alterações para o ramo principal (ou depois de fazeres merge se tiveres feito um pull request), o trabalho será ativado. O modelo deve ser puxado do Azure, empacotado dentro do contentor e, finalmente, publicado como um Pacote GitHub na sua oferta de registo de contentores, semelhante à [Figura 4-6](#).



Figura 4-6. Contentor do GitHub Package

Agora que o contentor está a empacotar e a distribuir o modelo ONNX de uma forma totalmente automatizada, aproveitando a oferta de CI/CD do GitHub e o registo de contentores, resolvemos o cenário problemático que assumi no início do capítulo: um modelo precisa de ser empacotado num contentor, mas os ficheiros do contentor não estão disponíveis. Desta forma, estás a proporcionar clareza aos outros e ao próprio processo. É segmentado em pequenas etapas e permite que quaisquer atualizações sejam feitas no contêiner. Finalmente, os passos publicam o contentor num registo selecionado.

Podes realizar muitas outras coisas com ambientes CI/CD para além de empacotar e publicar um contentor. As plataformas de CI/CD são a base para automação e resultados confiáveis. Na próxima secção, abordarei outras ideias que funcionam bem independentemente da plataforma. Ao

conheceres os padrões gerais disponíveis noutras plataformas, podes tirar partido dessas funcionalidades sem te preocupares com as implementações.

Usando Cloud Pipelines

A primeira vez que ouvi falar de pipelines, pensei neles como algo mais avançado do que o típico padrão de scripting (um conjunto de instruções procedimentais que representam uma construção). Mas os pipelines não são conceitos avançados de forma alguma. Se já lidaste com scripts shell em qualquer plataforma de integração contínua, então um pipeline parecerá simples de usar. Um pipeline nada mais é do que um conjunto de etapas (ou instruções) que podem atingir um objetivo específico, como publicar um modelo em um ambiente de produção quando executado. Por exemplo, um pipeline com três etapas para treinar um modelo pode ser tão simples quanto a [Figura 4-7](#).



Figura 4-7. Conduta simples

Podes representar o mesmo pipeline como um script de shell que faz as três coisas ao mesmo tempo. Há vários benefícios com um pipeline que separa as preocupações. Quando cada etapa tem uma responsabilidade específica (ou preocupação), é mais fácil de entender. Se um pipeline de etapa única que recupera os dados, valida-os e treina o modelo está falhando, não fica imediatamente claro por que isso pode falhar. Na verdade, podes aprofundar os detalhes, consultar os registos e verificar o erro real. Se separares o pipeline em três passos e o passo *do modelo de treino* estiver a falhar, podes reduzir o âmbito da falha e chegar a uma possível resolução mais rapidamente.

DICA

Uma recomendação geral que podes aplicar aos muitos aspectos da operacionalização da aprendizagem automática é considerar a possibilidade de tornar qualquer operação mais simples para uma futura situação de falha. Evita a tentação de ser rápido e de ter um pipeline (como neste caso) implementado e a funcionar num único passo, porque é mais fácil. Dedica algum tempo a pensar no que te facilitaria (e a outros) a construção de infra-estruturas de ML. Quando ocorrer uma falha e identificares aspectos problemáticos, volta à implementação e melhora-a. Podes aplicar os conceitos de CI/CD à melhoria: a avaliação e a melhoria contínuas dos processos são uma estratégia sólida para ambientes robustos.

Os pipelines Cloud não são diferentes de qualquer plataforma de integração contínua existente, exceto pelo facto de serem alojados ou geridos por um fornecedor Cloud.

Algumas definições de pipelines de CI/CD que podes encontrar tentam definir elementos ou partes de um pipeline de forma rígida. Na realidade, acho que as partes do pipeline devem ser definidas de forma livre e não limitadas por definições. A RedHat [tem uma boa explicação sobre pipelines](#) que descreve cinco elementos comuns: construir, testar, lançar, implantar e validar. Esses elementos são principalmente para misturar e combinar, não para incluí-los estritamente no pipeline. Por exemplo, se o modelo que estás a construir não precisa de ser implementado, então não há necessidade de seguir um passo de implementação. Do mesmo modo, se o teu fluxo de trabalho exigir a extração e o pré-processamento de dados, tens de o implementar como outro passo.

Agora que sabes que um pipeline é basicamente o mesmo que uma plataforma CI/CD com vários passos, deve ser simples aplicar operações de aprendizagem automática a um pipeline acionável. [A Figura 4-8](#) mostra um suposto pipeline bastante simplista, mas ele também pode envolver várias outras etapas e, como mencionei, esses elementos podem ser misturados e combinados em qualquer número de operações e etapas.

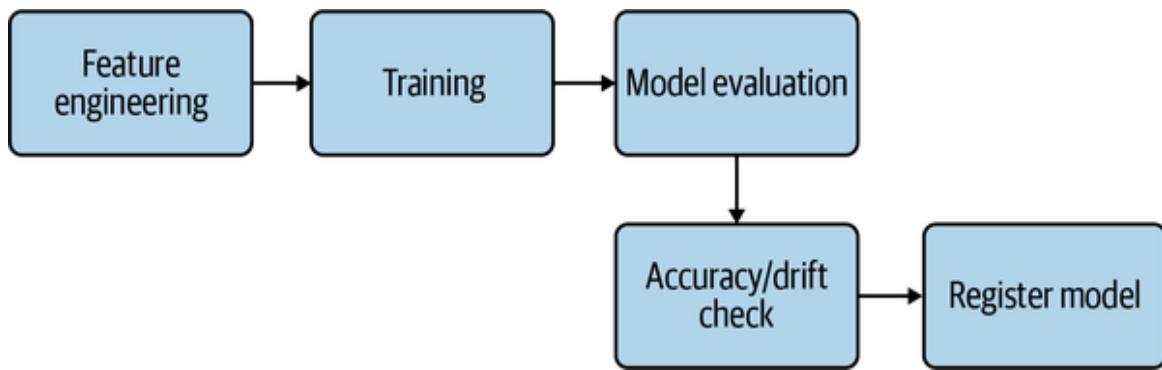


Figura 4-8. Conduta envolvida

O AWS SageMaker faz um excelente trabalho ao fornecer exemplos que estão prontos a usar para criar pipelines envolvidos que incluem tudo o que precisas para executar várias etapas. O SageMaker é uma plataforma especializada em aprendizado de máquina que vai além de oferecer etapas em um pipeline para atingir um objetivo como publicar um modelo. Como é especializada em aprendizagem automática, estás exposto a funcionalidades que são particularmente importantes para colocar modelos em produção. Esses recursos não existem em outras plataformas comuns, como o GitHub Actions, ou, se existem, não são tão bem pensados porque o objetivo principal de plataformas como o GitHub Actions ou o Jenkins não é treinar modelos de aprendizado de máquina, mas sim ser o mais genérico possível para acomodar os casos de uso mais comuns.

Outro problema crucial que é um pouco difícil de resolver é que as máquinas especializadas para formação (por exemplo, tarefas intensivas de GPU) não estão disponíveis ou são difíceis de configurar numa oferta genérica de pipeline.

Abra o SageMaker Studio e vá até a seção Componentes e Registros na barra lateral esquerda e selecione Projetos. Aparecem vários modelos de projectos SageMaker para escolher, como mostrado na [Figura 4-9](#).

Create project

Group related SageMaker components, and resources such as code repositories, pipelines, experiments, model groups, and endpoints into a project. You can also automate model building, and deployment by choosing a project template.

SageMaker project templates

Name

MLOps template for model building, training, and deployment

MLOps template for model deployment

MLOps template for model building and training

Figura 4-9. Modelos do SageMaker

NOTA

Embora os exemplos se destinem a ajudá-lo a começar, e os Jupyter Notebooks sejam fornecidos, eles são ótimos para aprender mais sobre as etapas envolvidas e como alterá-las e adaptá-las às suas necessidades específicas. Depois de criar uma instância do pipeline no SageMaker, treinar e finalmente registrar o modelo, podes navegar pelos parâmetros do pipeline, como na Figura 4-10.

Executions	Graph	Parameters	Settings
		Parameters	Type
		ProcessingInstanceType	String
		ProcessingInstanceCount	Integer
		TrainingInstanceType	String
		ModelApprovalStatus	String
		InputDataUrl	String

Figura 4-10. Parâmetros da conduta

Outra parte crucial do pipeline que mostra todos os passos envolvidos também está disponível, como mostra a Figura 4-11.

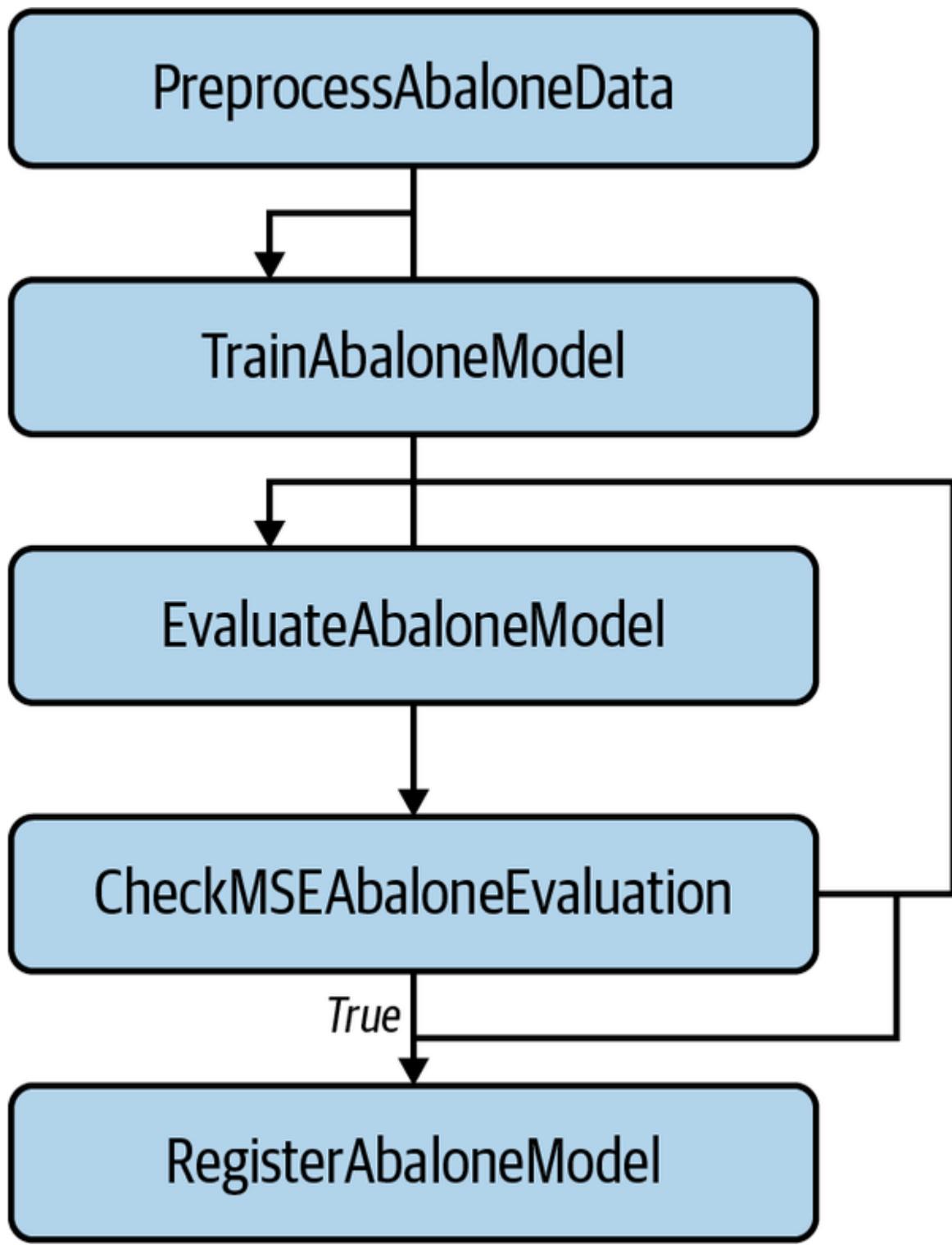


Figura 4-11. Pipeline do SageMaker

Como podes ver, a preparação dos dados, a formação, a avaliação e o registo de um modelo fazem parte do pipeline. O principal objetivo é

register o modelo para implantá-lo posteriormente para inferências ao vivo após o empacotamento. Nem todas as etapas precisam ser capturadas nesse pipeline específico. Podes criar outros pipelines que podem ser executados sempre que houver um modelo recentemente registado disponível. Desta forma, esse pipeline não está ligado a um modelo específico, mas podes reutilizá-lo para qualquer outro modelo que seja treinado com sucesso e registado. A reutilização de componentes e automação é outro componente crítico do DevOps que funciona bem quando aplicado ao MLOps.

Agora que os pipelines estão desmistificados, podemos ver certas melhorias que podem torná-los mais robustos, controlando manualmente o lançamento de modelos ou mesmo mudando a inferência de um modelo para outro.

Lançamento controlado de modelos

Existem alguns conceitos das implantações de serviços da Web que se encaixam perfeitamente nas estratégias de implantação de modelos em ambientes de produção, como a criação de várias instâncias de um aplicativo de inferência ao vivo para escalabilidade e a mudança progressiva de um modelo mais antigo para um mais novo. Antes de entrar em alguns dos detalhes que englobam a parte de controlo da implementação de modelos em produção, vale a pena descrever as estratégias em que estes conceitos entram em jogo.

Nesta secção, abordarei duas destas estratégias em pormenor. Embora essas estratégias sejam semelhantes, elas têm um comportamento específico do qual podes tirar proveito ao implantar:

- Implantação azul-verde
- Implantação de canários

Uma implantação azul-verde é uma estratégia que coloca uma nova versão em um ambiente de teste idêntico ao de produção. Por vezes, este ambiente de staging é igual ao de produção, mas o tráfego é encaminhado de forma diferente (ou separada). Sem entrar em detalhes, Kubernetes é uma plataforma que permite este tipo de implantação com facilidade, uma vez

que podes ter as duas versões no mesmo cluster Kubernetes, mas encaminhando o tráfego para um endereço separado para a versão mais recente ("azul") enquanto o tráfego de produção continua a ir para a mais antiga ("verde"). A razão para esta separação é que permite mais testes e garantia de que o novo modelo está a funcionar como esperado. Quando esta verificação estiver concluída e certas condições forem satisfatórias, modifica a configuração para mudar o tráfego do modelo atual para o novo.

Existem alguns problemas com implantações blue-green, principalmente associados a como pode ser complicado replicar ambientes de produção. Mais uma vez, esta é uma daquelas situações em que o Kubernetes é perfeito, uma vez que o cluster pode acomodar a mesma aplicação com diferentes versões com facilidade.

Uma estratégia de implantação canário é um pouco mais complexa e um pouco mais arriscada. Dependendo do teu nível de confiança e da capacidade de alterar progressivamente a configuração com base nas restrições, é uma boa maneira de enviar modelos para a produção. Neste caso, o tráfego é encaminhado progressivamente para o modelo mais recente *ao mesmo tempo que o modelo anterior está a fornecer previsões*. Assim, as duas versões estão activas e a processar pedidos em simultâneo, mas em proporções diferentes. A razão para esta implementação baseada em percentagem é que podes ativar métricas e outras verificações para captar problemas em tempo real, permitindo-te reverter imediatamente se as condições forem desfavoráveis.

Por exemplo, supõe que um novo modelo com melhor precisão e sem desvio notado está pronto para entrar em produção. Depois que várias instâncias dessa nova versão estiverem disponíveis para começar a receber tráfego, faz uma alteração na configuração para enviar 10% de todo o tráfego para a nova versão. Enquanto o tráfego começa a ser encaminhado, repara numa quantidade desanimadora de erros nas respostas. Os erros HTTP 500 indicam que a aplicação tem um erro interno. Depois de alguma investigação, mostra que uma das dependências Python que fazem a inferência está tentando importar um módulo que foi movido, causando uma exceção. Se a aplicação receber cem pedidos por minuto, apenas dez

deles terão sofrido a condição de erro. Depois de reparar nos erros, altera rapidamente a configuração para enviar todo o tráfego para a versão mais antiga atualmente implementada. Essa operação também é chamada de *reversão*.

A maioria dos provedores de Cloud tem a capacidade de fazer uma implementação controlada de modelos para essas estratégias. Embora este não seja um exemplo totalmente funcional, o Azure Python SDK pode definir a percentagem de tráfego para uma versão mais recente aquando da implementação:

```
from azureml.core.webservice import AksEndpoint

endpoint.create_version(version_name = "2",
                        inference_config=inference_config,
                        models=[model],
                        traffic_percentile = 10)
endpoint.wait_for_deployment(True)
```

A parte complicada é que o objetivo de uma implementação canário é aumentar progressivamente até que o `traffic_percentile` esteja a 100%. O aumento tem de ocorrer em simultâneo com o cumprimento das restrições relativas à integridade da aplicação e a taxas de erro mínimas (ou nulas).

O monitoramento, o registro e as métricas detalhadas dos modelos de produção (além do desempenho do modelo) são absolutamente críticos para uma estratégia de implantação robusta. Eu os considero cruciais para a implantação, mas eles são um pilar central das práticas robustas de DevOps abordadas no [Capítulo 6](#). Além do monitoramento, do registro e das métricas que têm seu próprio capítulo, há outras coisas interessantes a serem verificadas para a entrega contínua. Na próxima secção, veremos algumas que fazem sentido e aumentam a confiança na implementação de um modelo em produção.

Técnicas de teste para implantação de modelos

Até agora, o contentor construído neste capítulo funciona muito bem e faz exatamente o que precisamos: a partir de alguns pedidos HTTP com uma mensagem cuidadosamente elaborada num corpo JSON, uma resposta JSON prevê o sentimento. Um engenheiro de aprendizagem automática experiente pode ter implementado a precisão e a deteção de desvios (abordadas em pormenor no [Capítulo 6](#)) antes de chegar à fase de empacotamento do modelo. Vamos assumir que esse já é o caso e concentrar-nos noutras testes úteis que podes efetuar antes de implementar um modelo na produção.

Quando envias um pedido HTTP ao contentor para produzir uma previsão, é necessário percorrer várias camadas de software do princípio ao fim. A um nível elevado, estas são críticas:

1. O cliente envia um pedido HTTP, com um corpo JSON, na forma de uma matriz com uma única cadeia.
2. Uma porta HTTP específica(*5000*) e um ponto final(*predict*) têm de existir e ser encaminhados para.
3. A aplicação Python Flask tem de receber o payload JSON e carregá-lo em Python nativo.
4. O tempo de execução do ONNX precisa de consumir a cadeia e produzir uma previsão.
5. Uma resposta JSON com uma resposta HTTP 200 tem de conter o valor booleano da previsão.

Cada um destes passos de alto nível pode (e deve) ser testado.

Controlos automáticos

Enquanto montava o contentor para este capítulo, tive alguns problemas com o módulo Python *onnxruntime*: a documentação não indica (um número de versão exato) a versão, o que fez com que fosse instalada a versão mais recente, que necessitava de argumentos diferentes como entrada. A precisão do modelo foi boa, e não consegui detetar um desvio

significativo. E, no entanto, implementei o modelo apenas para o descobrir completamente avariado quando os pedidos foram consumidos.

Com o tempo, as aplicações tornam-se melhores e mais resistentes. Outro engenheiro pode adicionar o tratamento de erros para responder com uma mensagem de erro quando são detectadas entradas inválidas e, talvez, com uma resposta HTTP com um código de erro HTTP apropriado, juntamente com uma mensagem de erro agradável que o cliente possa compreender. Deves testar estes tipos de adições e comportamentos antes de permitir que um modelo seja enviado para produção.

Às vezes, não há nenhuma condição de erro HTTP e também não há rastreamentos Python. O que aconteceria se eu fizesse uma alteração como a seguinte na resposta JSON:

```
{  
    "positive": "false"  
}
```

Sem olhar para as secções anteriores, consegues ver a diferença? A mudança passaria despercebida. A estratégia de implementação do canário iria para 100% sem quaisquer erros detectados. O engenheiro de aprendizagem automática ficaria satisfeito com a elevada precisão e a ausência de desvios. E, no entanto, esta mudança quebrou completamente a eficácia do modelo. Se não conseguiste perceber a diferença, não há problema. Encontro este tipo de problemas a toda a hora e, por vezes, levo horas a detetar o problema: em vez de `false` (um valor booleano), utiliza `"false"` (uma cadeia de caracteres).

Nenhuma destas verificações deve ser manual; a verificação manual deve ser reduzida ao mínimo. A automação deve ser uma prioridade alta, e as sugestões que fiz até agora podem ser adicionadas como parte do pipeline. Essas verificações podem ser generalizadas para outros modelos para reutilização, mas em um nível alto, elas podem ser executadas em paralelo, como mostrado na [Figura 4-12](#).

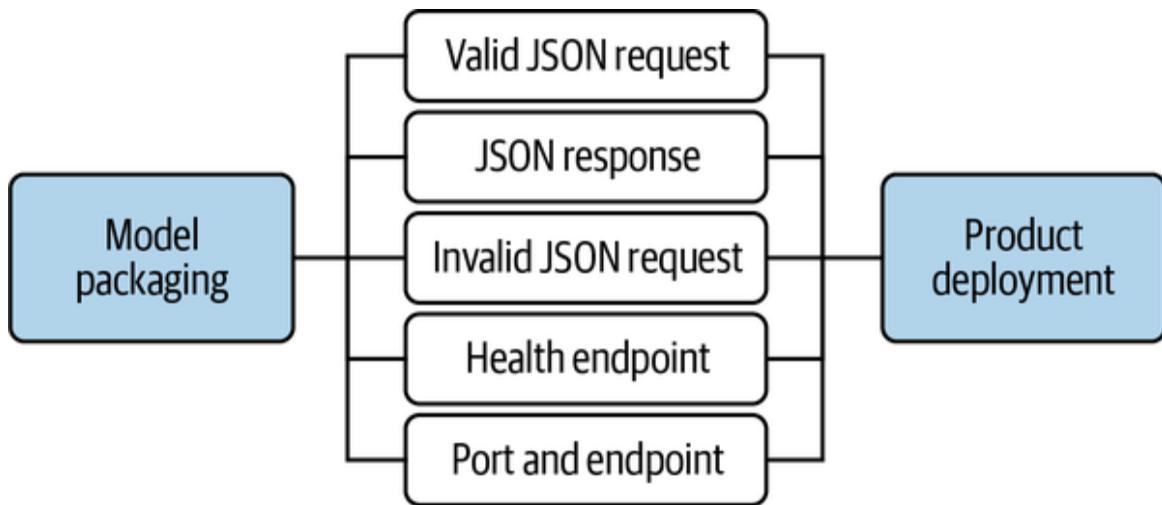


Figura 4-12. Controlos automáticos

Linting

Para além de algumas das verificações funcionais que mencionei, como o envio de pedidos HTTP, há outras verificações mais próximas do código na aplicação Flask que são muito mais simples de implementar, como a utilização de um linter([recomendo o Flake8 para Python](#)). Seria melhor automatizares todas estas verificações para evitares ter problemas quando for necessário fazer o lançamento em produção. Independentemente do ambiente de desenvolvimento em que te encontrares, recomendo *vivamente* que actives um linter para escrever código. Ao criar a aplicação Flask, encontrei erros à medida que adaptava o código para trabalhar com pedidos HTTP. Aqui está um pequeno exemplo da saída do linter:

```
$ flake8 webapp/app.py
webapp/app.py:9:13: F821 undefined name 'RobertaTokenizer'
```

Nomes indefinidos quebram aplicações. Neste caso, esqueci-me de importar o `RobertaTokenizer` do módulo `transformers`. Assim que me apercebi disso, adicionei a importação e resolvi o problema. Isto não me levou mais do que alguns segundos.

De facto, quanto mais cedo conseguires detetar estes problemas, melhor. Quando se fala em segurança de software, é comum ouvirmos "cadeia de fornecimento de software", em que *a cadeia* são todos os passos desde o

desenvolvimento até ao envio do código para produção. E nesta cadeia de eventos, há um constante impulso para *mudar para a esquerda*. Se vires estes passos como uma grande cadeia, o elo mais à esquerda é o programador que cria e actualiza o software, e o fim da cadeia (o mais à direita) é o produto lançado, onde o consumidor final pode interagir com ele.

Quanto mais cedo puderes deixar a deteção de erros, melhor. Isto porque é mais barato e mais rápido do que esperar até estar em produção quando é necessário fazer um rollback.

Melhoria contínua

Há alguns anos atrás, eu era o gestor de lançamento de um grande software de código aberto. O software era tão complicado de lançar que me levava de dois dias a uma semana inteira. Era difícil fazer melhorias porque eu também era responsável por outros sistemas. Uma vez, enquanto tentava lançar uma versão, seguindo os muitos passos diferentes para publicar os pacotes, um programador principal pediu-me para fazer uma última alteração. Em vez de dizer "*não*" imediatamente, perguntei-lhe: "*Esta alteração já foi testada?*"

A resposta foi completamente inesperada: "*Não sejas ridículo, Alfredo, esta é uma alteração de uma linha, e é um comentário de documentação numa função. Precisamos mesmo que esta alteração faça parte do lançamento*". A pressão para que a mudança fosse feita veio do topo, e eu tive que ceder. Acrescentei a alteração de última hora e fiz o lançamento.

Logo na manhã seguinte, voltámos a ter utilizadores (e, mais importante, clientes) a queixarem-se de que a última versão estava completamente avariada. Instalava, mas não funcionava de todo. O culpado era a alteração de uma linha que, embora fosse um comentário dentro de uma função, estava a ser analisada por outro código. Havia uma sintaxe inesperada nesse comentário, o que impedia a aplicação de arrancar. A história não tem como objetivo castigar o programador. Ele não sabia o que fazer. Todo o processo foi um momento de aprendizagem para todos os envolvidos, e agora era claro o quanto cara era esta alteração de uma linha.

Seguiu-se um conjunto de acontecimentos perturbadores. Para além de reiniciar o processo de lançamento, a fase de testes para a única alteração demorou mais um dia (extra). Por fim, tive de *retirar* os pacotes lançados e refazer os repositórios para que os novos utilizadores recebessem a versão anterior.

Foi mais do que dispendioso. O número de pessoas envolvidas e o elevado impacto fizeram com que esta fosse uma excelente oportunidade para afirmar que isto não deve ser permitido novamente - mesmo que seja uma alteração de uma linha. Quanto mais cedo for detectado, menor será o impacto e mais barato será corrigir .

Conclusão

A entrega contínua e a prática de feedback constante são cruciais para um fluxo de trabalho robusto. Como este capítulo prova, há muito valor na automação e na melhoria contínua do ciclo de feedback. Os contentores de empacotamento, juntamente com os pipelines e as plataformas CI/CD em geral, destinam-se a facilitar a adição de mais verificações, que visam aumentar a confiança dos modelos de envio paraprodução.

Enviar modelos para produção é o objetivo número um, mas fazê-lo com uma confiança muito elevada, num conjunto de passos resilientes, é o que deves procurar. A tua tarefa não termina quando os processos estão implementados. Deves continuar a encontrar formas de te agradeceres mais tarde, fazendo a seguinte pergunta: o que posso acrescentar hoje para facilitar a minha vida se este processo falhar? Por último, recomendo vivamente que cries estes fluxos de trabalho de uma forma que facilite a adição de mais controlos e verificações. Se for difícil, ninguém lhe vai querer tocar, anulando o objetivo de um fluxo de trabalho robusto para enviar modelos paraprodução.

Agora que já tens uma boa noção da entrega de modelos e do aspetto da automatização, vamos mergulhar no AutoML e no Kaizen no próximo capítulo.

Exercícios

- Cria a tua própria aplicação Flask num contentor, publica-a num repositório GitHub, documenta-a cuidadosamente e adiciona GitHub Actions para garantir que é construída corretamente.
- Faz alterações ao contentor ONNX para que este faça push para o Docker Hub em vez de para o GitHub Packages.
- Modifica um pipeline do SageMaker, para que ele te pergunte antes de registrar o modelo depois de treiná-lo.
- Utilizando o SDK do Azure, cria um bloco de notas Jupyter que aumente o percentil de tráfego que vai para um contentor.

Questões para discussão sobre pensamento crítico

- Indica pelo menos quatro verificações críticas que podes adicionar para verificar se um modelo empacotado num contentor foi construído corretamente.
- Quais são as diferenças entre as implementações canário e azul-verde? Qual delas preferes? Qual preferes? Porquê?
- Porque é que os pipelines Cloud são úteis em comparação com a utilização de GitHub Actions? Cita pelo menos três diferenças.
- O que significa *embalar um contentor*? Porque é que é útil?
- Quais são as três características dos modelos de aprendizagem automática de pacotes?

Capítulo 5. AutoML e KaizenML

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

Se fores demasiado preso por regras, os pensamentos parciais não podem florescer. Regras sem ideias é prisão. Ideias sem regras é o caos. O bonsai ensina-nos o equilíbrio. O equilíbrio entre as regras e a inovação é um problema generalizado em toda a vida. Uma vez vi uma peça de teatro intitulada "O Jogo da Vida". A mensagem era que, muitas vezes, nos pedem para jogar o jogo com apostas altas antes de alguém nos ter explicado as regras. Além disso, não é assim tão fácil saber se estás a ganhar. Muitas vezes parece que os principiantes (e os jovens em geral) precisam de regras ou de teorias gerais para se orientarem. Depois, à medida que a experiência se acumula, as muitas excepções e variações invalidam gradualmente as regras, ao mesmo tempo que as regras se tornam menos necessárias. Uma grande vantagem do bonsai em relação à vida é que podes aprender com os erros fatais.

—Dr. Joseph Bogen

É uma altura empolgante para te envolveres na construção de sistemas de aprendizagem automática. A aprendizagem automática, ou seja, a aprendizagem a partir de dados, tem um valor claro para a humanidade na resolução de problemas, desde veículos autónomos a rastreios e tratamentos mais eficazes do cancro. Ao mesmo tempo, a automatização desempenha um papel fundamental para permitir este avanço na automatização da criação de modelos, AutoML, e o resto das tarefas relacionadas com a aprendizagem automática, algo a que chamo KaizenML.

Enquanto o AutoML se concentra estritamente na criação de um modelo a partir de dados limpos, o KaizenML tem a ver com a automatização de *tudo*

o que diz respeito ao processo de aprendizagem automática e à sua melhoria. Vamos analisar os dois tópicos, começando pela razão pela qual o AutoML é tão essencial.

NOTA

Especialistas em aprendizagem automática, como Andrew Ng, reconhecem agora que uma abordagem centrada nos dados tem méritos sobre um processo centrado no modelo. Outra forma de afirmar isto é que o Kaizen, ou seja, a melhoria contínua de todo o sistema, desde os dados ao software, ao modelo e ao ciclo de feedback do cliente, é essencial. KaizenML, na minha opinião, significa que estás a melhorar todos os aspectos de um sistema de aprendizagem automática: qualidade dos dados, qualidade do software e qualidade do modelo.

AutoML

O autor Upton Sinclair disse a famosa frase "é difícil fazer com que um homem comprehenda algo quando o seu salário depende do facto de não o compreender". Um excelente exemplo da citação de Upton Sinclair em ação é a desinformação das redes sociais documentada no documentário da Netflix, *The Social Dilemma*. Imagina que trabalhas numa empresa que espalha desinformação em grande escala e é muito bem paga. Nesse caso, é quase impossível aceitar que és um ator nesse processo e que a tua empresa lucra, de facto, muito com a desinformação. Isso contribui para o teu excelente salário e estilo de vida.

Da mesma forma, criei algo a que chamo a lei do automatizador. Uma vez iniciada a conversa sobre a automatização de uma tarefa, a automatização acaba por ocorrer. Alguns exemplos incluem a substituição de centros de dados por computação em Cloud e a substituição de operadores de centrais telefónicas por máquinas. Muitas empresas têm-se agarrado aos seus centros de dados com "unhas e dentes", dizendo que a Cloud era a raiz de todo o mal no mundo. No entanto, acabaram por mudar para a Cloud ou estão em vias de o fazer.

Demorou quase 100 anos, de cerca de 1880 a 1980, para automatizar totalmente a troca de chamadas à mão para que uma máquina as fizesse,

mas aconteceu. As máquinas são óptimas para automatizar tarefas manuais de trabalho intensivo. Se, em 1970, o teu trabalho envolvesse a troca de chamadas telefónicas, talvez tivesses desdenhado da ideia de automatizar o que fazias, uma vez que sabias o quanto difícil era a tarefa. Hoje, com a ciência dos dados, pode acontecer que sejamos operadores de central telefónica a introduzir furiosamente valores de hiperparâmetros em funções Python e a partilhar os nossos resultados no Kaggle, sem saber que tudo isto está em vias de ser automatizado.

No livro "*How We Know What Isn't So*" (*Como sabemos o que não é verdade*), Thomas Gilovich aponta as estratégias de auto-humilhação:

Existem duas classes de estratégias de auto-humilhação: a real e a fingida. A auto-humilhação "real" consiste em colocar obstáculos visíveis ao sucesso no teu próprio caminho. Os obstáculos tornam menos provável o sucesso, mas fornecem uma desculpa pronta para o fracasso. O estudante que não estuda antes de um exame ou o aspirante a ator que bebe antes de uma audição são bons exemplos. Por vezes, o fracasso é quase garantido, mas pelo menos não se pensa que a pessoa não tem as capacidades necessárias (ou assim se espera).

O self-handicapping "fingido", por outro lado, é, em certos aspectos, uma estratégia menos arriscada, em que a pessoa se limita a afirmar que houve obstáculos difíceis no caminho para o sucesso. Este tipo de auto-humilhação consiste simplesmente em inventar desculpas para um eventual mau desempenho, antes ou depois do facto.

Quando as iniciativas de ciência de dados falham, é fácil mergulhar em qualquer uma destas estratégias de auto-humilhação. Um exemplo disso na ciência de dados poderia ser não usar o AutoML quando ele poderia ajudar em certos aspectos de um projeto; essa é uma desvantagem "real" para o seu sucesso. No entanto, uma das regras de ouro da engenharia de software é usar as melhores ferramentas que puderdes para a tarefa em questão. A razão para utilizar as melhores ferramentas disponíveis é que elas reduzem a complexidade do software desenvolvido. Um excelente exemplo de uma ferramenta "melhor da categoria" que reduz a complexidade é o GitHub Actions, porque é simples criar testes automatizados. Outro exemplo é um

editor como o Visual Studio Code, devido à sua capacidade de executar o recurso de autocompletar código, realce de sintaxe e linting com configuração mínima. Ambas as ferramentas aumentam drasticamente a produtividade do desenvolvedor, simplificando o processo de criação de software.

Com a ciência dos dados, este mantra de "utiliza as melhores ferramentas disponíveis" precisa de ser evangelizado. Em alternativa, se um projeto de ciência de dados falhar, como acontece muitas vezes, uma estratégia de auto-humilhação pode ser dizer que o problema era demasiado difícil. Em ambos os casos, a solução para a auto-humilhação é adotar a automação, quando apropriado.

Compara a comida com a aprendizagem automática na [Figura 5-1](#). Repara que a comida tem muitas formas, desde a farinha que compras na loja para fazeres a tua própria pizza até à que mandas entregar em tua casa. Só porque uma é muito mais complexa do que outra (ou seja, fazer uma pizza a partir do zero versus encomendar uma pizza quente pronta) não significa que a opção de entrega ao domicílio não seja também considerada comida. A dificuldade, ou a falta dela, não equivale a completude ou realidade.

Da mesma forma, não aceitar a realidade não significa que ela não esteja a acontecer. Outra forma de descrever a negação da realidade é chamar-lhe "pensamento mágico". No início da pandemia da COVID-19, muitos pensadores mágicos disseram: "Isto é como a gripe", como forma de se tranquilizarem a si próprios (e aos outros) de que o perigo não era tão grave como parecia ser. No entanto, os dados de 2021 dizem algo completamente diferente. A COVID-19 nos EUA aproximou-se de cerca de 75% das mortes de todas as formas combinadas de doenças cardíacas, atualmente a principal causa de morte nos EUA. Da mesma forma, Justin Fox, num [artigo da Bloomberg](#) que utiliza os dados do CDC, mostra que esta pandemia é várias vezes mais mortal para a maioria dos grupos etários do que a gripe. Vê a [Figura 5-2](#).

MLOps

You don't need to build it to use it!

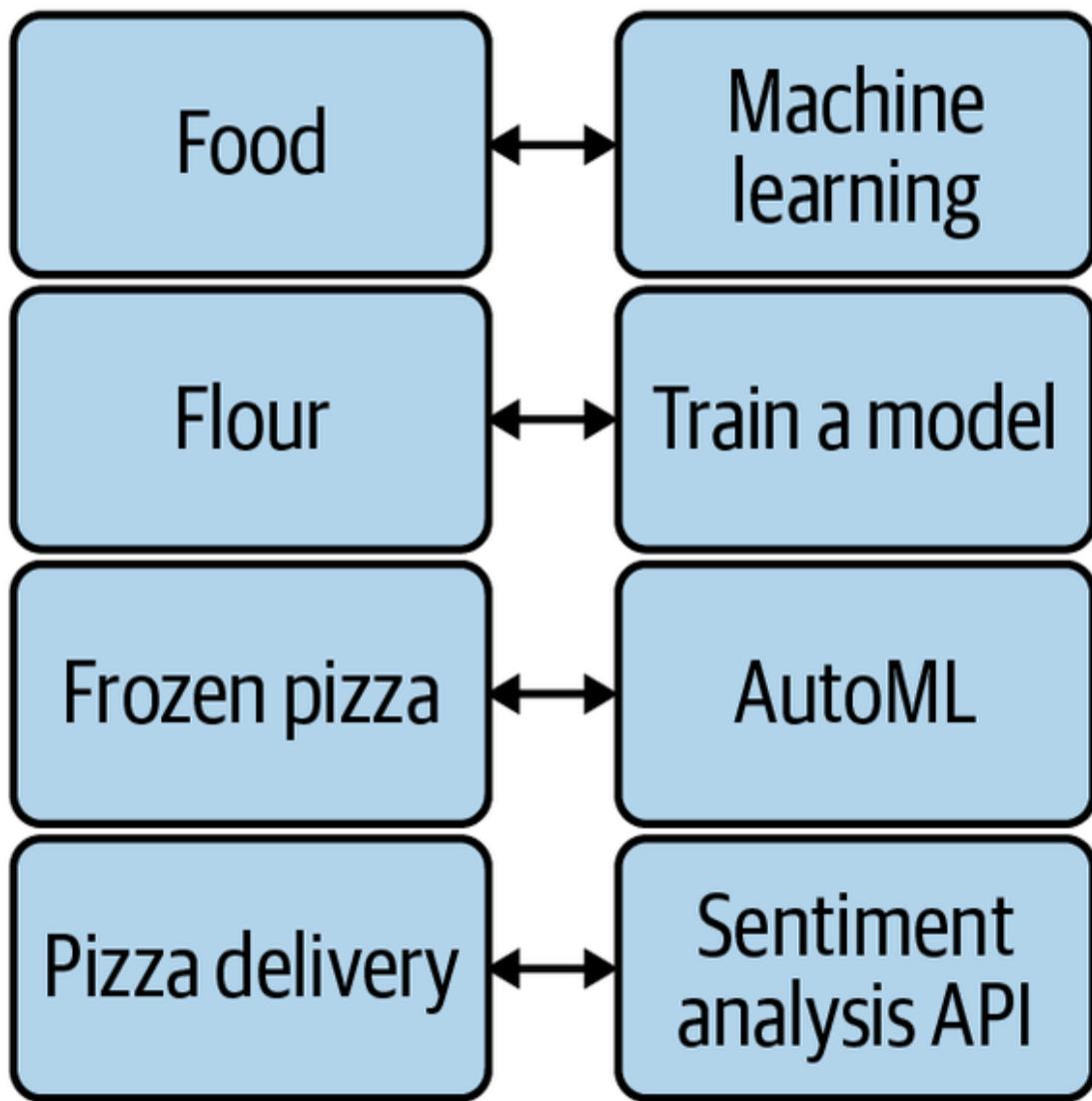
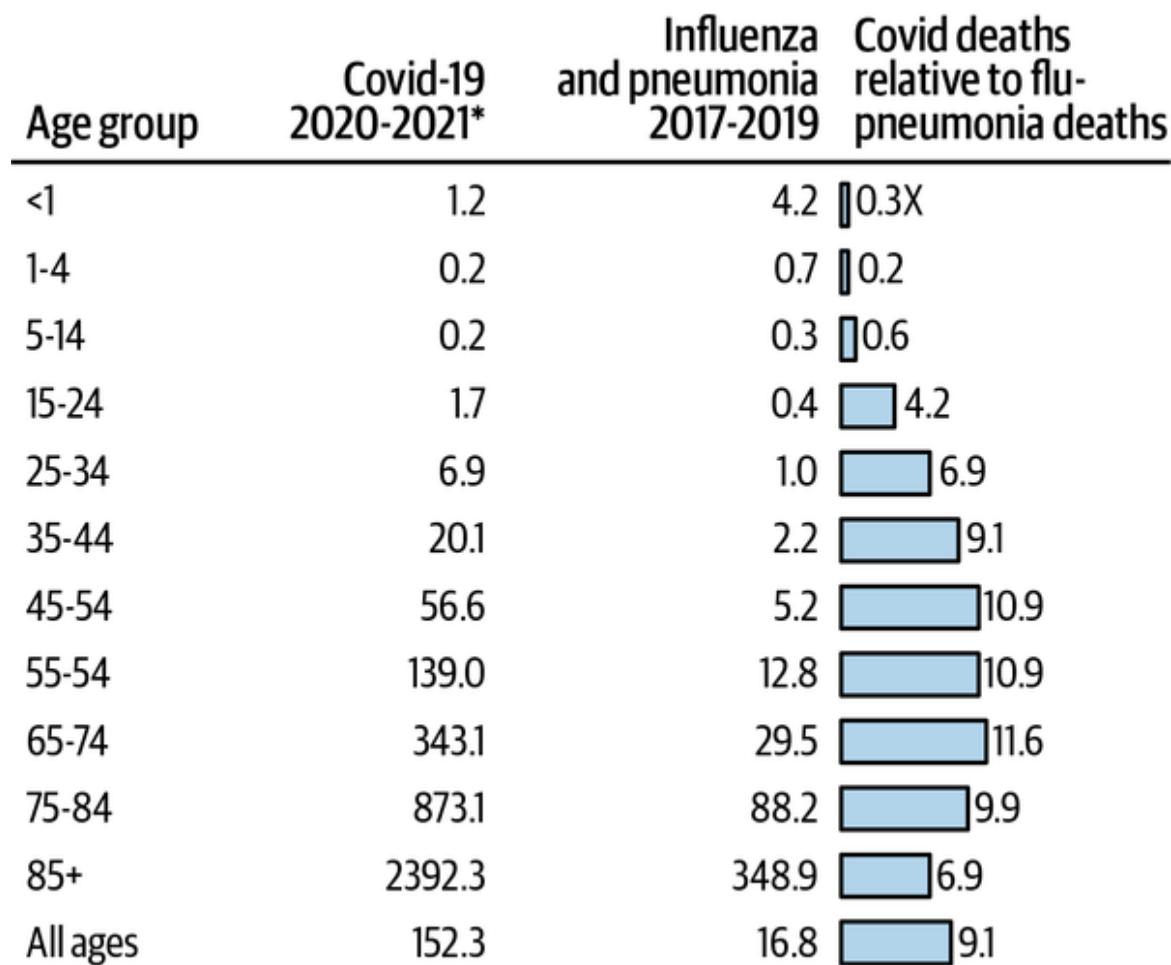


Figura 5-1. Alimentos versus ML

Covid Versus Flu and Pneumonia

Annual US deaths per 100,000 population age group



Source: Centers for Disease Control and Prevention

*Assuming 500,000 deaths in a 12 month period

Figura 5-2. Covid contra a gripe e a pneumonia (fonte: [Bloomberg News](#))

O AutoML é um ponto de inflexão para os cientistas de dados porque partilha semelhanças com outras tendências históricas: automação e pensamento mágico. Tudo o que pode ser automatizado será automatizado. Aceitar esta tendência em vez de a combater conduzirá a fluxos de trabalho muito mais produtivos na aprendizagem automática. O AutoML, em particular, pode ser uma das tecnologias mais críticas a adotar para implementar totalmente a filosofia MLOps.

Antes do surto de COVID-19, uma cientista investigadora da UC Berkeley, a Dra. Jennifer Doudna, e a colaboradora Dra. Emmanuelle Charpentier

trabalharam na difícil tarefa de pesquisar a edição de genes. Quando a pandemia da COVID-19 começou, a Dra. Doudna sabia que precisava de converter rapidamente esta pesquisa numa forma inovadora de acelerar a criação de uma vacina. Como resultado, começou a trabalhar para "salvar o mundo".

NOTA

Como é que a descoberta de medicamentos para a COVID-19 está relacionada com os MLOps? Um problema crítico na ciência dos dados é colocar uma solução de pesquisa em produção. Da mesma forma, um problema fundamental da pesquisa médica é fazer com que a descoberta chegue às mãos de um paciente que dela beneficie.

Em *The Code Breaker: Jennifer Doudna, Gene Editing, and the Future of the Human Race* (Simon & Schuster Australia), Walter Isaacson descreve como a Dra. Doudna era "...agora uma forte crente de que a pesquisa básica deve ser combinada com a pesquisa translacional, movendo as descobertas da bancada para a cabeceira da cama..." A cientista vencedora do Prémio Nobel, Dra. Jennifer Doudna, é co-creditada, juntamente com a Dra. Emmanuelle Charpentier, na criação da pesquisa que levou à edição de genes e à aplicação comercial destes mecanismos CRISPR.

Um dos seus rivais, o Dr. Feng Zhang, que acabou por trabalhar numa vacina concorrente, a Moderna, referiu que o laboratório da UC Berkeley não estava a trabalhar na aplicação desta vacina em células humanas. A sua crítica é que o seu laboratório estava a trabalhar na utilização da pesquisa CRISPR visando células humanas, enquanto a Dra. Doudna estava focada estritamente na pesquisa.

Esta crítica é o cerne de uma disputa de patentes sobre quem pode reivindicar o crédito por estas descobertas, ou seja, quanto trabalho foi pesquisa e quanto foi a aplicação da pesquisa? Não te parece um pouco semelhante às disputas entre ciência de dados e engenharia de software? Em última análise, o Dr. Doudna conseguiu, de facto, "levá-la à produção" sob a

forma da vacina da Pfizer. Recebi recentemente esta vacina e, tal como muitas pessoas, estou entusiasmado por ela ter chegado à produção.

Que mais poderíamos realizar coletivamente se tivéssemos o mesmo sentido de urgência que os cientistas que estão a "operacionalizar" a vacina para a COVID-19? Quando era gestor de engenharia em startups, gostava de fazer perguntas hipotéticas às pessoas. Uma variante de "E se tivesses de salvar o mundo dentro de um prazo"? Gosto desta pergunta porque vai rapidamente ao cerne das coisas. Vai ao encontro da natureza do problema, porque se o relógio estiver a contar para salvar milhões de vidas, trabalha apenas nos componentes essenciais do problema.

Há um documentário incrível no Netflix intitulado *World War II in Colour* (*A Segunda Guerra Mundial a Cores*). O que é impressionante no documentário é que mostra as imagens reais restauradas e coloridas de eventos históricos trágicos. Isto ajuda-te realmente a imaginar como teria sido estar presente durante esses acontecimentos. Nesse sentido, imagina-te numa situação em que tens de resolver um problema técnico que salvaria o mundo. Claro que, se te enganares, todas as pessoas que conheces sofrerão um destino horrível. No entanto, o AutoML ou qualquer forma de automação, associado a um sentido de urgência na resolução apenas dos componentes necessários de um problema, pode conduzir a melhores resultados a nível global: por exemplo, a descoberta de medicamentos e a deteção do cancro.

Esta forma de pensamento situacional acrescenta uma componente de clarificação à decisão de como resolver um problema. Ou o que estás a fazer é importante, ou não é. É muito semelhante à forma como pensavam os cientistas que estavam a trabalhar nas vacinas contra a COVID-19. Ou o que os cientistas fizeram conduziu a uma vacina contra a COVID-19 mais rápida, ou não. Como resultado, cada dia desperdiçado foi um dia em que mais pessoas em todo o mundo sucumbiram ao vírus.

Da mesma forma, lembro-me de ter ido a uma startup elegante na Bay Area por volta de 2016 e de ter comentado que tinham recebido um financiamento de 30 milhões de muitas empresas de capital de risco de

renome. O diretor de operações disse-me então, em privado, que estava muito preocupado com o facto de não terem um produto real ou uma forma de ganhar dinheiro. Receberam ainda mais dinheiro muitos anos depois, e ainda não tenho a certeza de qual é o seu produto real.

Como esta empresa não consegue criar receitas, angaria fundos. Se não consegues angariar fundos, então tens de criar receitas. Da mesma forma, se não puderes colocar o teu modelo em produção com aprendizagem automática, continuas a fazer "pesquisa de ML", ou seja, a trabalhar no ajuste de hiperparâmetros num projeto Kaggle. Por isso, uma boa pergunta que os profissionais do Kaggle devem fazer é: "Temos a certeza de que não estamos apenas a automatizar o nosso trabalho de ajustar os hiperparâmetros através do treino da tecnologia AutoML da Google?"

Nós gravitamos em torno daquilo em que somos excelentes. Há muitas coisas bonitas em concentrarmo-nos naquilo que fazemos bem, como, por exemplo, o facto de isso nos levar a uma carreira de sucesso. No entanto, também há alturas em que deves desafiar-te a pensar temporariamente numa situação para resolver um problema da forma mais urgente possível, tal como fizeram os Drs. Doudna e Zhang com a COVID-19. Isto altera a abordagem que usas? Por exemplo, se eu tivesse quatro horas para treinar um modelo e colocá-lo em produção para salvar o mundo, escreveria o mínimo de código possível e usaria ferramentas de automação prontas para uso, como o Azure AutoML, o Apple Create ML, o Google AutoML, o H2O ou o Ludwig. Nesse caso, a pergunta que se segue é: por que razão escrevo código ou, pelo menos, escrevo a menor quantidade de código possível para todos os projectos de engenharia de aprendizagem automática?

O mundo precisa de modelos de aprendizagem automática de alta qualidade na produção, especialmente porque há muitos problemas urgentes para resolver: encontrar uma cura para o cancro, otimizar a energia limpa, melhorar a descoberta de medicamentos e criar transportes mais seguros. Uma forma de a sociedade o fazer coletivamente é automatizar o que pode ser automatizado agora e concentrar-se em encontrar formas de automatizar o que não pode ser automatizado hoje.

O AutoML é a automatização das tarefas relacionadas com o treino de um modelo em dados limpos. No entanto, nem todos os problemas são tão simples no mundo real e, como resultado, *tudo o que* está relacionado com a aprendizagem automática necessita de automatização. É nesta lacuna que entra o KaizenML. Kaizen, em japonês, significa melhoria contínua. Com o KazienML, a melhoria contínua e a automatização são a forma central de desenvolver sistemas de aprendizagem automática. Vamos aprofundar este conceito a seguir.

MLOps Revolução Industrial

Muitos estudantes e profissionais de aprendizagem automática consideram o AutoML um tópico polarizador. A ciência de dados é um comportamento; o AutoML é uma técnica - eles são uma pequena parte da construção de um sistema de ML, e são complementares. O AutoML é polarizador porque os cientistas de dados presumem que ele substituirá seu trabalho, quando, na verdade, o AutoML é 5% de um processo gigantesco de automação e melhoria contínua, ou seja, MLOps/engenharia de ML/KaizenML, conforme descrito na [Figura 5-3](#).

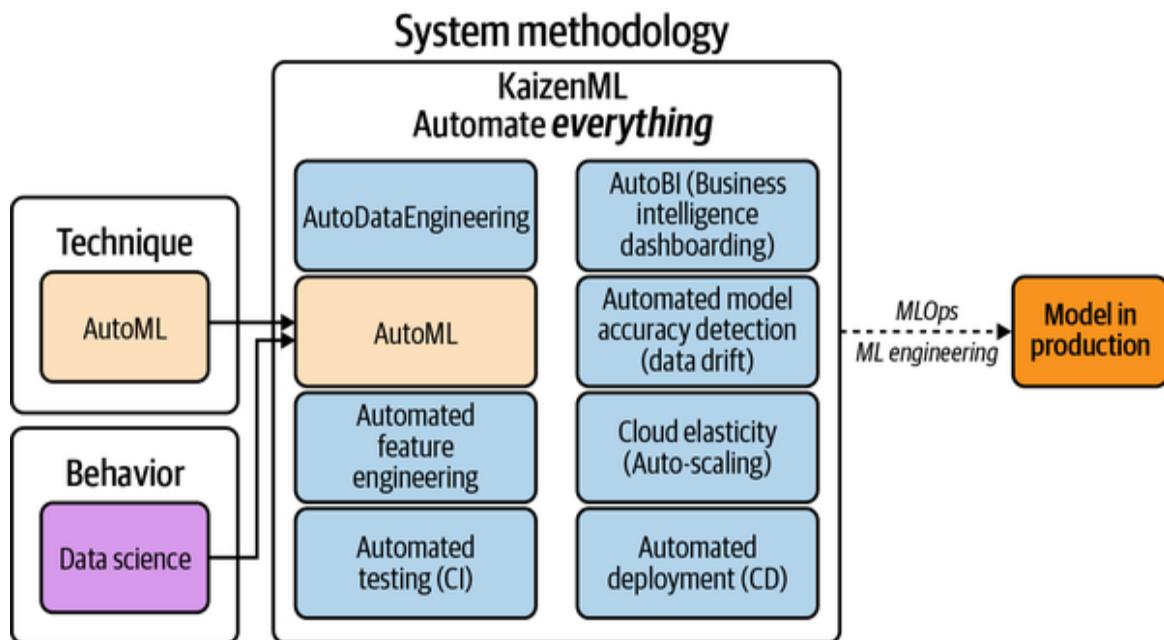


Figura 5-3. O AutoML é uma pequena parte do KaizenML

A revolução industrial, que decorreu entre 1760 e 1840, foi um período em que as tarefas humanas passaram a ser automatizadas, graças às máquinas a vapor e a carvão. Esta automatização levou a um aumento da população, do PIB e da qualidade de vida. Mais tarde, por volta de 1870, ocorreu a segunda revolução industrial, que permitiu a produção em massa e novos sistemas de redes eléctricas.

Há uma série excelente no Disney+ chamada *Made in a Day*. O primeiro episódio mostra como a Tesla utiliza robôs nas fases de desenvolvimento dos automóveis. Os robôs aparafulsam coisas, aparafulsam coisas e soldam peças. Ao olhar para esta fábrica, penso em como os humanos estão a ajudar os robôs. Essencialmente, alimentam os robôs com trabalho que eles próprios ainda não conseguem automatizar totalmente.

Da mesma forma, quando olhamos para os fluxos de trabalho tradicionais da ciência dos dados, cheios de configurações Snowflake únicas, com humanos a "aparafulsar" hiperparâmetros, faz-me pensar numa fábrica de montagem da Ford na segunda revolução industrial. Eventualmente, as tarefas manuais humanas são automatizadas e a primeira coisa que é automatizada é a mais fácil de automatizar.

Outra questão que as pessoas colocam é se muitos aspectos das técnicas de aprendizagem automática são mesmo necessários, como o ajuste manual dos hiperparâmetros, ou seja, a escolha do número de clusters. Imagina ir a uma fábrica da Tesla cheia de robótica avançada e dizer aos engenheiros de automação que os humanos também podem soldar peças. Esta afirmação seria um non sequitur. É claro que nós, humanos, podemos realizar tarefas que as máquinas fazem melhor do que nós, mas será que devemos fazê-lo? Da mesma forma, em muitos aspectos tediosos e manuais da aprendizagem automática, as máquinas fazem um trabalho melhor.

O que pode acontecer em breve na aprendizagem automática e na inteligência artificial é que a técnica seja essencialmente comoditizada. Em vez disso, a chave é a própria automatização e a capacidade de a executar. O programa de televisão sobre fabrico físico tem o nome "Made in a Day" porque os carros ou as guitarras são fabricados num só dia! Muitas

empresas que fazem aprendizagem automática não conseguem construir um modelo baseado em software num ano inteiro, mas como é que este pode ser o processo futuro?

Um cenário possível que vejo acontecer em breve é que pelo menos 80% dos modelos de formação manual da ciência dos dados sejam substituídos por ferramentas AutoML de código aberto comoditizadas ou por modelos pré-construídos descarregados. Este futuro pode acontecer à medida que os projectos de código aberto, como o Ludwig, ou os projectos comerciais, como o Apple CreateML, avançam em sofisticação. O software para treinar modelos de aprendizagem automática pode tornar-se algo como o kernel do Linux, gratuito e omnipresente.

Se estiverem na sua forma atual, a ciência dos dados pode tornar-se bimodal; ou te pagam \$1M/ano, ou és um principiante. A maior parte das vantagens competitivas está relacionada com as melhores práticas tradicionais de engenharia de software: dados/utilizadores, automação, execução e gestão sólida de produtos e práticas comerciais. Um cientista de dados pode tornar-se uma competência normal como a contabilidade, a escrita ou o pensamento crítico noutros casos, em vez de ser apenas um título profissional. Podes chamar a isto a revolução industrial dos MLOps.

A Figura 5-4 é um exemplo disso na prática. Imagina o Kaggle como um ciclo de feedback que a Google utiliza para melhorar muito as suas ferramentas de AutoML. Porque é que não usariam os modelos de treino dos cientistas de dados humanos para melhorar os serviços de AutoML? Na ciência de dados 1.0, os humanos estão a "clicar em botões" manualmente, tal como os operadores de centrais telefónicas do passado. Entretanto, se quisesse, a Google poderia utilizar estes humanos para treinar os seus sistemas de AutoML para realizar estas tarefas manuais de ciência de dados. Na ciência de dados 2.0, que em muitos casos já está aqui, as ferramentas automatizadas treinam exaustivamente os modelos previamente treinados no Kaggle.

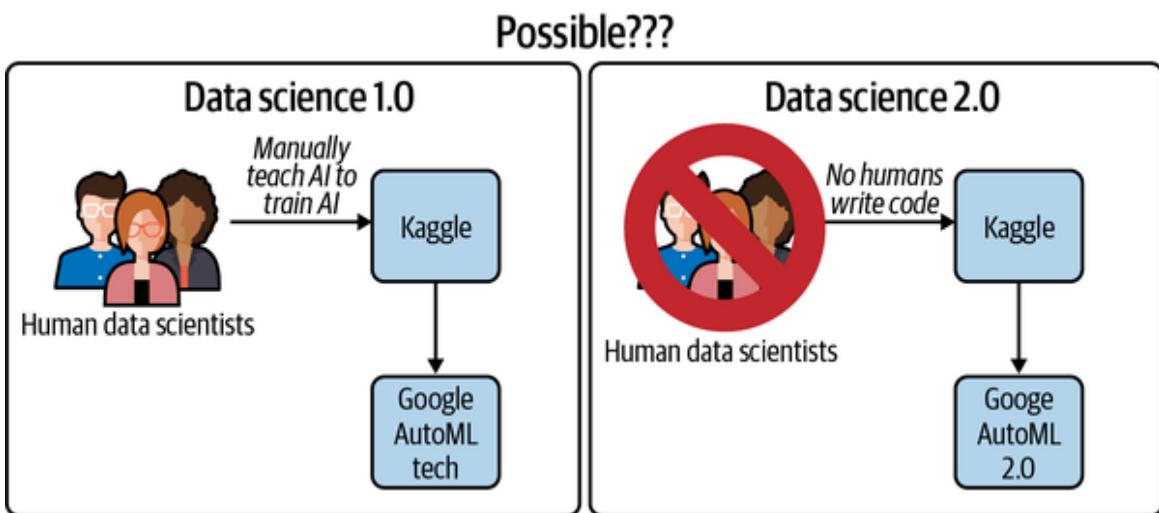


Figura 5-4. Automação do Kaggle

A revolução industrial dos MLOps está a acontecer diante dos nossos olhos, à medida que as máquinas desempenham um papel cada vez mais importante na aprendizagem automática e na ciência dos dados. Em que competências deves investir se estas mudanças estão em curso? Sei um especialista mundial em automação e execução, tanto do ponto de vista técnico como do ponto de vista empresarial. Além disso, alia estas capacidades a uma sólida experiência no domínio. No livro *How Innovation Works: And Why It Flourishes in Freedom* (Harper), o autor Matt Ridley explica claramente que as ideias não são a base da inovação, mas sim a combinação das ideias na execução. Essencialmente, funciona ou não funciona, e alguém te pagará por isso?

Kaizen versus KaizenML

Um problema quando se fala de ciência de dados, AutoML e MLOps (KaizenML) é que as pessoas muitas vezes não entendem o que cada um deles é. A ciência de dados não é uma solução, assim como a estatística não é uma solução para um problema; ela é comportamental. O AutoML é apenas uma técnica, como a integração contínua (CI); automatiza tarefas triviais. Consequentemente, o AutoML não está a competir diretamente com a ciência dos dados; o cruise control, ou a condução semi-autónoma, não compete com o condutor de um automóvel. O condutor continua a ter de controlar o veículo e agir como árbitro central do que acontece. Da

mesma forma, mesmo com uma automatização extensiva na aprendizagem automática, um ser humano tem de tomar decisões executivas sobre o panorama geral.

KaizenML/MLOps é uma metodologia de sistemas que conduz a modelos em produção. Os modelos de aprendizagem automática que funcionam na produção e resolvem os problemas dos clientes são o resultado do MLOps. Na [Figura 5-5](#), podes ver uma hipotética revolução industrial MLOps que pode ocorrer no futuro. Os dados e a experiência no seu tratamento eficaz tornam-se uma vantagem competitiva, uma vez que são um recurso escasso. À medida que a tecnologia AutoML progride, é possível que muitas coisas que os cientistas de dados fazem atualmente desapareçam. É raro encontrar veículos modernos sem uma forma de controlo de velocidade de cruzeiro ou com transmissão manual. Da mesma forma, pode ser pouco frequente que os cientistas de dados ajustem os hiperparâmetros no futuro. O que pode acontecer então é que os actuais cientistas de dados se transformem em engenheiros de ML ou especialistas no domínio que "fazem ciência de dados" como parte do seu trabalho.

MLOps industrial revolution

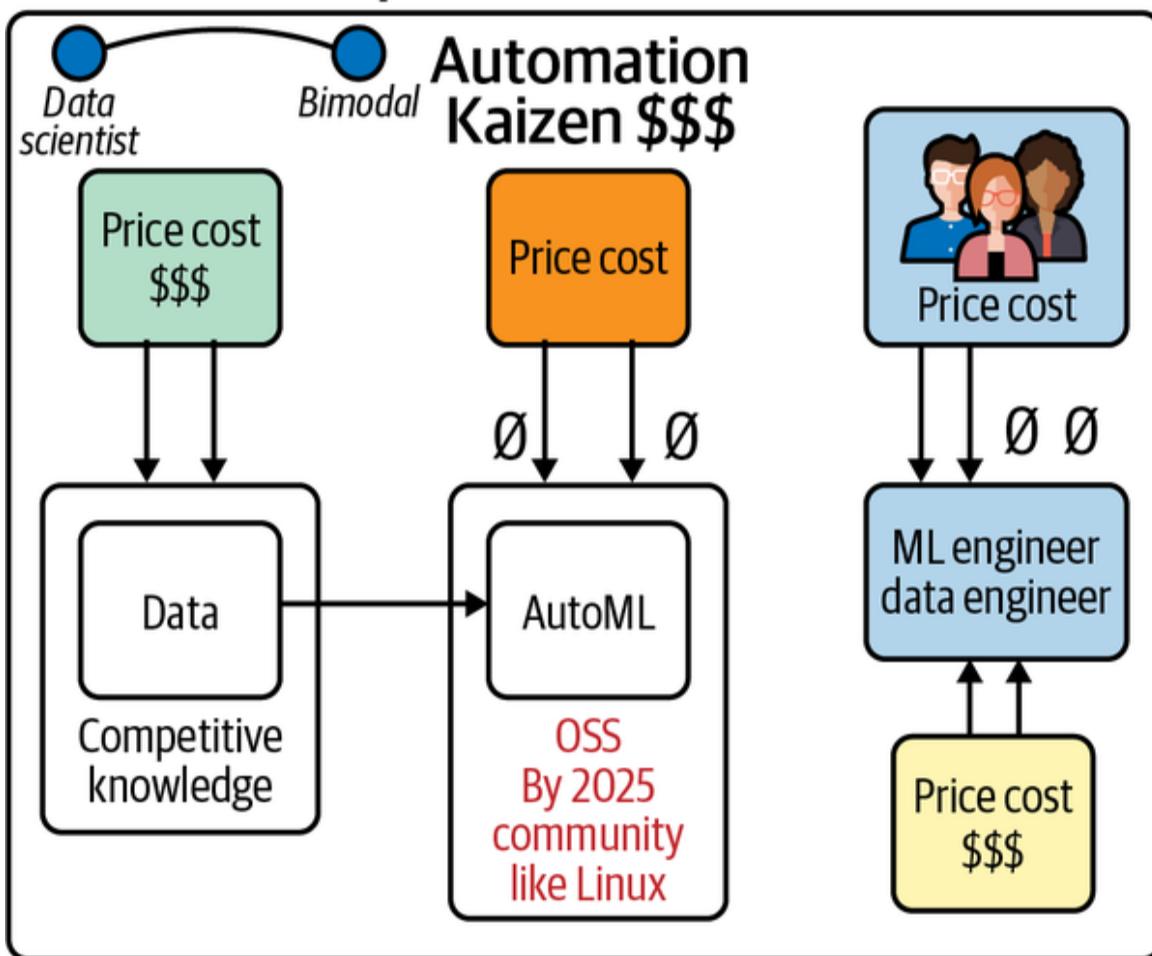


Figura 5-5. Revolução industrial dos MLOps

Um problema em falar apenas de AutoML versus ciência de dados é que banaliza as questões mais significativas de automação e melhoria contínua. A automação das técnicas de aprendizado de máquina é tão polarizada que a questão central desaparece: tudo deve ser automatizado, não apenas os aspectos tediosos do ML, como o ajuste de hiperparâmetros. A automatização através da melhoria contínua permite que os cientistas de dados, os engenheiros de ML e as organizações inteiras se concentrem no que é importante, ou seja, na execução. Como podes ver ilustrado na [Figura 5-6](#), Kaizen é um termo japonês para melhoria contínua. No pós-Segunda Guerra Mundial, o Japão construiu sua indústria automobilística em torno desse conceito. Essencialmente, se encontrares algo partido ou não optimizado, conserta-o. Da mesma forma, com o KaizenML, todos os

aspectos do aprendizado de máquina, da engenharia de recursos ao AutoML, são aprimorados.

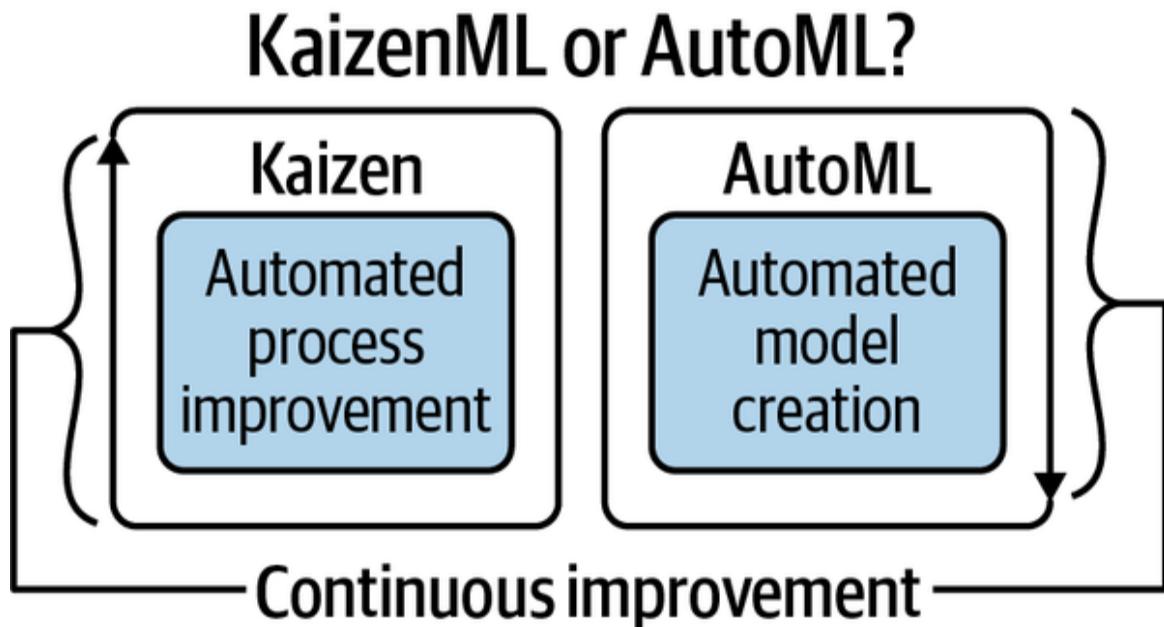


Figura 5-6. Kaizen ou AutoML

Todos os seres humanos no mundo deveriam fazer ciência de dados e programação porque estas são formas de pensamento crítico. A recente pandemia foi uma grande chamada de atenção para a importância de compreender a ciência dos dados para a vida de uma pessoa. Muitas pessoas morreram porque não compreenderam os dados que mostravam que a COVID-19 não era, de facto, igual à gripe; era muito, muito mais mortal. Da mesma forma, abundam histórias de pessoas que se recusam a tomar uma vacina porque calcularam incorretamente o risco que a vacina representava para si próprias e o risco que a COVID-19 representava para si próprias ou para os membros vulneráveis da sua comunidade. Compreender a ciência dos dados pode salvar a tua vida e, por isso, qualquer pessoa deve ter acesso às ferramentas que os cientistas de dados têm.

Estas ferramentas são "direitos humanos" que não pertencem às mãos de um sacerdócio de elite. É um non sequitur sugerir que as pessoas da "elite" só podem escrever programas simples, compreender a aprendizagem automática ou fazer ciência de dados. A automatização tornará a ciência dos dados e a programação suficientemente fáceis para que todos os seres

humanos o façam e, em muitos casos, podem fazê-lo mesmo utilizando a automatização existente.

O KaizenML/MLOps tem a ver com um foco estreito na resolução de problemas com aprendizagem automática e engenharia de software (influenciado pelo DevOps) para levar ao valor comercial ou à melhoria da condição humana, por exemplo, curar o cancro.

Lojas de artigos

Todos os sistemas de software complexos requerem automação e simplificação de componentes críticos. O DevOps consiste em automatizar os testes e a implementação de software. O MLOps tem a ver com isto e também com a melhoria da qualidade dos dados e dos modelos de aprendizagem automática. Anteriormente, chamei a estas melhorias contínuas dos dados e dos modelos de aprendizagem automática KaizenML. Uma forma de pensar sobre isto é que DevOps + KaizenML = MLOps. O KaizenML inclui a criação de Armazéns de Funcionalidades, ou seja, um registo de inputs de aprendizagem automática de alta qualidade e a capacidade de monitorizar os dados para detetar desvios e registar e fornecer modelos de ML.

Na [Figura 5-7](#), repara que na ciência de dados manual, tudo é feito à medida. Como resultado, os dados são de baixa qualidade e é difícil chegar ao ponto em que um modelo funcional entra em produção e resolve um problema. No entanto, à medida que mais coisas são automatizadas, desde os dados até às funcionalidades, através de um Armazém de Funcionalidades, até à utilização efectiva do modelo em produção, obtém-se um melhor resultado.

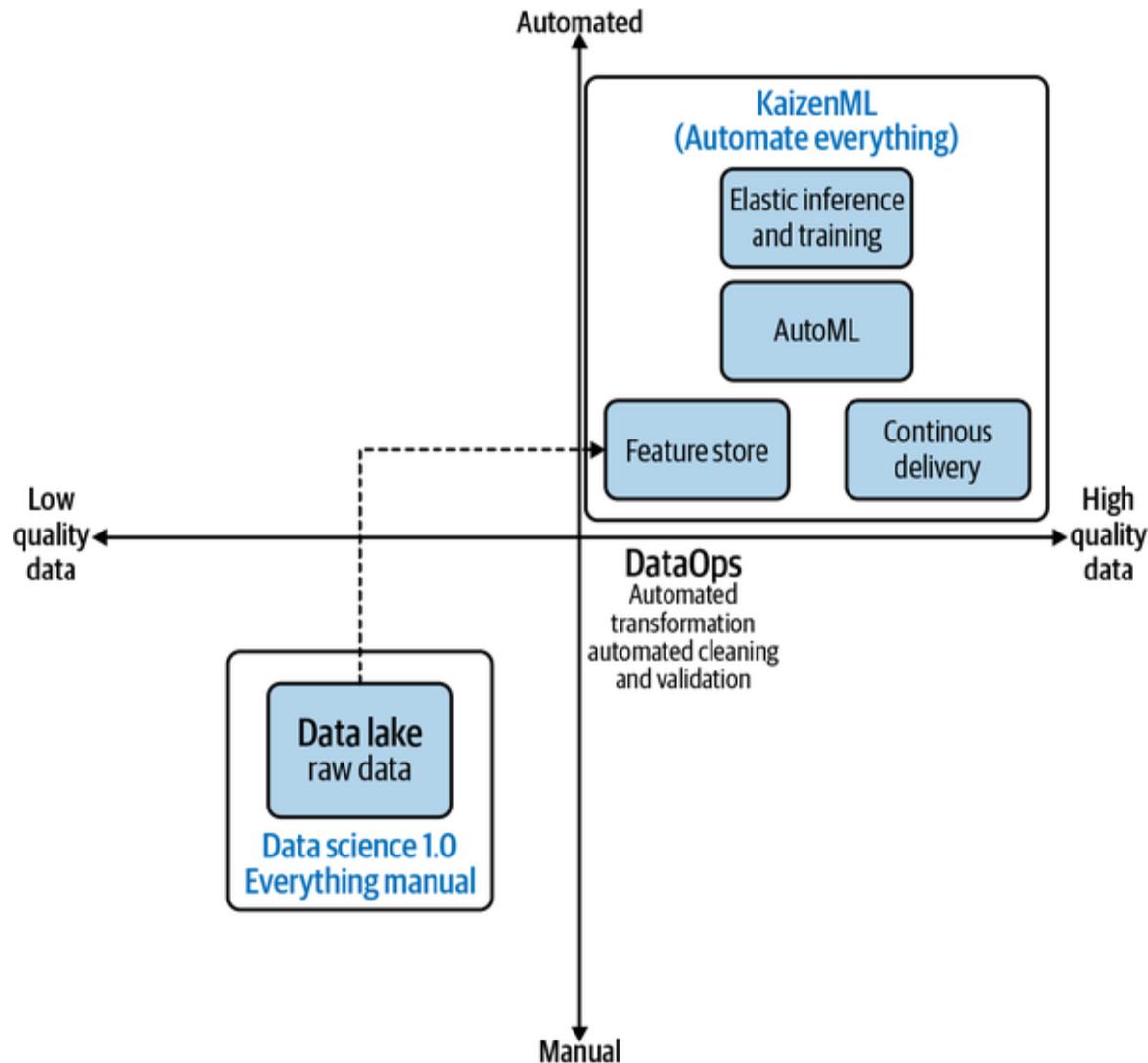


Figura 5-7. Armazenamento de caraterísticas como parte do KaizenML

Fortemente relacionado com o KaizenML, ou seja, com a melhoria contínua da aprendizagem automática, está o conceito de um Armazém de Funcionalidades. O blogue Uber Engineering tem uma boa descrição do problema que uma **Feature Store resolve**. De acordo com a Uber, faz duas coisas:

- Permite que os utilizadores adicionem funcionalidades que criaram a um Armazém de Funcionalidades partilhado.
- Quando as caraterísticas estão no Armazém de Caraterísticas, são fáceis de utilizar na formação e na previsão.

Na [Figura 5-8](#), podes ver que a ciência de dados é um comportamento, mas o AutoML é uma técnica. O AutoML pode ser apenas 5% de todo o problema resolvido pela automação. Os dados em si precisam de automação por meio do gerenciamento de tarefas ETL. O Feature Store precisa de automação para melhorar as entradas do ML. Por fim, a implementação requer automação através da implementação automatizada (CD) e da utilização nativa da elasticidade da Cloud (autoscaling). Tudo requer automação com sistemas de software complexos, e os Armazéns de Recursos são apenas um dos muitos componentes MLOps que necessitam de melhoria contínua, ou seja, KaizenML.

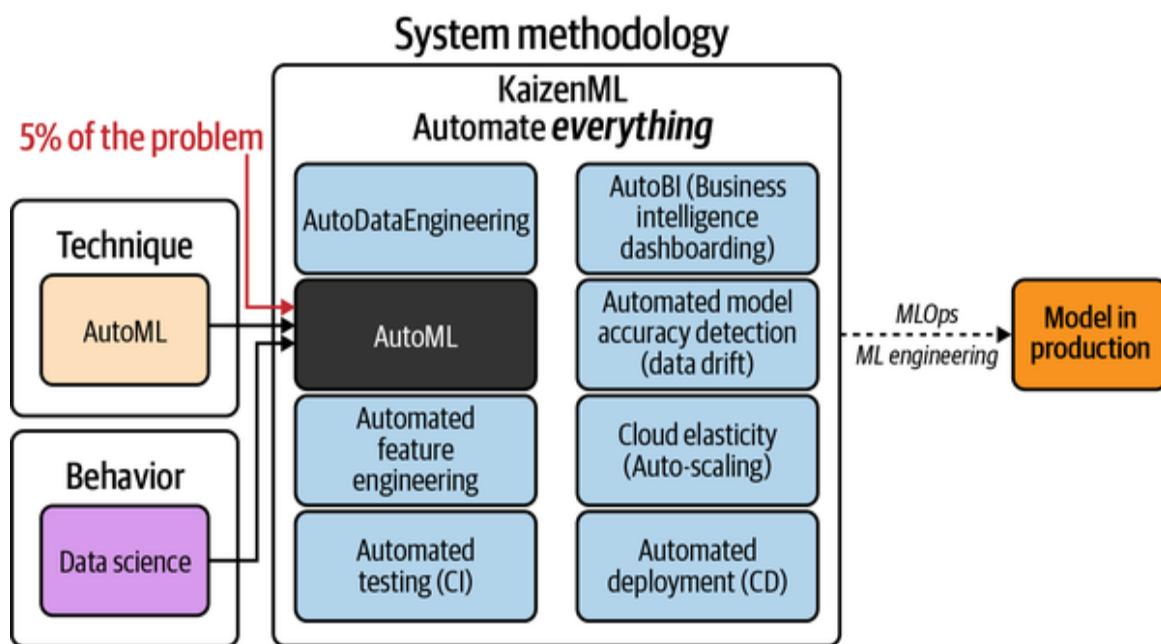


Figura 5-8. Os depósitos de características fazem parte de uma metodologia de automatização de sistemas

Existem muitos casos de utilização no mundo real para as Feature Stores. Por exemplo, a Uber explica que [utilizou 10.000 funcionalidades no Armazém de Funcionalidades](#) para acelerar projectos de máquinas e criar soluções AutoML em cima. Além disso, plataformas como [a Databricks](#) têm Feature Stores integrados no seu sistema de grandes volumes de dados. Por exemplo, na [Figura 5-9](#), podes ver como os dados em bruto são a entrada que é transformada num registo de características mais refinado e especializado que pode resolver problemas em lote e online.

Na [Figura 5-10](#), repara que existem semelhanças e diferenças entre um armazém de dados tradicional e um MLOps Feature Store. Um data warehouse alimenta os sistemas de business intelligence a um nível elevado e um Feature Store fornece entradas para um sistema de ML. O processamento de dados de aprendizagem automática é repetitivo, incluindo a normalização de dados, a limpeza de dados e a procura de características adequadas que melhorem um modelo de ML. A criação de um sistema de armazenamento de características é mais uma forma de adotar plenamente a automatização de todo o processo de aprendizagem automática, desde a conceção até à produção .

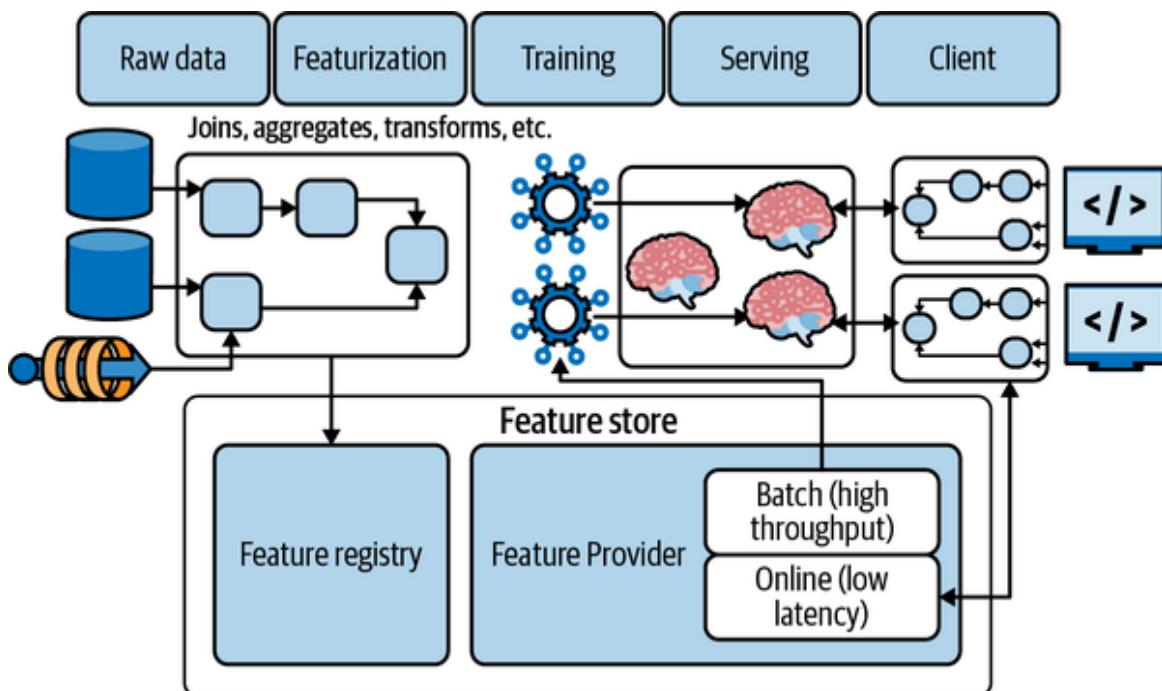


Figura 5-9. Armazenamento de recursos do Databricks

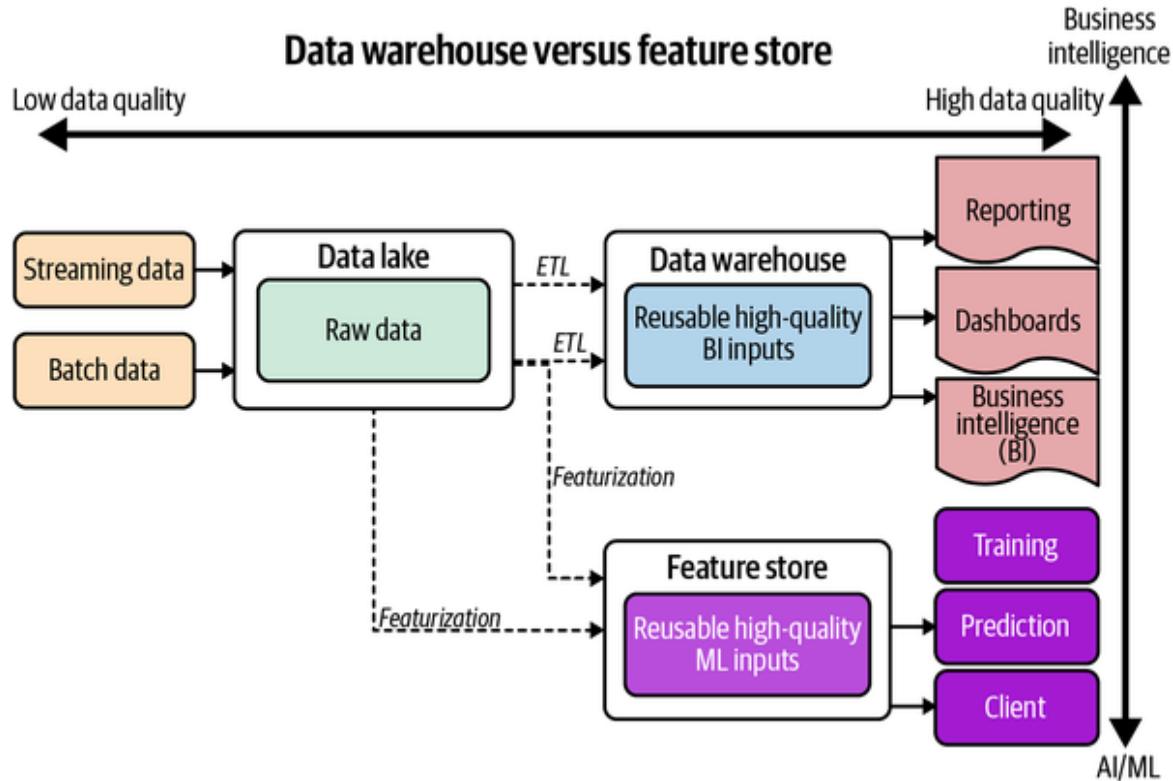


Figura 5-10. Data warehouse versus feature store

Em seguida, vamos sair da teoria e da técnica prática utilizando o ecossistema de ML da Apple para criar modelos de aprendizagem automática. Fá-lo-emos utilizando a sua estrutura AutoML de alto nível, CreateML.

O ecossistema da Apple

A Apple pode parecer um candidato improvável para entrar no espaço das ferramentas de aprendizagem automática, até olhares um pouco mais a fundo. A Apple tem um ecossistema rico em torno do desenvolvimento móvel. De acordo com o [Statista](#), a receita bruta mundial de aplicações da Apple App Store cresceu de 55,5 mil milhões de dólares americanos em 2019 para 72,3 mil milhões de dólares americanos em 2020. A Apple beneficia dos programadores que criam produtos que vendem na sua loja de aplicações.

Lembro-me de falar com um professor bastante desdenhoso sobre a construção de "aplicações para aprendizagem automática", presumivelmente porque ele gravitava em torno da complexidade e da descoberta na pesquisa. De certa forma, a indústria do software pensa de forma oposta a um investigador numa universidade. Escrever artigos académicos sobre aprendizagem automática é o oposto de operacionalizar a aprendizagem automática para "construir aplicações". Essa diferença ideológica é a desconexão entre "ideia" e "execução" discutida anteriormente.

A Apple quer que crie aplicações na sua loja de aplicações, uma vez que recebe entre 15% e 30% de cada transação. Quanto melhor a Apple fizer as ferramentas para programadores, mais aplicações estarão disponíveis na loja de aplicações. Há uma expressão na escola de gestão: "Onde é que constróis um Burger King? Ao lado do McDonald's". Esta expressão é uma forma de dizer que não precisas de gastar o dinheiro para pesquisar onde expandir, porque o principal concorrente já fez o trabalho. Podes aproveitar a sua experiência - da mesma forma, os profissionais da aprendizagem automática podem aproveitar a pesquisa que a Apple fez. Vê um futuro na aprendizagem automática de alto nível executada em hardware especializado.

Do mesmo modo, porque é que muitas empresas de capital de risco financiam apenas empresas que as empresas de capital de risco de topo já financiaram? Porque assim não precisam de fazer nenhum trabalho; podem beneficiar da experiência de uma empresa com mais conhecimentos. Da mesma forma, a Apple tem feito enormes investimentos, do ponto de vista do hardware, na aprendizagem automática no dispositivo. Em particular, a Apple desenvolve ela própria chips, como os da série A: A12-A14, mostrada na [Figura 5-11](#), incluindo CPUs, GPUs e hardware de rede neural dedicado.



Figura 5-11. O chip A14 da Apple

Além disso, os chips mais recentes incluem a arquitetura Apple M1, que a Apple utiliza em dispositivos móveis, computadores portáteis e computadores de secretária, conforme ilustrado na Figura 5-12.



Figura 5-12. O chip M1 da Apple

O ambiente de desenvolvimento utiliza esta tecnologia através do formato de modelo **Core ML** da Apple. Existe também um pacote Python para converter modelos de bibliotecas de treino de terceiros, como o TensorFlow e o Keras.

O Core ML está optimizado para o desempenho no dispositivo e funciona em conjunto com o hardware da Apple. Há vários fluxos de trabalho não óbvios a considerar:

- Utiliza a estrutura Create ML da Apple para criar soluções AutoML.

- Transfere um modelo pré-treinado e, opcionalmente, converte-o para o formato Core ML. Um local para descarregar modelos é o [tfhub](#).
- Treina um modelo escrevendo o código noutra estrutura e convertendo-o para o Core ML utilizando [o coremltools](#).

Vamos analisar o AutoML da Apple.

AutoML da Apple: Cria ML

Uma das principais inovações da plataforma ML da Apple é a forma como expõe a poderosa tecnologia AutoML incluída numa GUI intuitiva. O Apple Create ML permite-te fazer o seguinte:

- Cria modelos Core ML
- Pré-visualiza o desempenho do modelo
- Treina modelos no Mac (tirando partido da sua pilha de chips M1)
- Utiliza o controlo da formação: ou seja, pausa, guarda e retoma a formação
- Usa eGPU (GPUs externas)

Além disso, lida com vários domínios de imagem, vídeo, movimento, som, texto e tabela. Vamos mergulhar no AutoML com o CreateML da Apple. Repara na lista completa de muitas formas automatizadas de aprendizagem automática na [Figura 5-13](#) e como acabam por convergir para o mesmo modelo Core ML que corre no iOS.

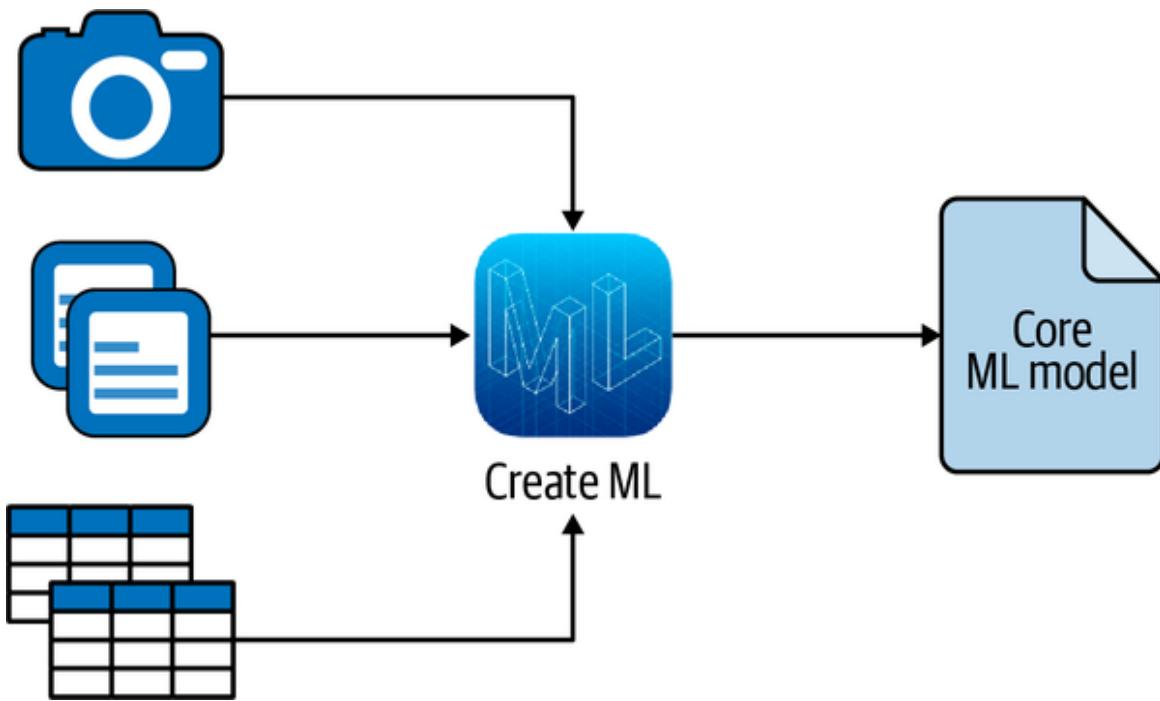


Figura 5-13. Criar ML

Para começar a utilizar o Criar ML, faz o seguinte:

1. Transfere o XCode.
2. Abre o XCode e clica com o botão direito do rato no ícone para iniciar o Create ML([Figura 5-14](#)).

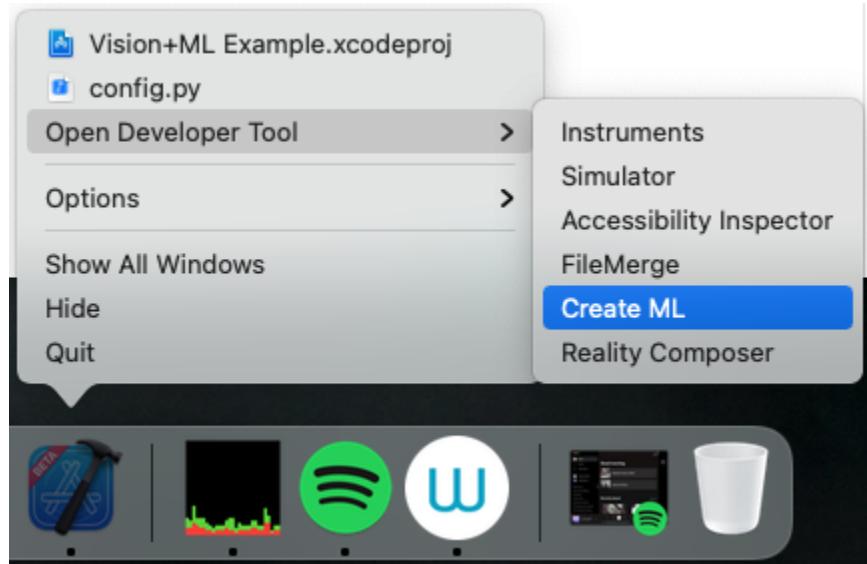


Figura 5-14. Abre Criar ML

Em seguida, utiliza o modelo Image Classifier (Classificador de imagens) (ver [Figura 5-15](#)).

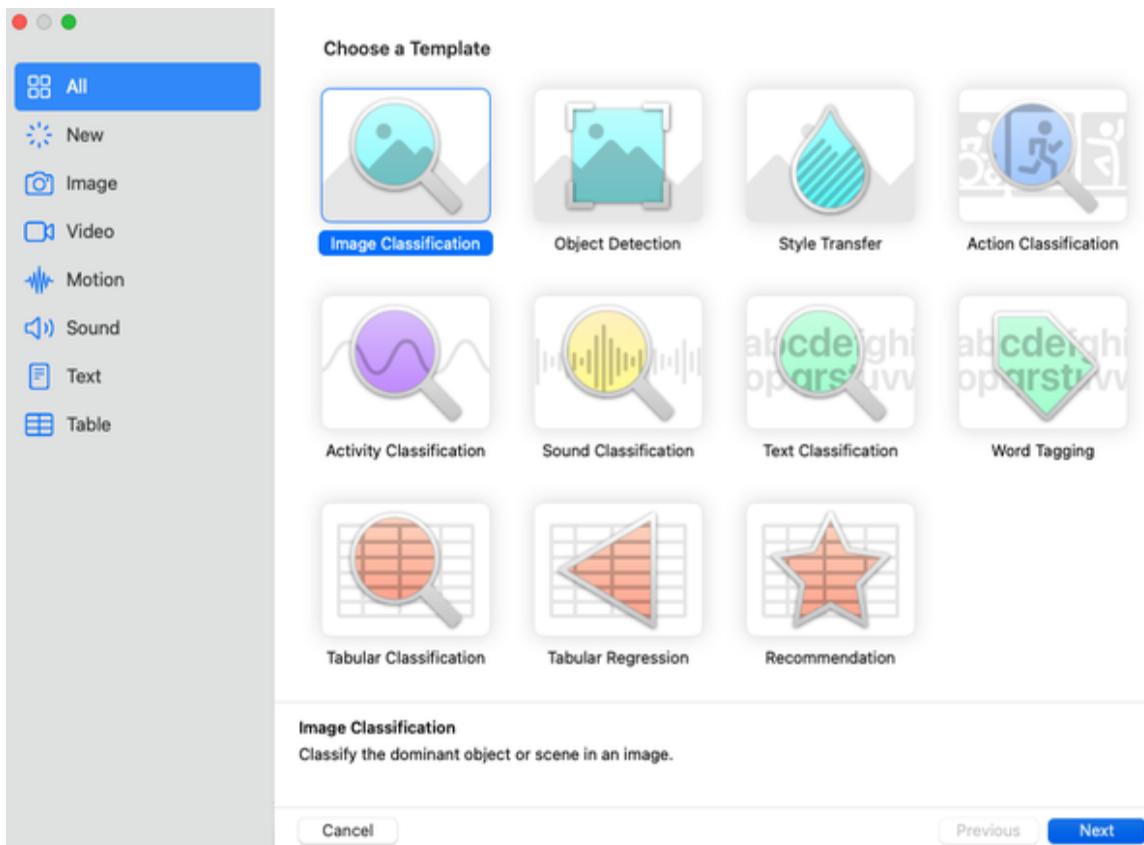


Figura 5-15. Modelo de classificador de imagens

Podes obter uma versão mais pequena do conjunto de dados Kaggle "cats and dogs" no [repositório GitHub do livro](#). Solta o conjunto de dados `cats-dogs-small` na IU do Create ML (consulte a [Figura 5-16](#)).

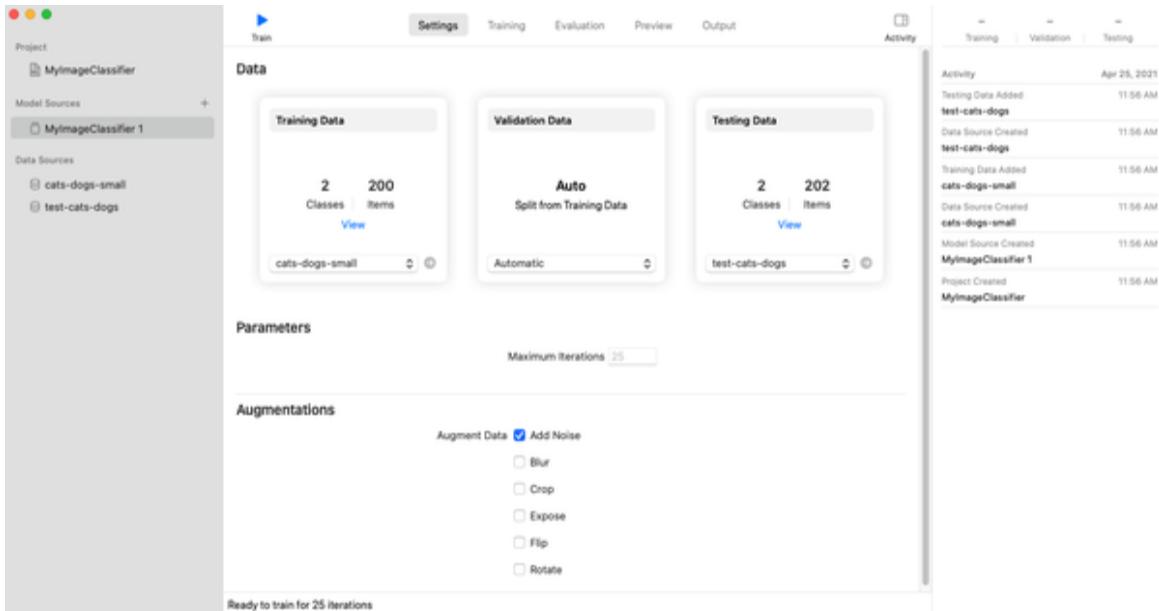


Figura 5-16. Carrega dados

Além disso, **coloca os dados de teste na secção de teste da interface do utilizador para Criar ML.**

Em seguida, treina o modelo clicando no ícone treinar. Nota que podes treinar o modelo várias vezes, clicando com o botão direito do rato em Model Sources. Podes querer fazer experiências com isto porque te permite testar com "Augmentations" como Noise, Blur, Crop, Expose, Flip e Rotate (ver **Figura 5-17**). Isto permitir-te-á criar um modelo mais robusto que seja mais generalizável em relação aos dados do mundo real.

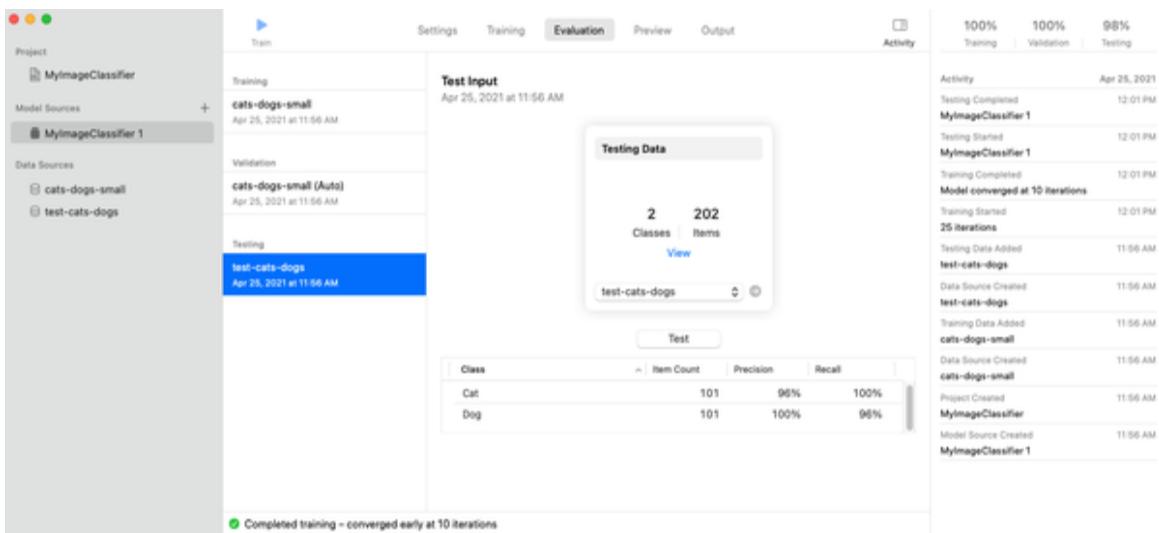


Figura 5-17. Modelo treinado

Este pequeno conjunto de dados deve demorar apenas alguns segundos a treinar o modelo (especialmente se tiveres o mais recente hardware Apple M1). Podes testá-lo encontrando imagens de gatos e cães na Internet, descarregando-as e arrastando-as para o ícone de pré-visualização (ver Figura 5-18).

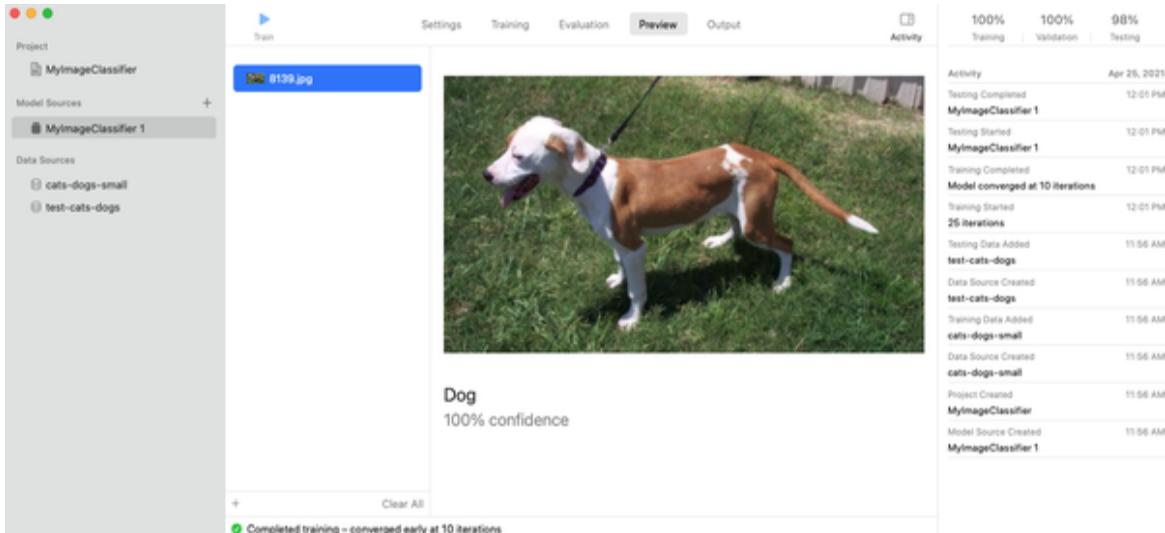


Figura 5-18. Pré-visualização

Um passo final é descarregar o modelo e usá-lo numa aplicação iOS. Repara que, na Figura 5-19, uso o menu Finder do OS X, dou um nome ao modelo e guardo-o no meu ambiente de trabalho. Este passo final pode ser o passo terminal para um amador que queira criar uma aplicação iOS personalizada que funcione apenas no seu telefone. Depois de guardares o modelo, podes opcionalmente convertê-lo para outro formato como **ONNX** e depois executá-lo numa plataforma Cloud como o Microsoft Azure.

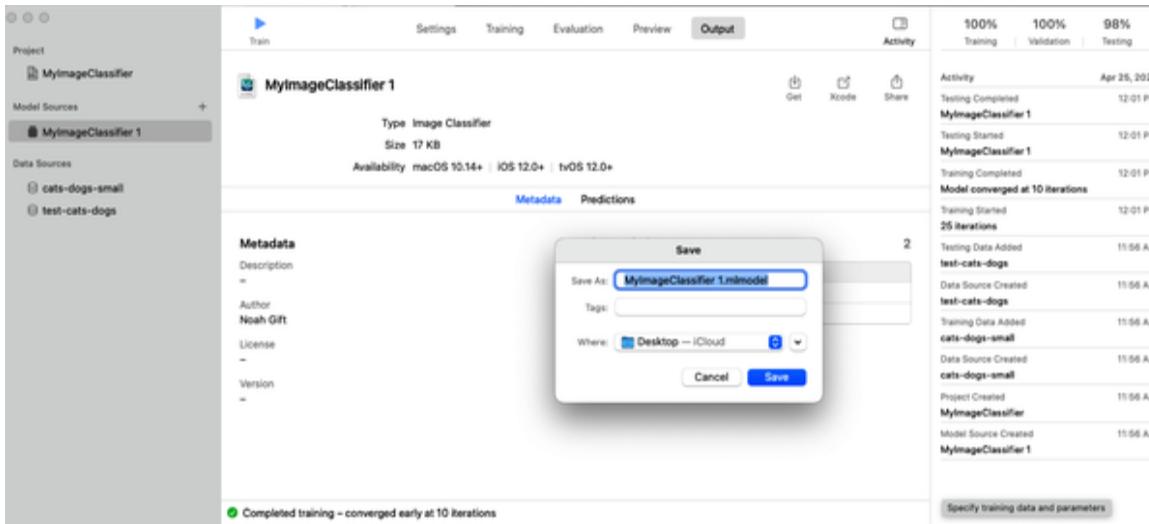


Figura 5-19. Cria um modelo ML

Trabalha muito bem! Treinaste o teu primeiro modelo que não necessitava de código. O futuro será fantástico à medida que mais destas ferramentas evoluírem e chegarem às mãos dos consumidores.

Opta por dar os próximos passos:

- Podes treinar um modelo mais complexo **descarregando um conjunto de dados Kaggle maior**
- Podes experimentar outros tipos de AutoML
- Podes fazer experiências com aumentos

Agora que já sabes como treinar um modelo utilizando o Create ML , vamos aprofundar um pouco mais a forma como podes tirar partido das ferramentas Core ML da Apple.

As principais ferramentas de ML da Apple

Um dos fluxos de trabalho mais interessantes disponíveis para o ecossistema Apple é o download de modelos e a sua conversão para as ferramentas Core ML através de uma biblioteca Python. Existem muitos locais para obter modelos pré-treinados, incluindo o TensorFlow Hub.

Neste exemplo, vamos percorrer o código **deste bloco de notas do Colab**.

Primeiro, instala a biblioteca coremltools:

```
!pip install coremltools
import coremltools
```

Em seguida, transfere o modelo (com base no [guia oficial de início rápido](#)).

Importa TensorFlow como tf:

```
# Download MobileNetv2 (using tf.keras)
keras_model = tf.keras.applications.MobileNetV2(
    weights="imagenet",
    input_shape=(224, 224, 3, ),
    classes=1000,
)
# Download class labels (from a separate file)
import urllib
label_url =
'https://storage.googleapis.com/download.tensorflow.org/\n    data/ImageNetLabels.txt'
class_labels =
urllib.request.urlopen(label_url).read().splitlines()
class_labels = class_labels[1:] # remove the first class which is
background
assert len(class_labels) == 1000

# make sure entries of class_labels are strings
for i, label in enumerate(class_labels):
    if isinstance(label, bytes):
        class_labels[i] = label.decode("utf8")
```

Converte o modelo e define os metadados para o modelo com os parâmetros corretos:

```
import coremltools as ct

# Define the input type as image,
# set preprocessing parameters to normalize the image
# to have its values in the interval [-1,1]
# as expected by the mobilenet model
image_input = ct.ImageType(shape=(1, 224, 224, 3, ),
                            bias=[-1, -1, -1], scale=1/127)

# set class labels
```

```
classifier_config = ct.ClassifierConfig(class_labels)

# Convert the model using the Unified Conversion API
model = ct.convert(
    keras_model, inputs=[image_input],
    classifier_config=classifier_config,
)
```

Agora atualiza os metadados do modelo:

```
# Set feature descriptions (these show up as comments in XCode)
model.input_description["input_1"] = "Input image to be
classified"
model.output_description["classLabel"] = "Most likely image
category"

# Set model author name
model.author = "" # Set the license of the model

# Set the license of the model
model.license = ""# Set a short description for the Xcode UI

# Set a short description for the Xcode UI
model.short_description = "" # Set a version for the model

# Set a version for the model
model.version = "2.0"
```

Finalmente, guarda o modelo, transfere-o do Colab e abre-o no XCode para fazer a previsão (ver [Figura 5-20](#)).

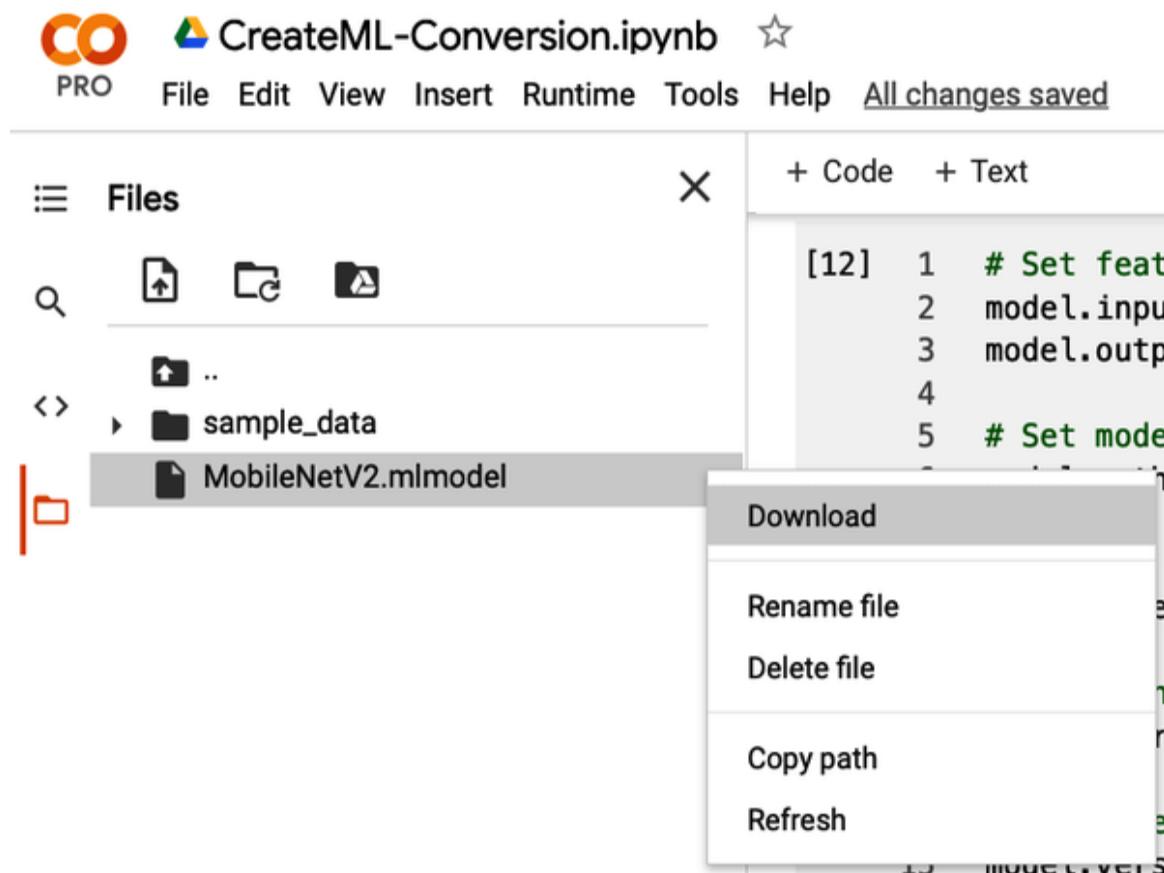


Figura 5-20. Descarrega o modelo

```
# Save model
model.save("MobileNetV2.mlmodel")

# Load a saved model
loaded_model = ct.models.MLModel("MobileNetV2.mlmodel")
```

A Figura 5-21 mostra um exemplo de previsão.

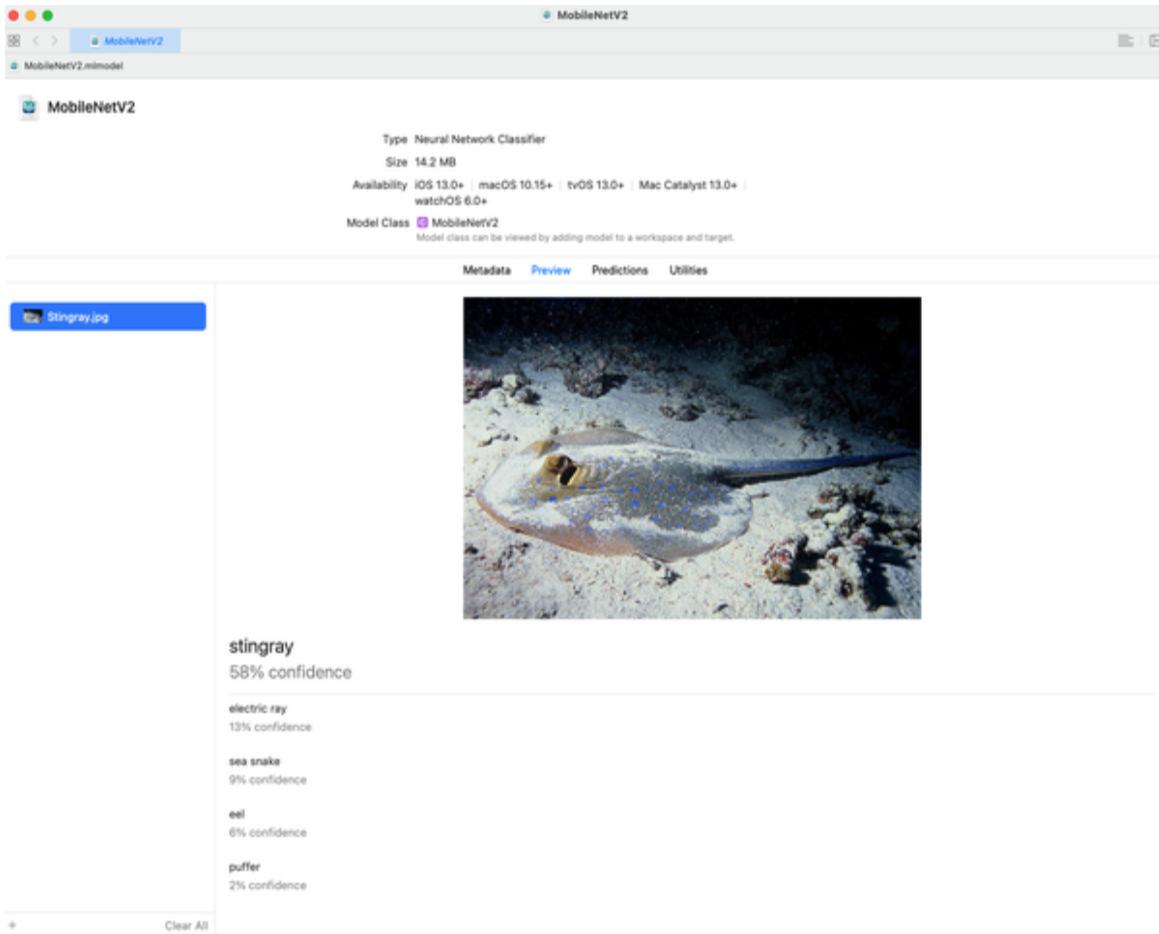


Figura 5-21. Previsão de arraia

A grande conclusão é que este processo é ainda mais fácil do que utilizar o AutoML. Por isso, pode fazer mais sentido descarregar um modelo criado por especialistas com acesso a clusters de computação dispendiosos do que treiná-lo tu próprio em muitos casos. A estrutura Core ML da Apple permite tanto o uso de AutoML personalizado como a utilização de modelos pré-treinados .

AutoML e visão computacional de ponta da Google

Nos últimos anos, ensinei centenas de alunos numa aula chamada "Visão computacional aplicada" nas melhores universidades de ciência de dados. A premissa do curso é criar soluções rapidamente utilizando as ferramentas de

mais alto nível disponíveis, incluindo o Google AutoML e hardware de ponta como o chip **Coral.AI** que contém uma TPU ou o Intel Movidius.

A Figura 5-22 mostra dois exemplos desoluções de aprendizagem automática de ponta com um formato pequeno.



Figura 5-22. Ferragem do bordo

Uma das coisas surpreendentes de dar esta aula é a rapidez com que os alunos conseguem pegar em soluções "prontas a usar", juntá-las e criar uma solução que resolve um problema. Vi projectos que incluíam a deteção de máscaras, a deteção de matrículas e aplicações de triagem de lixo a funcionar em dispositivos móveis com pouco ou nenhum código. Estamos

numa nova era, a era dos MLOps, e é mais fácil colocar código em aplicações funcionais.

Tal como a Apple e a Google, muitas empresas constroem uma pilha verticalmente integrada que fornece uma estrutura de aprendizagem automática, sistemas operativos e hardware especializado, como um ASIC (circuito integrado de aplicação específica) que executa tarefas específicas de aprendizagem automática. Por exemplo, a TPU, ou Unidade de Processamento TensorFlow, é desenvolvida ativamente com atualizações regulares no design do chip. A versão edge é um ASIC criado especificamente para executar modelos de ML. Esta integração estreita é essencial para as organizações que procuram a criação rápida de soluções de aprendizagem automática do mundo real.

Existem várias abordagens críticas à visão computacional na plataforma GCP (semelhante a outras plataformas Cloud, os nomes dos serviços são diferentes). Estas opções aparecem por ordem de dificuldade:

- Escreve código de aprendizagem automática que treina um modelo
- Utiliza o Google AutoML Vision
- Transfere um modelo pré-treinado do **TensorFlow Hub** ou de outro local
- Utiliza a **API do Vision AI**

Vamos examinar um fluxo de trabalho do Google AutoML Vision que termina num modelo de visão computacional implementado num dispositivo iOS. Este fluxo de trabalho é essencialmente o mesmo, quer utilizes um conjunto de dados de amostra fornecido pela Google ou o teu próprio:

1. Inicia a Consola Google Cloud e abre um shell de nuvem.
2. Ativa a API Google AutoML Vision e dá permissão ao teu projeto para a utilizar; define os endereços PROJECT_ID e USERNAME:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member="user:$USERNAME" \
--role="roles/automl.admin"
```

3. Carrega os dados de treino e as etiquetas através de um ficheiro CSV para o Google Cloud Storage.

Se definires um `${BUCKET}` variable `export `BUCKET=$FOOBAR`, só precisas de três comandos para copiar os dados de amostra do Google. Aqui está um exemplo de classificação de nuvens (cirrus, cumulonimbus, cumulus). Podes encontrar um passo a passo no Google Qwiklabs em "["Classificar imagens de nuvens na Cloud com visão AutoML"](#)". O local `gs://splts/gsp223/images/` contém os dados neste exemplo, e o comando `sed` troca os caminhos específicos:

```
gsutil -m cp -r gs://splts/gsp223/images/* gs://${BUCKET}
gsutil cp gs://splts/gsp223/data.csv .
sed -i -e "s/placeholder/${BUCKET}/g" ./data.csv
```

CONJUNTOS DE DADOS ADICIONAIS IDEAIS PARA O GOOGLE AUTOML

Dois outros conjuntos de dados que também podes tentar incluir são [os dados tf_flowers](#) e [os dados cats and dogs](#). Outra ideia é carregar os teus dados.

4. Inspecciona visualmente os dados.

Um dos aspectos valiosos dos sistemas Cloud AutoML da Google é a utilização de ferramentas de alto nível para inspecionar os dados, adicionar novas etiquetas ou corrigir problemas de controlo de qualidade dos dados. Observa na [Figura 5-23](#) que tens a capacidade de alternar entre as diferentes categorias de classificação, que por acaso são flores.

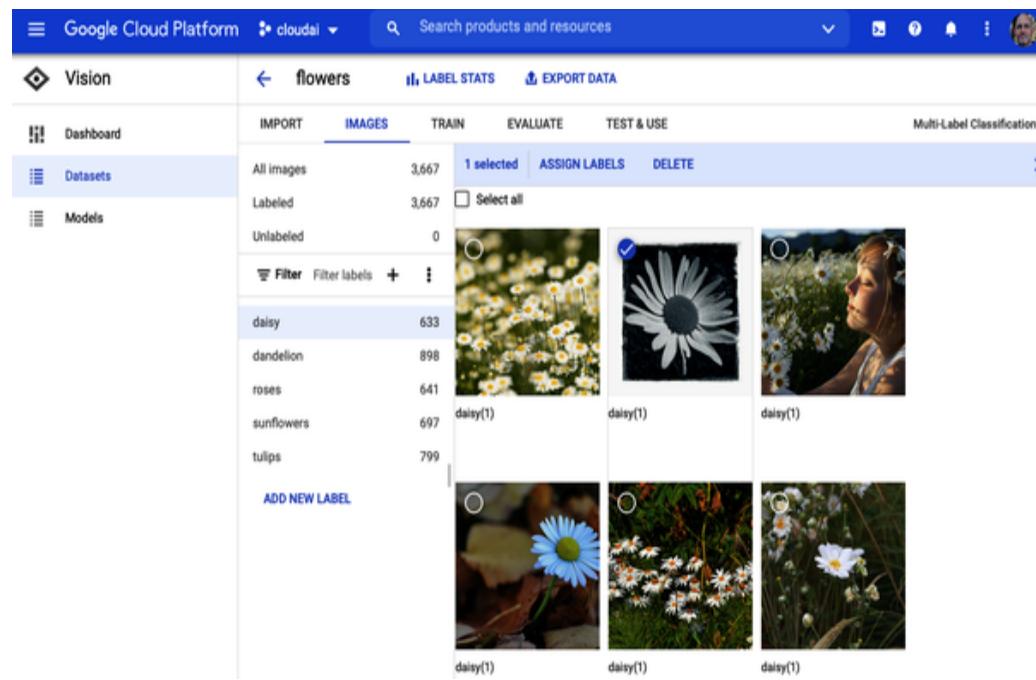


Figura 5-23. Inspecionar dados

5. Treina o modelo e avalia.

Treinar o modelo é um clique num botão da consola. O Google reuniu essas opções no seu produto **Google vértice AI**. Observa na [Figura 5-24](#) que existe uma série de ações, desde Notebooks a Previsões em lote, no painel esquerdo. Ao criar um novo trabalho de treinamento, o AutoML é uma opção, assim como o AutoML Edge.

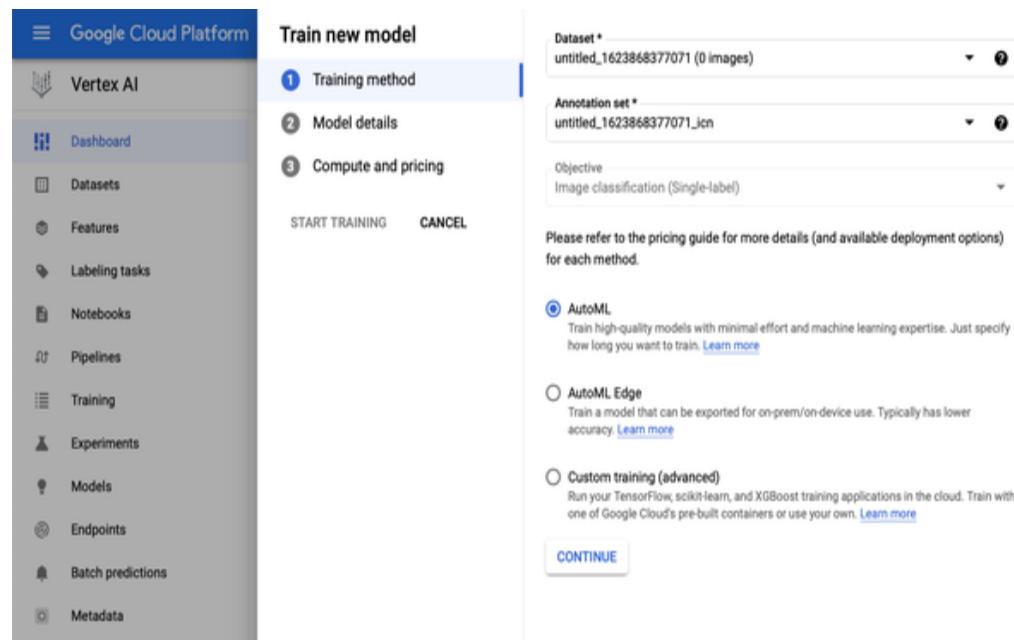


Figura 5-24. IA de vértice do Google

6. Em seguida, avalia o modelo treinado utilizando as ferramentas incorporadas (ver Figura 5-25).

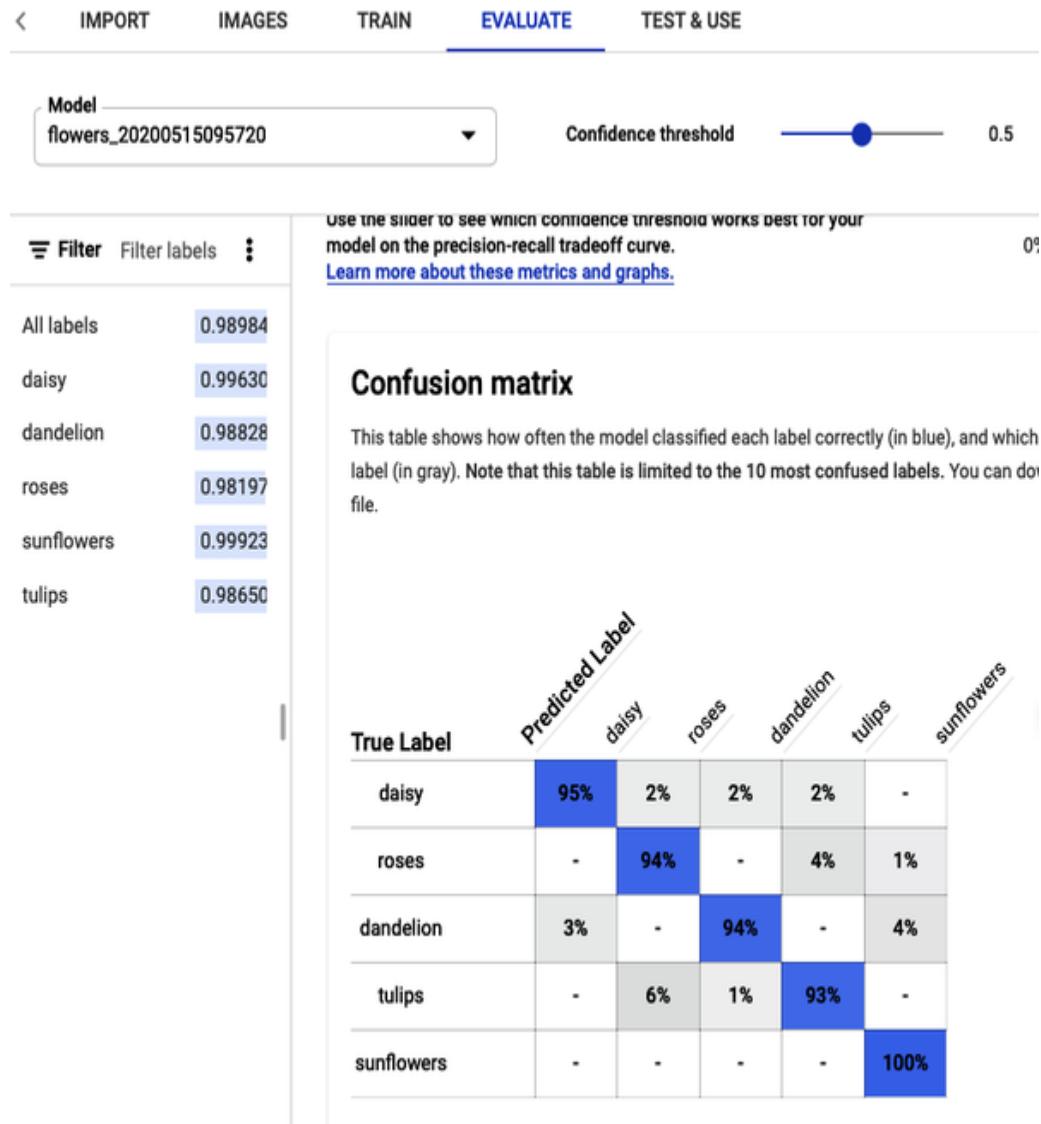


Figura 5-25. Avalia os dados

7. Faz alguma coisa com o modelo: prevê online ou descarrega.

Com a visão do Google AutoML, existe a possibilidade de criar um ponto de extremidade hospedado online ou de descarregar o modelo e fazer previsões num dispositivo de extremidade: iOS, Android, JavaScript, Coral Hardware ou um contentor (ver [Figura 5-26](#)).

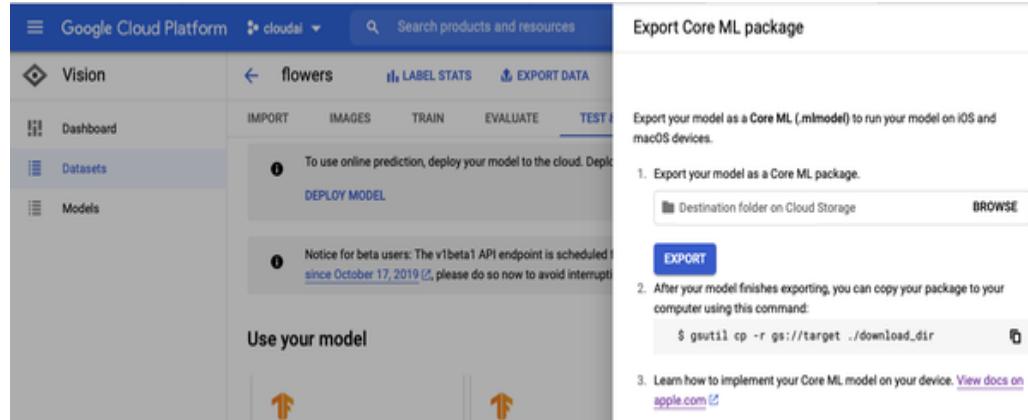


Figura 5-26. Descarrega o modelo

A principal conclusão é que o Google Cloud oferece um caminho bem percorrido desde o carregamento de dados de formação até à formação com o mínimo ou nenhum código necessário para criar soluções de aprendizagem automática que são implementadas em dispositivos de ponta. Estas opções estão todas integradas como parte da plataforma de aprendizagem automática gerida pela Google, a IA vértice.

Em seguida, vamos mergulhar nas soluções AutoML do Azure, que, tal como as da Google, têm uma história completa sobre a gestão do ciclo de vida dos MLOps.

AutoML do Azure

Existem duas formas principais de aceder ao Azure AutoML. Uma é a consola e a outra é o acesso programático ao **SDK Python** do AutoML. Vamos dar uma olhada no console primeiro.

Para começar a fazer o AutoML no Azure, tens de lançar uma instância do Azure ML Studio e selecionar a opção Automated ML (ver Figura 5-27).

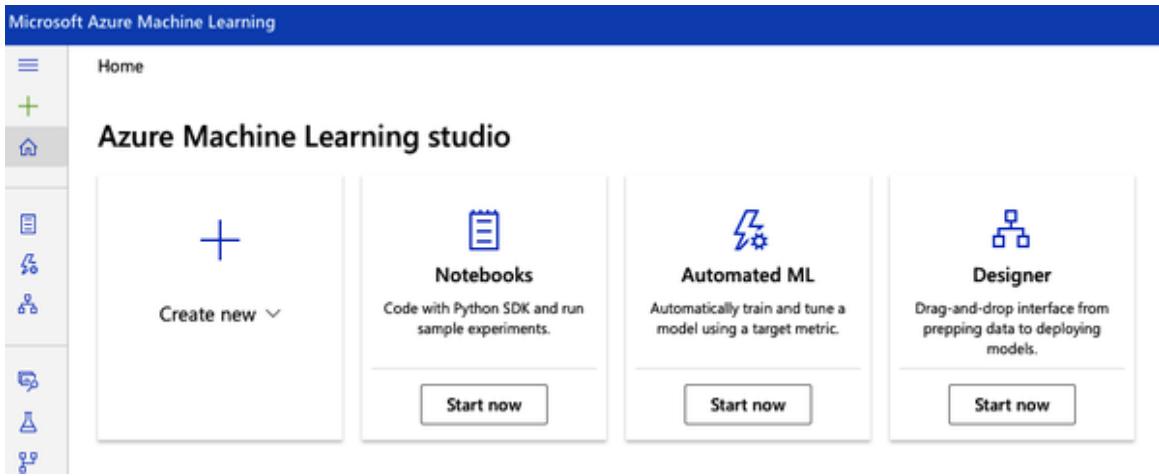


Figura 5-27. AutoML do Azure

Em seguida, cria um conjunto de dados carregando-o ou usando um conjunto de dados aberto. Neste exemplo, utilizo os dados do projeto [Kaggle Social Power NBA](#) (ver Figura 5-28).

Em seguida, executo uma tarefa de classificação para prever a posição em que um jogador pode jogar com base nas características do conjunto de dados. Nota que estão disponíveis muitos tipos diferentes de previsões de aprendizagem automática, incluindo regressão numérica e previsão de séries temporais. Terás de configurar o armazenamento e um cluster, se ainda não o tiveres feito (ver Figura 5-29).

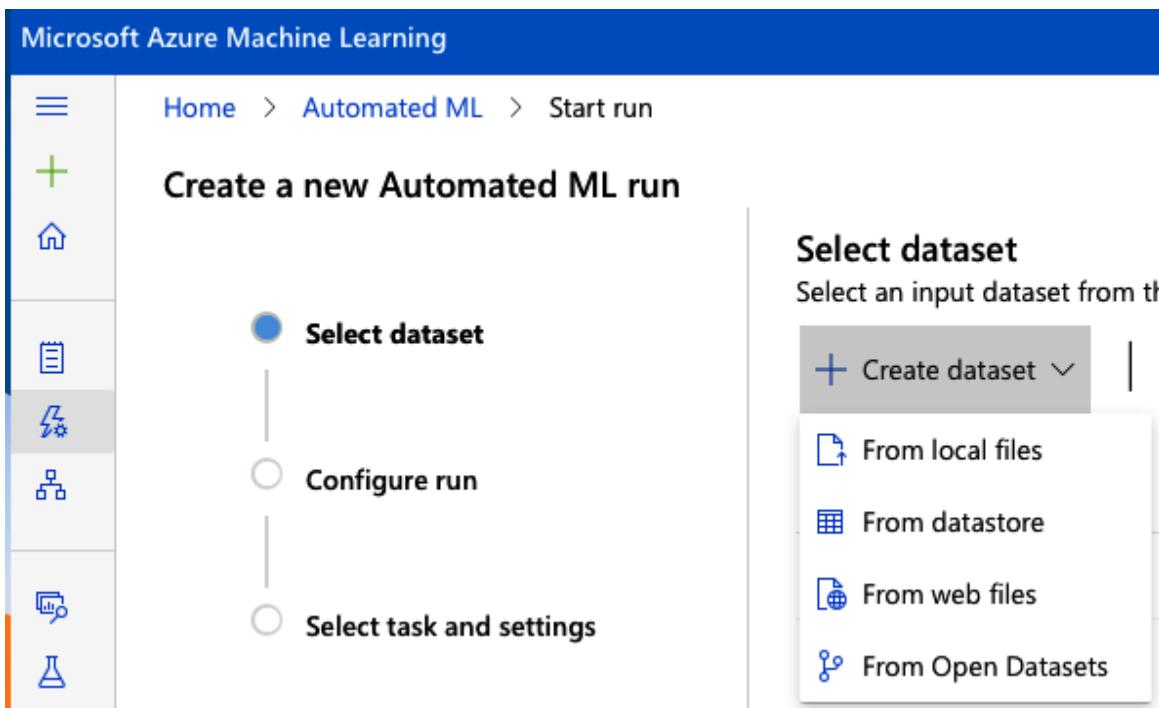


Figura 5-28. Cria um conjunto de dados do Azure AutoML

This screenshot shows the 'Select task type' configuration screen. The left sidebar shows the same three-step flow: 'Select dataset' (selected), 'Configure run', and 'Select task and settings'. The main area is titled 'Select task type' with the sub-instruction 'Select the machine learning task type for the experiment. To fine tune the experiment, choose additional featurization settings.' Below this, there are three sections: 'Classification' (selected, indicated by a checked checkbox), 'Regression', and 'Time series forecasting'. Each section has a brief description. At the bottom, there are links for 'View additional configuration settings' and 'View featurization settings'.

Figura 5-29. Classifica o Azure AutoML

Quando as tarefas estiverem concluídas, também podes pedir ao Azure ML Studio para "explicar" como chegou às suas previsões. Um sistema de aprendizagem automática explica como um modelo chega às previsões

através da "explicabilidade", que é uma capacidade crítica futura dos sistemas AutoML. Podes ver estas capacidades explicativas na [Figura 5-30](#). Observa como a integração profunda na solução ML Studio dá a esta tecnologia de plataforma uma sensação abrangente.

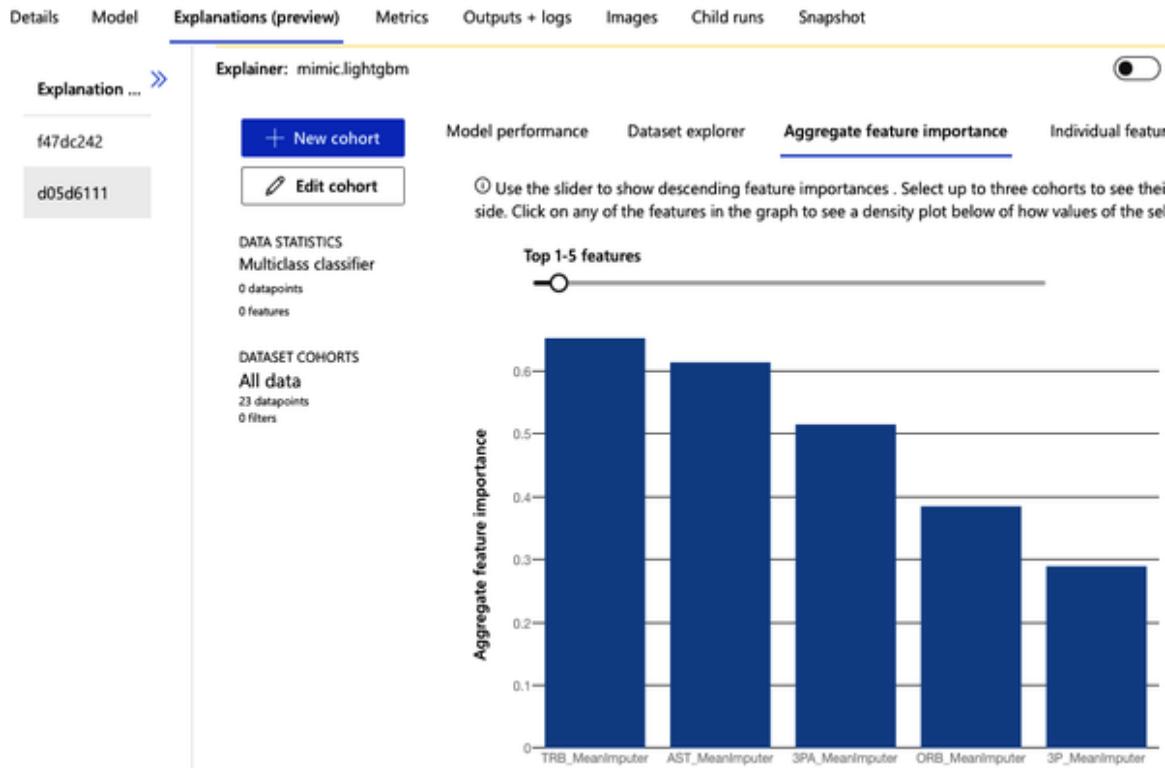


Figura 5-30. Explica o Azure AutoML

Vamos dar uma vista de olhos à outra abordagem. Podes utilizar o Python para chamar a mesma API disponível na consola do Azure ML Studio. Este [tutorial oficial da Microsoft](#) explica-o em pormenor, mas a secção crítica é mostrada aqui:

```
from azureml.train.automl import AutoMLConfig

automl_config = AutoMLConfig(task='regression',
                             debug_log='automated_ml_errors.log',
                             training_data=x_train,
                             label_column_name="totalAmount",
                             **automl_settings)
```

AWS AutoML

Como o maior fornecedor de Cloud, a AWS também tem muitas soluções de AutoML. Uma das primeiras soluções inclui uma ferramenta com um mau nome, "Machine Learning", que já não está amplamente disponível, mas que era uma solução AutoML. Agora a solução recomendada é SageMaker AutoPilot([Figura 5-31](#)). Pode ver muitos exemplos do SageMaker Autopilot em ação na [documentação oficial](#).

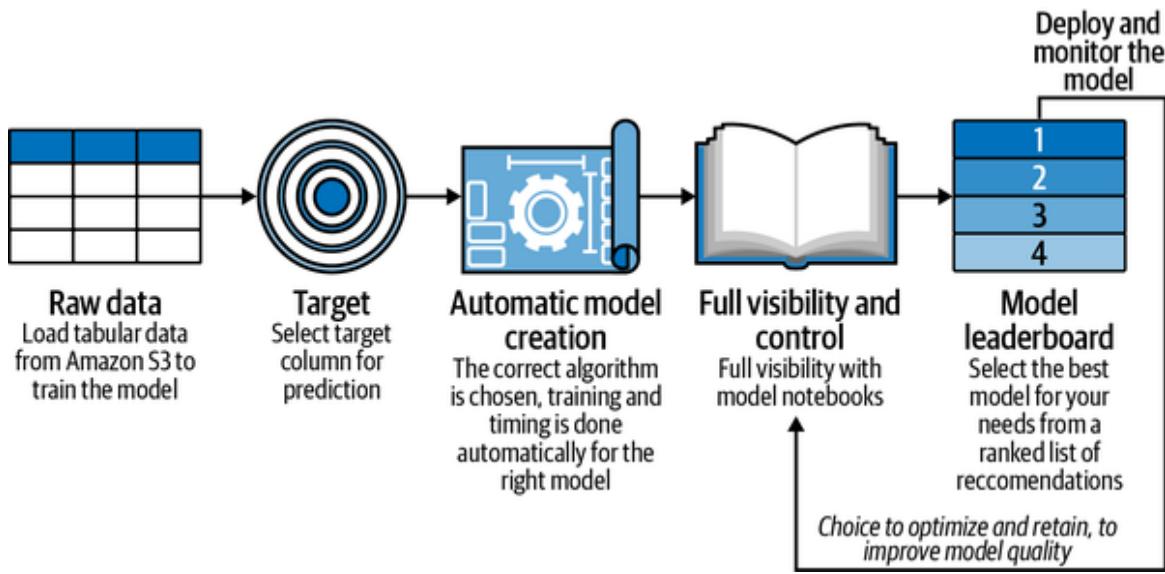


Figura 5-31. Piloto automático do SageMaker

Vamos ver como fazer uma experiência de piloto automático com o AWS SageMaker. Primeiro, como mostrado na [Figura 5-32](#), abre o SageMaker Autopilot e seleciona uma nova tarefa.

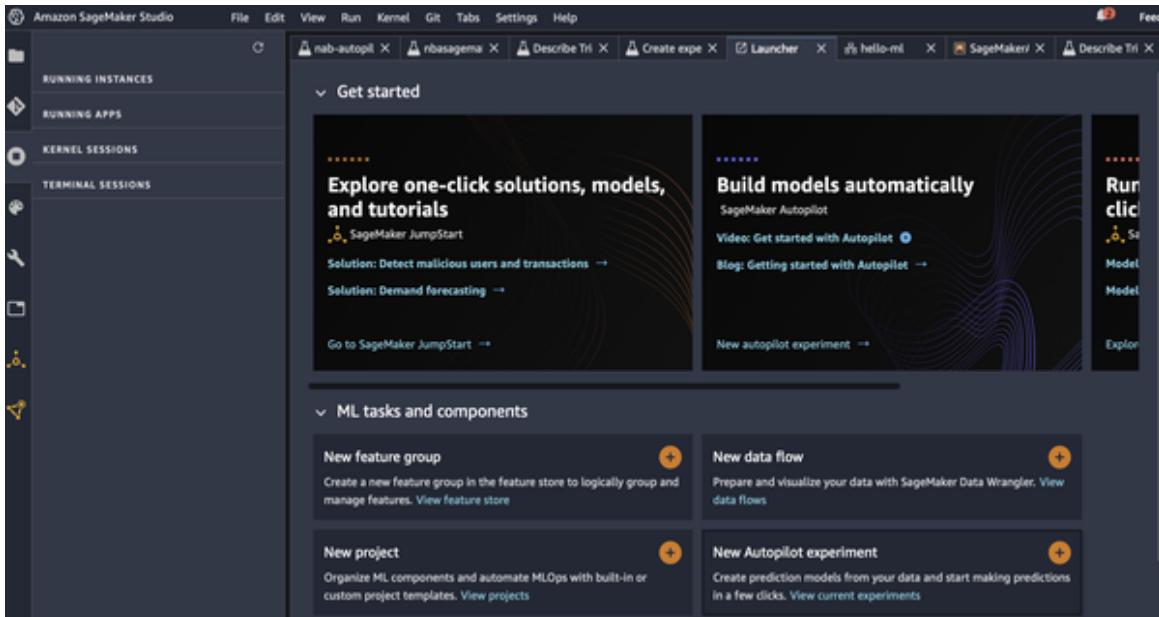


Figura 5-32. Tarefa do piloto automático do SageMaker

Em seguida, carrego o "Projeto Kaggle de dados de jogadores da NBA" no Amazon S3. Agora que tenho dados para trabalhar, crio um experimento como mostrado na Figura 5-33. Observa que selecionei como alvo a posição no draft. Essa classificação é porque quero criar um modelo de previsão que mostre qual posição no draft um jogador da NBA merece com base em seu desempenho.

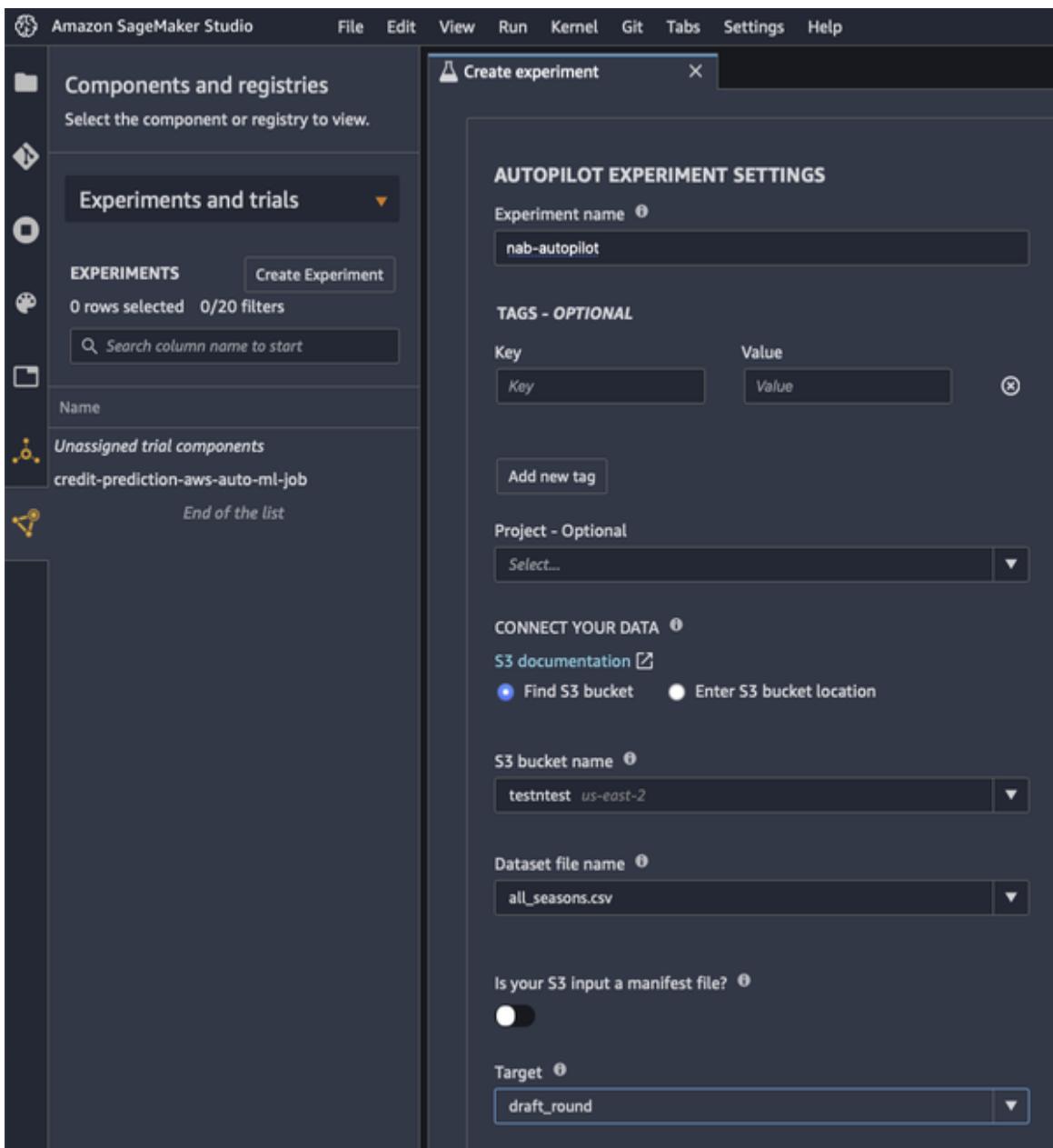


Figura 5-33. Cria uma experiência de piloto automático

Quando submeto a experiência, o piloto automático do SageMaker passa por uma fase de pré-processamento através do Model Tuning, como mostra a Figura 5-34.

Agora que o pipeline do AutoML está em execução, é possível ver os recursos que ele usa na guia Recursos, como mostrado na Figura 5-35.

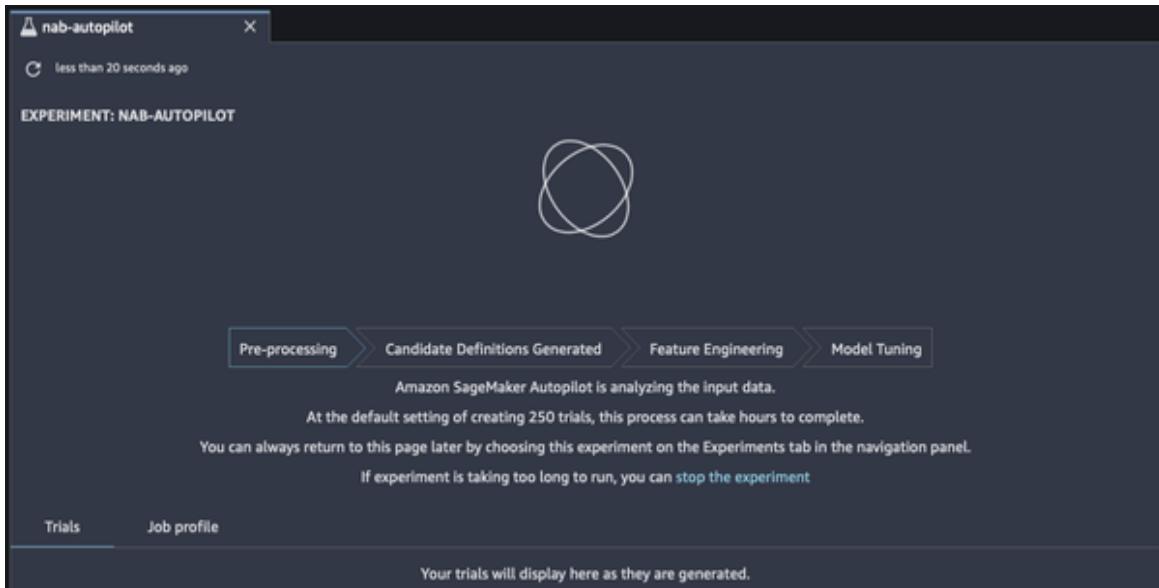


Figura 5-34. Executa a experiência do piloto automático

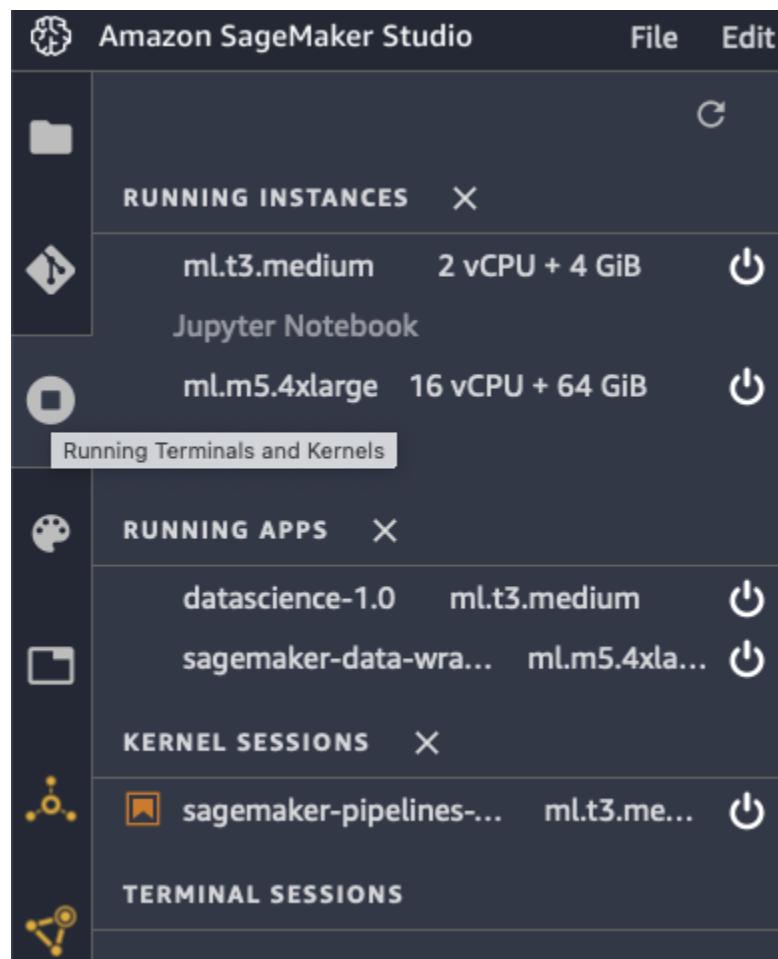


Figura 5-35. Instâncias do piloto automático

Quando o treinamento estiver concluído, poderá ver uma lista de modelos e sua precisão, como mostrado na [Figura 5-36](#). Observe que o SageMaker foi capaz de criar um modelo de classificação altamente preciso com uma precisão de 0,999945.

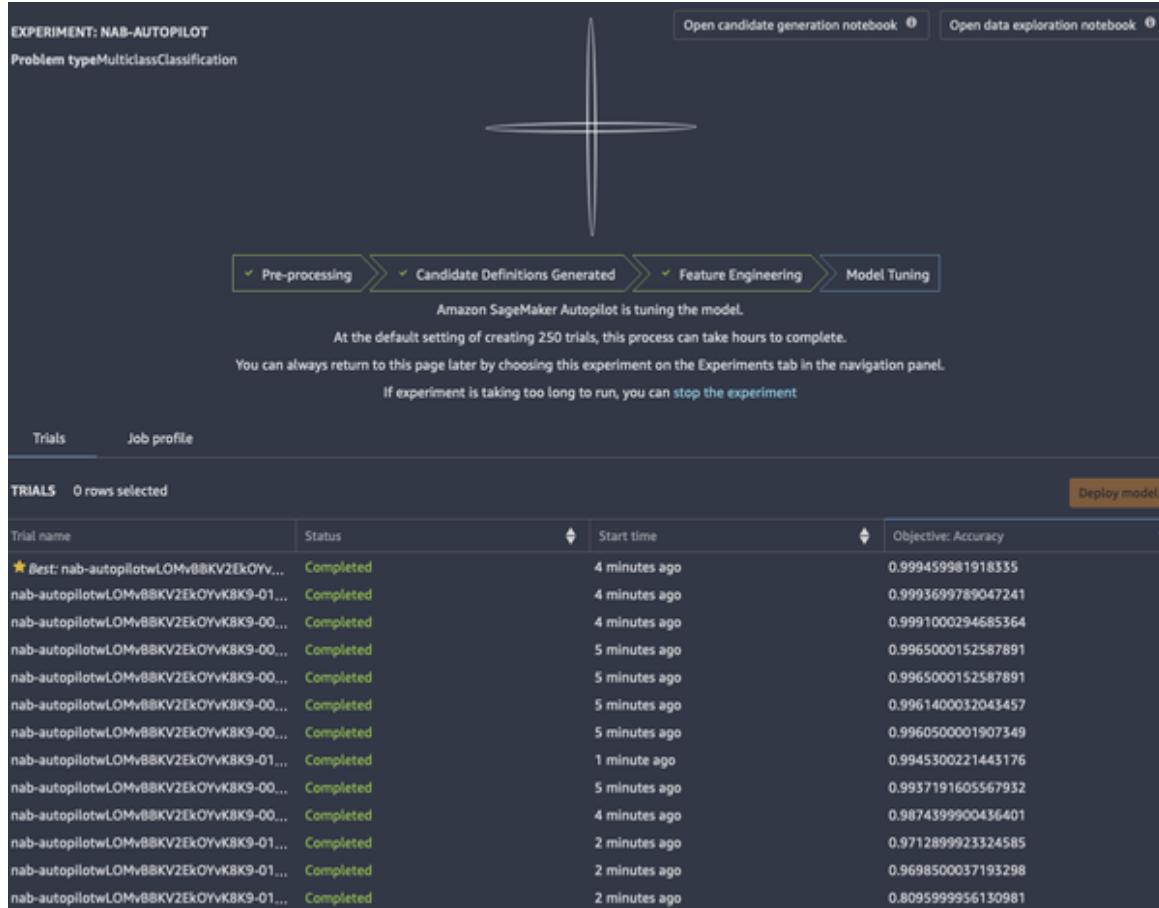


Figura 5-36. Execução concluída do piloto automático

Finalmente, como mostra a [Figura 5-37](#), uma vez concluída a tarefa, podes clicar com o botão direito do rato no modelo que pretendes controlar e implementá-lo na produção ou abri-lo no modo de detalhes do percurso para inspecionar a explicabilidade e/ou as métricas ou os gráficos.

O SageMaker Autopilot é uma solução completa para AutoML e MLOps, e se a tua organização já utiliza AWS, parece simples integrar esta plataforma nos teus fluxos de trabalho existentes. Parece especialmente útil quando se trabalha com conjuntos de dados maiores e com problemas em que a reproduzibilidade é essencial.

Trials		Job profile		
TRIALS 1 row selected				
Trial name	Status	Start time	Objective	
★ Best: nab-autopilotwLOMvBBKV2EkOY... Completed	Completed	11 minutes ago	0.999459981918335	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Comp	Open in trial component list	s ago	0.9993599789047241	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Comp	Open in trial details	s ago	0.9991000294685364	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Comp	Deploy model	s ago	0.9995000152587891	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Comp	Copy cell contents	s ago	0.9995000152587891	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Comp	Shift+Right Click for Browser Menu	s ago	0.99961400032043457	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	11 minutes ago	0.99960500001907349	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	8 minutes ago	0.9945300221443176	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	12 minutes ago	0.9937191605567932	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	11 minutes ago	0.9874399900436401	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	7 minutes ago	0.9853699803352356	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	7 minutes ago	0.9849299788475037	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	8 minutes ago	0.9757699966430664	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	9 minutes ago	0.9712899923324585	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	9 minutes ago	0.9698500037193298	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	9 minutes ago	0.8095999956130981	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	9 minutes ago	0.7774800062179565	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	7 minutes ago	0.7645599842071533	
nab-autopilotwLOMvBBKV2EkOYvK&K9-... Completed	Completed	8 minutes ago	0.1578475385904312	

Figura 5-37. Modelo de piloto automático

Em seguida, vamos discutir algumas das soluções AutoML de código aberto que estão a surgir.

Soluções AutoML de código aberto

Ainda me lembro com carinho dos meus dias de trabalho em clusters Unix enquanto trabalhava no Caltech em 2000. No entanto, esta foi uma altura de transição para o Unix, porque embora em muitos casos o Solaris fosse superior ao Linux, não podia competir com o preço do sistema operativo Linux, que era gratuito.

Vejo uma situação semelhante a acontecer com as soluções AutoML de código aberto. A capacidade de treinar e executar modelos utilizando ferramentas de alto nível parece estar a caminhar para a mercantilização. Por isso, vamos dar uma vista de olhos a algumas das opções de código aberto.

Ludwig

Uma das abordagens mais promissoras para o AutoML de código aberto é o [Ludwig AutoML](#). Na [Figura 5-38](#), a saída de uma execução do Ludwig mostra as métricas úteis para avaliar a força do modelo. A vantagem do código aberto é que uma corporação não o controla! Aqui está um exemplo de projeto que mostra a classificação de texto usando [Ludwig](#) através do [notebook Colab](#).

Primeiro, instala o Ludwig e configura uma transferência:

```
!pip install -q ludwig
!wget https://raw.githubusercontent.com/paiml/practical-mlops-
book/main/chap05/\
      config.yaml
!wget https://raw.githubusercontent.com/paiml/practical-mlops-
book/main/chap05/\
      reuters-allcats.csv
```

Em seguida, o modelo é apenas uma invocação de linha de comando. Esta etapa treina o modelo:

```
!ludwig experiment \
--dataset reuters-allcats.csv \
--config_file config.yaml
```

class	loss	accuracy	hits_at_k
train	0.9258	0.7148	0.9826
vali	0.9134	0.6992	0.9692
test	0.9420	0.7311	0.9781

Figura 5-38. Ludwig

Podes encontrar muitos outros exemplos excelentes do Ludwig na sua [documentação oficial](#).

Um dos aspectos mais interessantes do Ludwig é que ele está em desenvolvimento ativo. Como parte da Linux Foundation, eles lançaram recentemente a versão 4, que pode ser vista na [Figura 5-39](#). Acrescenta muitos recursos adicionais, como trabalhar com sistemas de arquivos

remotos e ferramentas distribuídas fora da memória, como Dask e Ray. Finalmente, Ludwig tem uma profunda integração com MLflow. O roteiro do Ludwig mostra que ele continuará a apoiar e melhorar essa integração.

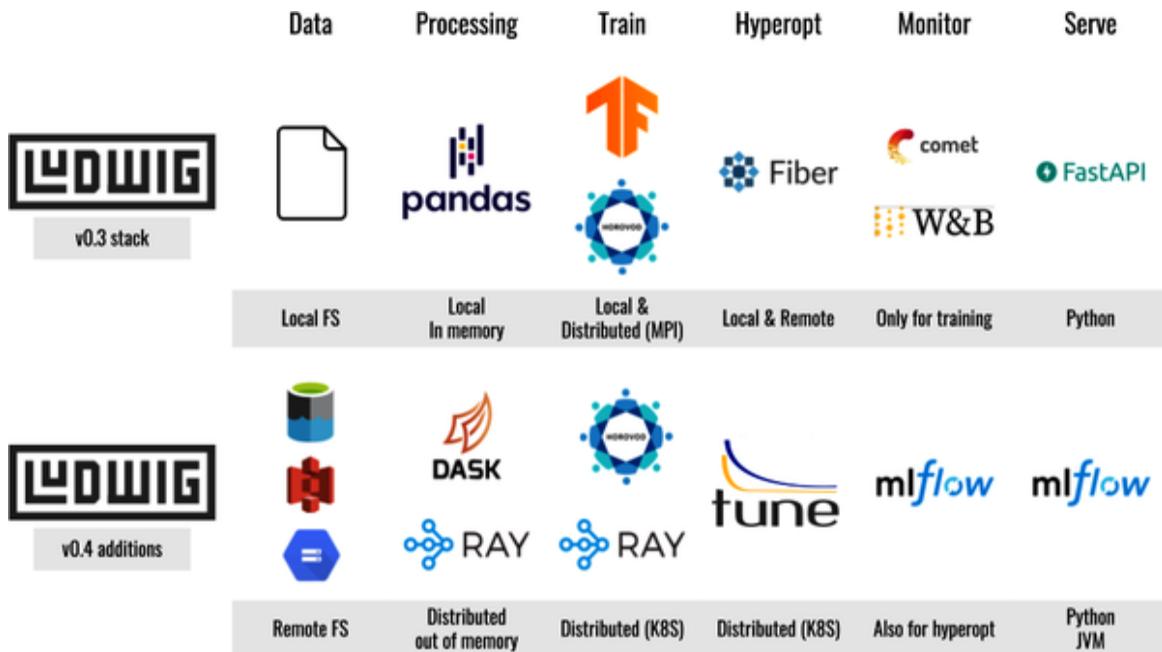


Figura 5-39. Ludwig versão 4

FLAML

Outro participante no AutoML de código aberto é o **FLAML**. Tem um desenho que permite uma otimização rentável dos hiperparâmetros. Podes ver o logótipo FLAML na Figura 5-40.

FLAML - Fast and Lightweight AutoML



Figura 5-40. FLAML da Microsoft Research

Um dos principais casos de utilização de FLAML é a automatização de todo um processo de modelação com apenas três linhas de código. Podes ver isto no exemplo seguinte:

```
from flaml import AutoML
automl = AutoML()
automl.fit(X_train, y_train, task="classification")
```

Um exemplo mais abrangente mostra que, num notebook Jupyter, instala primeiro a biblioteca !pip install -q flaml e, em seguida, define a configuração do AutoML. Em seguida, inicia um trabalho de treinamento para selecionar o modelo de classificação otimizado:

```
!pip install -q flaml

from flaml import AutoML
from sklearn.datasets import load_iris
# Initialize an AutoML instance
automl = AutoML()
# Specify automl goal and constraint
automl_settings = {
    "time_budget": 10, # in seconds
    "metric": 'accuracy',
    "task": 'classification',
}
X_train, y_train = load_iris(return_X_y=True)
```

```

# Train with labeled input data
automl.fit(X_train=X_train, y_train=y_train,
            **automl_settings)
# Predict
print(automl.predict_proba(X_train))
# Export the best model
print(automl.model)

```

Pode ver na [Figura 5-41](#) que, após várias iterações, seleciona um XGBClassifier com um conjunto de hiperparâmetros optimizados.

```

[flaml.automl: 06-16 17:20:13] {1013} INFO - iteration 46, current learner xgboost
[flaml.automl: 06-16 17:20:13] {1165} INFO - at 10.0s, best xgboost's error=0.0333,      best xgboo
st's error=0.0333
[flaml.automl: 06-16 17:20:13] {1013} INFO - iteration 47, current learner catboost
[flaml.automl: 06-16 17:20:13] {1165} INFO - at 10.1s, best catboost's error=0.0333,      best xgboo
st's error=0.0333
[flaml.automl: 06-16 17:20:13] {1205} INFO - selected model: XGBClassifier(colsample_bytree=0.690
2766231016318,
                           colsample_bytree=0.7657293008018354, grow_policy='lossguide',
                           learning_rate=0.42830712534058824, max_depth=0, max_leaves=5,
                           min_child_weight=0.2924296818378054, n_estimators=6, n_jobs=-1,
                           objective='multi:softprob', reg_alpha=0.00285817466554831,
                           reg_lambda=2.32876649803287, subsample=1.0, tree_method='hist',
                           use_label_encoder=False, verbosity=0)
[flaml.automl: 06-16 17:20:13] {963} INFO - fit succeeded
[[0.9206522  0.04071239  0.03863542]
 [0.91942585  0.04199015  0.03858395]]

```

Figura 5-41. Saída FLAML da seleção de modelos

O que é empolgante nesses frameworks de código aberto é sua capacidade de tornar possíveis coisas complicadas e automatizadas coisas fáceis . A seguir, vamos ver como funciona a explicabilidade do modelo com um passo-a-passo do projeto.

NOTA

Não há escassez de frameworks AutoML de código aberto. Aqui estão alguns frameworks adicionais para olhares para o AutoML:

- AutoML
 - [H2O AutoML](#)
 - [Aprendizagem automática](#)
 - [tpot](#)
 - [PyCaret](#)
 - [AutoKeras](#)

Explicabilidade do modelo

Um aspeto importante da automatização na aprendizagem automática é a automatização da explicabilidade do modelo. Todas as plataformas MLOps podem utilizar esta capacidade como mais um painel de controlo para a equipa analisar durante o trabalho. Por exemplo, uma equipa de MLOps que começa a trabalhar de manhã pode ver a utilização da CPU e da memória dos servidores e o relatório de explicabilidade do modelo que treinou ontem à noite.

As estruturas de MLOps baseadas na Cloud, como o AWS SageMaker, o Azure ML Studio e o Google Vertex AI, têm uma capacidade de explicação de modelos incorporada, mas também podes implementá-la com software de código aberto. Vamos percorrer um fluxo de trabalho de explicabilidade para ver como isso funciona usando este [projeto GitHub de explicabilidade de modelo](#).

NOTA

Duas estruturas populares de código aberto de explicabilidade de modelos são ELI5 e SHAP. Aqui tens um pouco mais de informação sobre cada um deles.

ELI5

ELI5 significa "explica como se eu tivesse cinco anos". Permite-te visualizar e depurar modelos de aprendizagem automática e suporta várias estruturas, incluindo sklearn.

SHAP

O **SHAP** é uma abordagem "teórica dos jogos" para explicar os resultados dos modelos de aprendizagem automática. Em particular, tem excelentes visualizações, bem como explicações.

Primeiro, usando um **notebook Jupyter**, vamos ingerir dados da NBA da temporada 2016-2017 e imprimir as primeiras linhas usando o comando `head`. Esses dados contêm IDADE, POSIÇÃO, FG (objetivos de campo/jogo) e dados de mídia social, como retweets do Twitter:

```
import pandas as pd

player_data =
"https://raw.githubusercontent.com/noahgift/socialpowernba/\nmaster/data/nba_2017_players_with_salary_wiki_twitter.csv"
df = pd.read_csv(player_data)
df.head()
```

Em seguida, vamos criar uma nova característica chamada `winning_season`, que nos permite prever se um jogador fará parte de uma equipa com uma época vitoriosa. Por exemplo, na **Figura 5-42**, podes ver o gráfico da idade do jogador da NBA versus as vitórias para descobrir um potencial padrão baseado na idade.

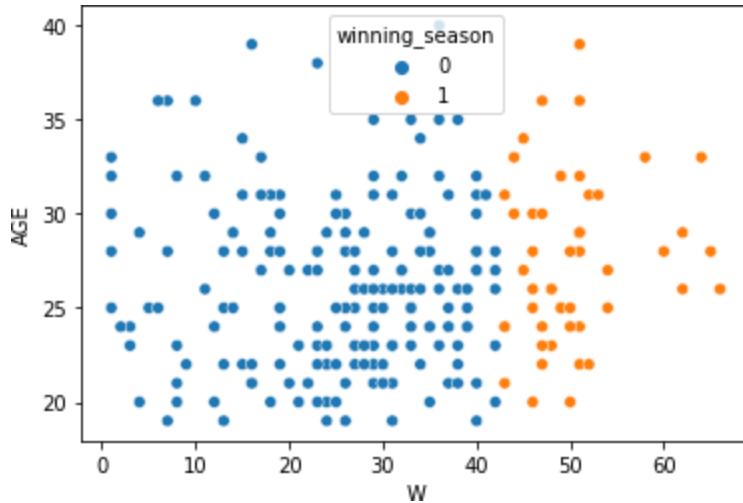


Figura 5-42. Característica da época de vitórias

Agora, vamos passar à modelação e prever vitórias. Mas primeiro, vamos limpar um pouco os dados e eliminar colunas que não são necessárias e eliminar valores em falta:

```
df2 = df[["AGE", "POINTS", "SALARY_MILLIONS", "PAGEVIEWS",
          "TWITTER_FAVORITE_COUNT", "winning_season", "TOV"]]
df = df2.dropna()
target = df["winning_season"]
features = df[["AGE", "POINTS", "SALARY_MILLIONS", "PAGEVIEWS",
               "TWITTER_FAVORITE_COUNT", "TOV"]]
classes = ["winning", "losing"]
```

Após esta limpeza, o comando `shape` imprime o número de linhas, 239, e colunas, 7:

```
df2.shape
(239, 7)
```

A seguir, vamos treinar o modelo, primeiro dividindo os dados e depois utilizando a regressão logística:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,
                                                    target,
                                                    test_size=0.25,
                                                    random_state=0)
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(solver='lbfgs', max_iter=1000)
model.fit(x_train, y_train)
```

Deves ver um resultado semelhante ao seguinte, que mostra que o treino do modelo foi bem sucedido:

```
LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
           intercept_scaling=1, l1_ratio=None,
max_iter=1000,
           multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001,
verbose=0,
           warm_start=False)
```

Agora, vamos passar à parte divertida que explica como é que o modelo chegou às previsões da estrutura SHAP. Mas, primeiro, o SHAP precisa de ser instalado:

```
!pip install -q shap
```

Em seguida, vamos utilizar `xgboost`, outro algoritmo de classificação, para explicar o modelo, uma vez que o SHAP tem um excelente suporte:

```
import xgboost
import shap
model_xgboost = xgboost.train({"learning_rate": 0.01},
                               xgboost.DMatrix(x_train,
label=y_train), 100)
# load JS visualization code to notebook
shap.initjs()
# explain the model's predictions using SHAP values
# (same syntax works for LightGBM, CatBoost, and scikit-learn
models)
explainer = shap.TreeExplainer(model_xgboost)
shap_values = explainer.shap_values(features)
# visualize the first prediction's explanation
shap.force_plot(explainer.expected_value, shap_values[0,:],
features.iloc[0,:])
```

Na [Figura 5-43](#), podes ver um gráfico de força no SHAP que mostra que as características a vermelho aumentam a previsão e as características a azul diminuem a previsão.

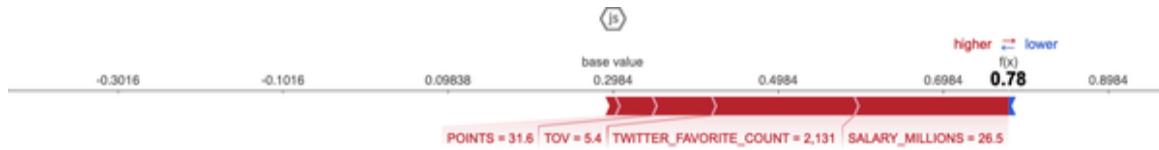


Figura 5-43. Saída SHAP xgboost

```
shap.summary_plot(shap_values, features, plot_type="bar")
```

Na [Figura 5-44](#), um gráfico de resumo mostra o valor médio absoluto das características que impulsionam o modelo. Assim, por exemplo, podes ver como as métricas "fora do campo", como o Twitter e o Salário, são factores essenciais para que o modelo preveja as vitórias da forma como o faz.

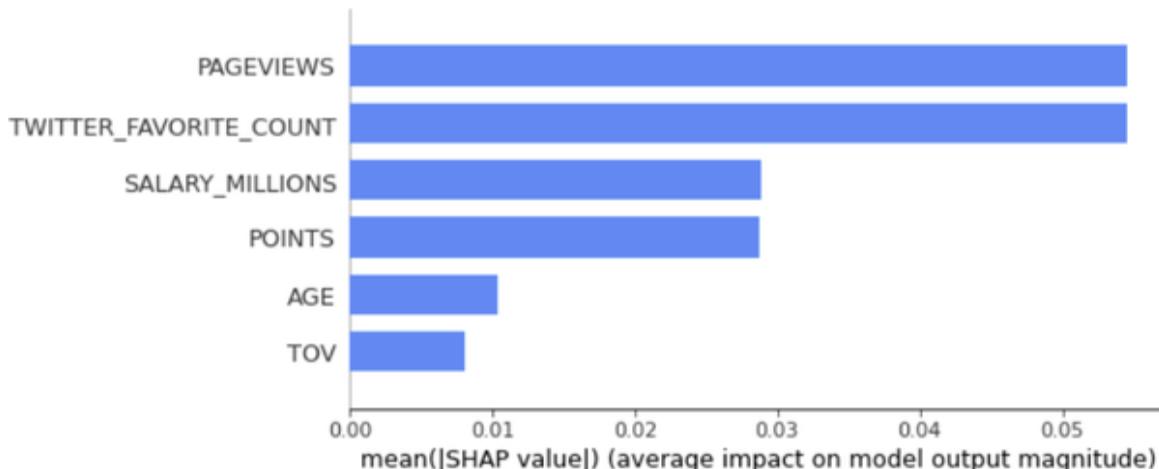


Figura 5-44. Importância da caraterística SHAP

Vamos ver como funciona outra ferramenta de código aberto; desta vez, vamos usar o ELI5. Primeiro, instala-o com pip:

```
!pip install -q eli5
```

Em seguida, a importância da permutação é executada no modelo de regressão logística original criado anteriormente. Este processo funciona medindo a quantidade de diminuição da precisão ao remover caraterísticas:

```

import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(model, random_state=1).fit(x_train,
y_train)
eli5.show_weights(perm, feature_names =
features.columns.tolist())

```

Podes ver na [Figura 5-45](#) que o modelo de regressão logística original tem uma importância de característica diferente do modelo XGBoost. Em particular, repara que a idade de um jogador tem uma correlação negativa com as vitórias.

Weight	Feature
0.0090 ± 0.0055	TOV
0.0079 ± 0.0136	POINTS
0.0056 ± 0.0000	PAGEVIEWS
0.0056 ± 0.0071	SALARY_MILLIONS
0.0034 ± 0.0055	TWITTER_FAVORITE_COUNT
-0.0079 ± 0.0055	AGE

Figura 5-45. Importância da permutação ELI5

A explicabilidade é um aspeto essencial dos MLOps. Tal como temos painéis de controlo e métricas para os sistemas de software, também deve ser explicável a forma como um sistema de IA/ML chegou à sua previsão. Esta explicabilidade pode conduzir a resultados mais saudáveis tanto para as partes interessadas da empresa como para a própria empresa.

De seguida, vamos terminar tudo o que foi abordado no capítulo.

Conclusão

O AutoML é uma nova capacidade essencial para qualquer equipa que faça MLOps. O AutoML melhora a capacidade de uma equipa para colocar modelos em produção, trabalhar em problemas complexos e, em última análise, trabalhar no que interessa. É essencial salientar que a Modelação Automatizada, ou seja, o AutoML, não é o único componente do KaizenML ou da melhoria contínua. No artigo frequentemente citado "["Hidden](#)

Technical Debt in Machine Learning Systems" (Dívida técnica oculta em sistemas de aprendizagem automática), os autores mencionam que a modelação é uma parte trivial do trabalho num sistema de aprendizagem automática do mundo real. Da mesma forma, não é de surpreender que o AutoML, ou seja, a automatização da modelação, seja uma pequena parte do que precisa de ser automatizado. Tudo, desde a ingestão de dados ao armazenamento de características, à modelação, à formação, à implementação e à avaliação do modelo em produção, é candidato a uma automatização total. KaizenML significa que estás constantemente a melhorar cada parte do sistema de aprendizagem automática.

Tal como os sistemas automatizados de transmissão e de controlo da velocidade de cruzeiro ajudam um condutor experiente, a automatização dos subcomponentes de um sistema de aprendizagem automática de produção melhora as decisões dos seres humanos responsáveis pelo ML. As coisas podem e devem ser automatizadas, incluindo aspectos de modelação, melhores práticas de engenharia de software, testes, engenharia de dados e outros componentes essenciais. A melhoria contínua é uma mudança cultural que não tem data de término e se encaixa em qualquer organização que queira fazer mudanças impactantes com IA e aprendizado de máquina.

Uma última conclusão é que existem muitas soluções AutoML gratuitas ou quase gratuitas. Tal como os programadores de todo o mundo utilizam ferramentas de alto nível gratuitas ou quase gratuitas, como servidores de construção e editores de código, para melhorar o software, os profissionais de ML devem utilizar ferramentas de automatização de todos os tipos para aumentar a sua produtividade.

Segue-se o capítulo sobre monitorização e registo de dados. Eu chamo isso de "ciência de dados para operações". Antes de entrares nesse tópico, dá uma vista de olhos aos seguintes exercícios e perguntas de reflexão crítica.

Exercícios

- Transfere o XCode e utiliza o Create ML da Apple para treinar um modelo a partir de um conjunto de dados de amostra que encontres no Kaggle ou noutra localização de conjuntos de dados abertos.
- Utiliza a plataforma AutoML Computer Vision da Google para treinar um modelo e implementá-lo num dispositivo Coral.AI.
- Utiliza o Azure ML Studio para treinar um modelo e explorar as funcionalidades de explicabilidade do Azure ML Studio.
- Utiliza o ELI5 para explicar um modelo de aprendizagem automática.
- Utiliza o Ludwig para treinar um modelo de aprendizagem automática.
- Selecciona um exemplo de ajuste automático de modelo do SageMaker nos exemplos oficiais do SageMaker e executa-o na sua conta AWS.

Questões para discussão sobre pensamento crítico

- Porque é que o AutoML é apenas uma parte da história da automação com aprendizagem automática moderna?
- Como é que o NIH (National Institutes of Health) poderia utilizar uma Loja de Funcionalidades para aumentar a velocidade das descobertas médicas?
- Até 2025, que partes da aprendizagem automática serão totalmente automatizadas e que aspectos não o serão? Até 2035, que partes da aprendizagem automática serão totalmente automatizadas e que factores não o serão?
- Como é que as plataformas de IA integradas verticalmente (chips, estruturas, dados, etc.) podem dar a determinadas empresas uma

vantage competitiva?

- De que forma é que a indústria do software de xadrez fornece informações sobre a forma como a IA e os seres humanos trabalham em conjunto para obter melhores resultados na resolução de problemas para a AutoML?
- Como é que uma abordagem centrada nos dados difere de uma abordagem centrada no modelo para a aprendizagem automática? O que achas de uma abordagem KaizenML em que os dados, o software e a modelação são todos tratados como igualmente importantes?

Capítulo 6. Monitorização e registo

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

Não só a anatomia cerebral é dupla, e não só é indiscutível que um hemisfério é suficiente para a consciência; para além disso, foi demonstrado que dois hemisférios após calosotomia são conscientes simultânea e independentemente. Como Nagel disse sobre o cérebro dividido, "O que o hemisfério direito pode fazer por si só é demasiado elaborado, demasiado intencionalmente dirigido e demasiado psicologicamente inteligível para ser considerado meramente como um conjunto de respostas automáticas inconscientes.

—Dr. Joseph Bogen

Tanto o registo como a monitorização são pilares fundamentais dos princípios DevOps que são cruciais para práticas de ML robustas. O registo e a monitorização úteis são difíceis de acertar e, embora possas aproveitar os serviços Cloud que tratam do trabalho pesado, cabe-te a ti decidir e apresentar uma estratégia sólida que faça sentido. A maioria dos engenheiros de software tende a preferir escrever código e a deixar para trás outras tarefas como testes, documentação e, muitas vezes, registo e monitorização.

Não te surpreendas se ouvires sugestões sobre soluções automatizadas que podem "resolver o problema do registo". Uma base sólida é possível se pensares bem no problema em questão para que a informação produzida seja utilizável. O trabalho árduo e os ideais de base sólida que descrevo tornam-se muito claros quando te deparas com informação inútil (não ajuda

a narrar uma história) ou críptica (demasiado difícil de compreender). Um exemplo perfeito desta situação é um problema de software que abri em 2014 e que captou a seguinte pergunta de um chat online sobre o produto:

"Alguém me pode ajudar a interpretar esta linha:

```
7fede0763700 0 -- :/1040921 >> 172.16.17.55:6789/0
pipe(0x7feddc022470 \
sd=3 :0 s=1 pgs=0 cs=0 l=1 c=0x7feddc0226e0).fault
```

Na altura, já trabalhava com este produto de software há quase dois anos e não fazia ideia do que isso significava. Consegues pensar numa resposta possível? Um engenheiro experiente tinha a tradução perfeita: "A máquina em que estás não consegue contactar o monitor em 172.16.17.55." Fiquei perplexo com o significado da declaração do registo. Porque é que não podemos fazer uma alteração para dizer isso em vez disso? O ticket que captura esse problema de 2014, no momento em que escrevo este artigo, ainda está aberto. Ainda mais preocupante é o facto de a engenharia ter respondido nesse bilhete que "A mensagem de registo está bem".

O registo e a monitorização são um trabalho árduo porque é necessário esforço para produzir resultados significativos que nos ajudem a compreender o estado de um programa.

Já mencionei que é crucial ter informações que nos ajudem a narrar uma história. Isto é verdade tanto para a monitorização como para o registo. Há alguns anos atrás, trabalhei num grande grupo de engenharia que entregou um dos maiores CMSs (Content Management System) do mundo baseado em Python. Depois de propor a adição de métricas à aplicação, o sentimento geral era de que o CMS não precisava delas. A monitorização já estava implementada e a equipa de operações tinha todo o tipo de utilitários ligados a limites para alertas. Os diretores de engenharia recompensavam a excelência dando a um engenheiro tempo para trabalhar em qualquer projeto relevante (não apenas 20% como uma empresa de tecnologia famosa). Antes de trabalhar num projeto relevante, era necessário apresentar a ideia a toda a equipa de gestão para obter a sua aprovação.

Quando chegou a minha vez, optei, naturalmente, por acrescentar facilidades métricas à aplicação.

"Alfredo, já temos métricas, sabemos a utilização do disco e temos alertas de memória para cada servidor. Não percebemos o que ganhamos com esta iniciativa." É difícil estar à frente de um grande grupo de gestores seniores e tentar convencê-los de algo em que não acreditam. A minha explicação começou com o botão mais importante do sítio Web: o botão de subscrição. Esse botão de subscrição era o responsável pela produção de utilizadores pagantes e era crucial para o negócio. Expliquei-te que, "se lançarmos uma nova versão com um problema de JavaScript que torne este botão inutilizável, que métrica ou alerta nos pode dizer que é um problema?" É claro que a utilização do disco seria a mesma e a utilização da memória provavelmente não se alteraria em nada. E, no entanto, o botão mais importante da aplicação passaria despercebido num estado inutilizável. Neste caso particular, as métricas podem capturar a taxa de cliques de cada hora, dia e semana para esse botão. E, mais importante, pode ajudar a contar uma história sobre como o site gera hoje mais (ou talvez menos!) receitas do que no ano passado, neste mesmo mês. Vale a pena controlar a utilização do disco e o consumo de memória dos servidores, mas não é o objetivo final.

Estas histórias não estão explicitamente ligadas à aprendizagem automática ou ao fornecimento de modelos treinados para produção. No entanto, como verás neste capítulo, ajudar-te-ão a ti e à tua empresa a contar uma história e a aumentar a confiança no teu processo, revelando questões importantes e apontando a razão pela qual um modelo necessita provavelmente de melhores dados antes de entrar em produção. Identificar o desvio e a precisão dos dados ao longo do tempo é fundamental nas operações de ML. A implementação de um modelo na produção com uma alteração significativa na precisão nunca deve acontecer e precisa de ser prevenida. Quanto mais cedo estes problemas forem detectados, mais barato será corrigi-los. As consequências de ter um modelo impreciso em produção podem ser catastróficas.

Observabilidade para MLOps em Cloud

É seguro dizer que a maior parte da aprendizagem automática está a decorrer num ambiente Cloud. Como tal, existem serviços especiais de fornecedores de Cloud que permitem a observabilidade. Por exemplo, no AWS, tens o [Amazon CloudWatch](#), no GCP tens o [conjunto de operações do Google Cloud](#) e no Azure tens o [Azure Monitor](#).

Na [Figura 6-1](#), o Amazon CloudWatch é um exemplo melhor de como esses serviços de monitoramento funcionam. Em um nível alto, cada componente do sistema de nuvem envia métricas e logs para o CloudWatch. Essas tarefas incluem os servidores, os logs de aplicativos, os metadados de treinamento para trabalhos de aprendizado de máquina e os resultados dos pontos de extremidade de aprendizado de máquina de produção.

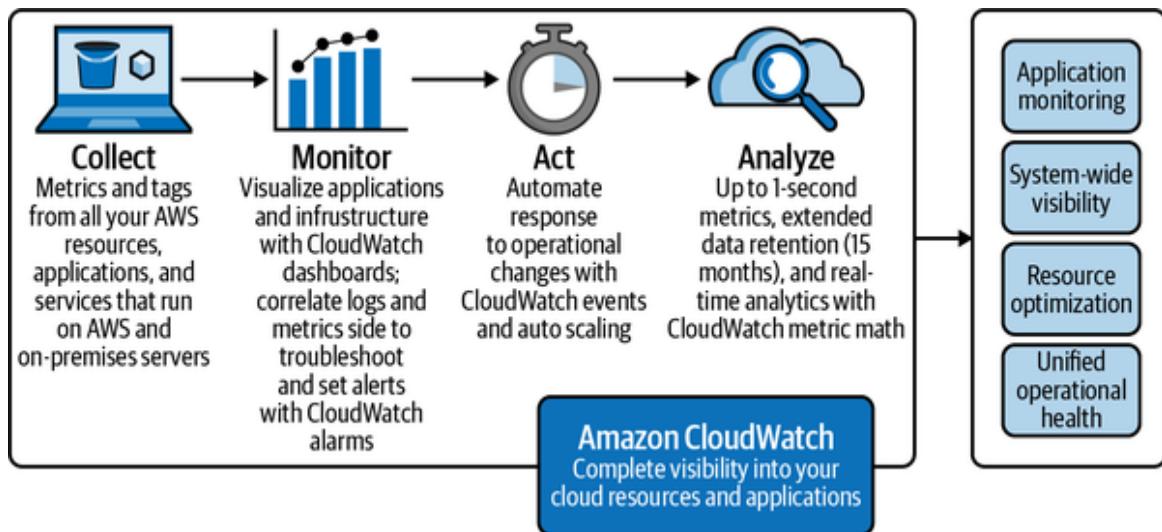
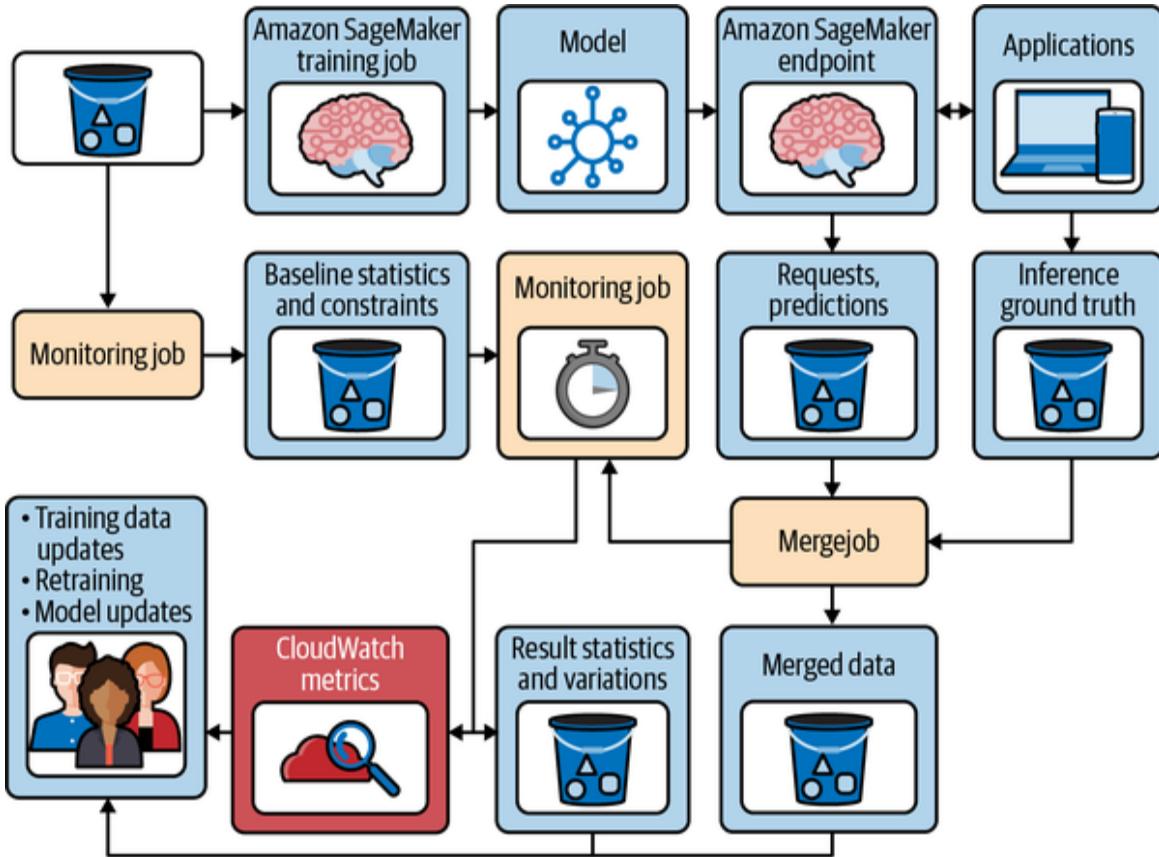


Figura 6-1. AWS CloudWatch

Em seguida, essas informações passam a fazer parte de muitas ferramentas diferentes, de painéis de controle a dimensionamento automático. Por exemplo, no AWS SageMaker, isso pode significar que o serviço de ML de produção será dimensionado automaticamente se os pontos de extremidade individuais excederem mais de 75% de sua CPU ou memória total. Por fim, toda essa observabilidade permite que humanos e máquinas analisem o que está acontecendo com um sistema de ML de produção e ajamde acordo com isso.

O engenheiro de software em nuvem experiente já sabe que as ferramentas de observabilidade em nuvem são componentes não opcionais de uma implantação de software em nuvem. No entanto, o que é único sobre MLOps na Cloud é que os novos componentes também precisam de monitoramento granular. Por exemplo, na [Figura 6-2](#), uma série totalmente nova de ações ocorre em uma implantação de modelo de ML. Observa, porém, que novamente o CloudWatch coleta essas novas métricas.



À medida que o restante do capítulo se desenrola, lembre-se de que, em um sistema mestre de alto nível como o CloudWatch, ele coleta dados, encaminha alertas e se envolve com os componentes dinâmicos da computação em nuvem, como o dimensionamento elástico. A seguir, vamos entrar nos detalhes de um membro mais refinado da observabilidade: o logging.

Introdução ao registo

A maioria dos sistemas de registo tem aspectos comuns no seu funcionamento. Os sistemas definem os níveis de registo com que podem operar e, em seguida, o utilizador pode selecionar em que nível essas declarações devem aparecer. Por exemplo, o servidor web Nginx vem com uma configuração padrão para logs de acesso salvos em `/var/log/nginx/access.log` e logs de erro em `/var/log/nginx/error.log`. Como utilizador do Nginx, a primeira coisa que tens de fazer se estiveres a resolver problemas do servidor Web é aceder a essas falhas e ver o resultado.

Instalei o Nginx num servidor Ubuntu com as configurações por defeito e enviei alguns pedidos HTTP. Imediatamente, os logs de acesso começaram a receber algumas informações:

```
172.17.0.1 [22/Jan/2021:14:53:35 +0000] "GET / HTTP/1.1" \
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0) \
Firefox/84.0" "-"
172.17.0.1 [22/Jan/2021:14:53:38 +0000] "GET / HTTP/1.1" \
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0) \
Firefox/84.0" "-"
```

As longas linhas de registo contêm muita informação útil (configurável) que inclui o endereço IP do servidor, a hora e o tipo de pedido juntamente com a informação do agente do utilizador. O agente do utilizador, no meu caso, é o meu browser a correr num computador Macintosh. No entanto, não se trata de erros. Essas linhas mostram o acesso ao servidor. Para forçar o Nginx a entrar em uma condição de erro, alterei as permissões em um arquivo para que ele não pudesse ser lido pelo servidor. Em seguida, enviei novas solicitações HTTP:

```
2021/01/22 14:59:12 [error] open()
"/usr/share/nginx/html/index.html" failed \
(13: Permission denied), client: 172.17.0.1, server: localhost,
\
request: "GET / HTTP/1.1", host: "192.168.0.200"
```

A entrada de registo mostra explicitamente que o nível da informação é de "erro". Isto permite a um consumidor, como eu, identificar a gravidade da informação produzida pelo servidor Web. Quando eu era um administrador de sistemas e estava a começar a realizar tarefas necessárias, como a configuração e a implementação de ambientes de produção, não era claro porque é que estes níveis eram úteis. O ponto principal para mim era que eu podia identificar um erro a partir do conteúdo informativo dos logs.

Embora a ideia de tornar mais fácil para os consumidores identificarem se um erro é válido, não é tudo o que existe nos níveis de registo. Se já tentaste depurar um programa antes, provavelmente usaste as instruções `print()` para te ajudar a obter informações úteis sobre um programa em execução. Há muitas outras maneiras de depurar programas, mas usar `print()` ainda é valioso. Uma desvantagem é que tens de limpar e remover todas essas instruções `print()` assim que o problema estiver resolvido. Da próxima vez que precisares de depurar o mesmo programa, terás de voltar a adicionar todas essas instruções. Esta não é uma boa estratégia, e é uma das muitas situações em que o registo pode ajudar.

Agora que algumas noções básicas sobre logging estão claras, vou discutir como configurar o logging numa aplicação, o que tende a ser complicado, uma vez que existem muitas opções e decisões diferentes a tomar.

Regista em Python

Vou trabalhar em Python, mas a maioria dos conceitos nesta secção devem aplicar-se a outras linguagens e frameworks. Os níveis de registo, o redireccionamento da saída e outras facilidades estão normalmente disponíveis noutras aplicações. Para começar a aplicar o registo, vou criar um pequeno script Python para processar um ficheiro CSV. Não há funções ou módulos; o script de exemplo é o mais próximo de uma célula do Jupyter Notebook que podes encontrar.

Cria um novo ficheiro chamado `describe.py` para veres como o registo pode ajudar num script básico:

```
import sys
import pandas as pd

argument = sys.argv[-1]

df = pd.read_csv(argument)
print(df.describe())
```

O script recebe o input do último argumento na linha de comando e diz à biblioteca Pandas para o ler e descrever. A ideia é produzir uma descrição de um ficheiro CSV, mas não é isso que acontece quando o executas sem argumentos:

```
$ python derscribe.py
      from os import path
count              5
unique             5
top    print(df.describe())
freq               1
```

O que acontece aqui é que o último argumento nesse exemplo é o próprio script, por isso o Pandas está a descrever o conteúdo do script. Isto não é muito útil e, para alguém que não tenha criado o script, os resultados são, no mínimo, chocantes. Vamos levar este script frágil um passo à frente e passar um argumento para um caminho que não existe:

```
$ python describe.py /bogus/path.csv
Traceback (most recent call last):
  File "describe.py", line 7, in <module>
    df = pd.read_csv(argument)
  File "../../site-packages/pandas/io/parsers.py", line 605, in
read_csv
    return _read(filepath_or_buffer, kwds)
...
  File "../../site-packages/pandas/io/common.py", line 639, in
get_handle
    handle = open(
FileNotFoundException: [Errno 2] No such file or directory:
'/bogus/path.csv'
```

Não há verificação de erros para nos dizer se a entrada é válida e o que o script espera. Este problema é pior se estiveres à espera que um pipeline seja executado ou que algum trabalho de processamento de dados remoto seja concluído e estiveres a receber este tipo de erros. Alguns desenvolvedores tentam se defender contra eles capturando todas as exceções e ocultando o erro real, tornando impossível saber o que está acontecendo. Esta versão ligeiramente modificada do script destaca melhor o problema:

```
import sys
import pandas as pd

argument = sys.argv[-1]

try:
    df = pd.read_csv(argument)
    print(df.describe())
except Exception:
    print("Had a problem trying to read the CSV file")
```

Executá-lo produz um erro que me deixaria muito aborrecido se o visse no código de produção:

```
$ python describe.py /bogus/path.csv
Had a problem trying to read the CSV file
```

O exemplo é trivial e, como o script tem apenas algumas linhas e tu conheces o seu conteúdo, não é assim tão difícil apontar o problema. Mas se isto estiver a ser executado num pipeline automatizado remotamente, não tens qualquer contexto e torna-se difícil compreender o problema. Vamos usar o módulo de registo Python para fornecer mais informações sobre o que está a acontecer neste script de processamento de dados.

A primeira coisa a fazer é configurar o registo. Não precisamos de nada muito complicado aqui, e adicionar algumas linhas é mais do que suficiente. Modifica o ficheiro *describe.py* para incluir estas linhas e depois volta a executar o script:

```

import logging

logging.basicConfig()
logger = logging.getLogger("describe")
logger.setLevel(logging.DEBUG)

argument = sys.argv[-1]
logger.debug("processing input file: %s", argument)

```

Se voltares a executá-lo, deve ser semelhante a isto:

```

$ python describe.py /bogus/path.csv
DEBUG:describe:processing input file: /bogus/path.csv
Had a problem trying to read the CSV file

```

Mesmo assim, ainda não é muito útil, mas já é informativo. Isso pode parecer um monte de código padrão para algo que uma simples declaração `print()` também poderia ter feito. O módulo de logging é resistente a falhas na construção da mensagem. Por exemplo, abre um interpretador Python e tenta usar menos argumentos para `print`:

```

>>> print("%s should break because: %s" % "statement")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: not enough arguments for format string

```

Agora vamos tentar a mesma operação com o módulo de registo:

```

>>> import logging
>>> logging.warning("%s should break because: %s", "statement")
--- Logging error ---
Traceback (most recent call last):
...
  File "/.../python3.8/logging/__init__.py", line 369, in
getMessage
    msg = msg % self.args
TypeError: not enough arguments for format string
Call stack:
  File "<stdin>", line 1, in <module>
Message: '%s should break because: %s'
Arguments: ('statement',)

```

O último exemplo não interromperia uma aplicação de produção. O registo nunca deve quebrar qualquer aplicação em tempo de execução. Neste caso, o módulo de registo está a tentar fazer a substituição da variável na cadeia de caracteres e a falhar, mas em vez de falhar está a anunciar o problema e depois a continuar. Uma instrução de impressão não pode fazer isso. Na verdade, eu vejo o uso de `print()` em Python muito parecido com `echo` em scripts de shell. Não há controlo, é fácil quebrar uma aplicação de produção e é difícil controlar a verbosidade.

A verbosidade é crucial no registo e, para além do *nível de erro* no exemplo do Nginx, tem várias facilidades para permitir que os consumidores de registos ajustem a informação necessária. Antes de entrar na granularidade do nível de registo e no controlo de verbosidade, dá ao script uma atualização na formatação do registo para ficar mais agradável. O módulo de logging do Python é atraente e permite muitas configurações. Actualiza o script `describe.py` onde a configuração do registo acontece:

```
log_format = "[%name)s][%(levelname)-6s] %(message)s"
logging.basicConfig(format=log_format)
logger = logging.getLogger("describe")
logger.setLevel(logging.DEBUG)
```

O `log_format` é um modelo com algumas palavras-chave utilizadas na construção da linha de registo. Repara como eu não tinha um timestamp antes, e embora ainda não o tenha com esta atualização, a configuração permite-me incluí-lo. Por enquanto, o nome do registrador (`describe` neste caso), o nível de registro e a mensagem estão todos lá com algum preenchimento e usando colchetes para separar as informações para melhor legibilidade. Volta a executar o script mais uma vez para verificar como o resultado muda:

```
$ python describe.py
[describe][DEBUG ] processing input file: describe.py
Had a problem trying to read the CSV file
```

Outro superpoder do registo é fornecer a informação de rastreio juntamente com o erro. Às vezes é útil capturar (e mostrar) o traceback sem entrar em

uma condição de erro. Para fazer isso, atualiza o script *describe.py* no bloco `except`:

```
try:  
    df = pd.read_csv(argument)  
    print(df.describe())  
except Exception:  
    logger.exception("Had a problem trying to read the CSV file")
```

É muito semelhante à declaração `print()` de antes. Volta a executar o script e verifica os resultados:

```
$ python logging_describe.py  
[describe][DEBUG ] processing input file: logging_describe.py  
[describe][ERROR ] Had a problem trying to read the CSV file  
Traceback (most recent call last):  
  File "logging_describe.py", line 15, in <module>  
    df = pd.read_csv(argument)  
  File ".../site-packages/pandas/io/parsers.py", line 605, in  
read_csv  
    return _read(filepath_or_buffer, kwds)  
...  
  File "pandas/_libs/parsers.pyx", line 1951, in  
pandas._libs.parsers.raise  
ParserError: Error tokenizing data. C error: Expected 1 field in  
line 12, saw 2
```

O traceback é a representação em string do erro, não o erro em si. Para verificar isso, adiciona outra linha de log logo após o bloco `except`:

```
try:  
    df = pd.read_csv(argument)  
    print(df.describe())  
except Exception:  
    logger.exception("Had a problem trying to read the CSV file")  
  
    logger.info("the program continues, without issue")
```

Volta a executar para verificar o resultado:

```
[describe][DEBUG ] processing input file: logging_describe.py  
[describe][ERROR ] Had a problem trying to read the CSV file
```

```
Traceback (most recent call last):
[...]
ParserError: Error tokenizing data. C error: Expected 1 field in
line 12, saw 2

[describe][INFO ] the program continues, without issue
```

Modificar os níveis de registo

Estes tipos de facilidades informativas fornecidas pelo registo não são simples com as instruções `print()`. Se estiveres a escrever um script de shell, isso seria quase impossível. No exemplo, temos agora três níveis de registo: debug, error e info. Outra coisa que o registo permite é definir seletivamente o nível para aquilo em que estamos interessados. Antes de o alterar, os níveis têm um *peso* associado, e é essencial compreendê-los para que as alterações reflectam a prioridade. Do mais para o menos verboso, a ordem é a seguinte:

1. `debug`
2. `info`
3. `warning`
4. `error`
5. `critical`

Embora este seja o módulo de registo Python, deves esperar uma prioridade ponderada semelhante de outros sistemas. Um nível de registo de `debug` irá incluir todos os outros níveis, assim como `debug`. Um nível de registo `critical` irá incluir apenas mensagens de nível `critical`. Actualiza o script mais uma vez para definir o nível de registo para `error`:

```
log_format = "[%(name)s]%(levelname)-6s %(message)s"
logging.basicConfig(format=log_format)
logger = logging.getLogger("describe")
logger.setLevel(logging.ERROR)
```

```
$ python logging_describe.py
[describe][ERROR ] Had a problem trying to read the CSV file
Traceback (most recent call last):
  File "logging_describe.py", line 15, in <module>
    df = pd.read_csv(argument)
  File "/.../site-packages/pandas/io/parsers.py", line 605, in
read_csv
    return _read(filepath_or_buffer, kwds)
...
  File "pandas/_libs/parsers.pyx", line 1951, in
pandas._libs.parsers.raise_
ParserError: Error tokenizing data. C error: Expected 1 field in
line 12, saw 2
```

A alteração faz com que apenas a mensagem do registo de erros seja apresentada. Isto é útil quando tentas reduzir a quantidade de registo para o que pode ser de interesse. A depuração abrange a maioria das mensagens (se não todas), enquanto os erros e as situações críticas são muito mais esporádicos e normalmente não são vistos com tanta frequência.

Recomendo que defina os níveis de registo de depuração para o novo código de produção, para que seja mais fácil detetar potenciais problemas. Deves esperar problemas ao produzir e implementar novo código. Uma saída altamente detalhada não é muito útil a longo prazo quando uma aplicação se mostrou estável e não há muitos problemas surpreendentes. Ajustar os níveis para `info` ou `error` apenas é algo que podes fazer progressivamente.

Registo de diferentes aplicações

Até agora, vimos níveis de registo de um script a tentar carregar um ficheiro CSV. E ainda não entrei em detalhes sobre o porquê do nome do registador ("`describe`" nos exemplos anteriores) ser significativo. Em Python, importas módulos e pacotes, e muitos destes pacotes vêm com os seus próprios loggers. A funcionalidade de registo permite-te definir opções particulares para estes registos independentemente da tua aplicação. Existe também uma hierarquia para os registadores, onde o registador "`raiz`" é o pai de todos os registadores e pode modificar as definições para todas as aplicações e registadores.

Alterar o nível de registo é uma das muitas coisas que podes configurar. Numa aplicação de produção, criei dois loggers para a mesma aplicação: um emitia mensagens para o terminal, enquanto o outro escrevia para um ficheiro de registo. Isto permitiu que as mensagens de fácil utilização fossem para o terminal, omitindo grandes rastreios e erros que poluíam o output. Ao mesmo tempo, o registo interno, orientado para o programador, iria para um ficheiro. Este é outro exemplo de algo que seria muito difícil e complicado de fazer com uma declaração `print()` ou uma diretiva `echo` num script de shell. Quanto mais flexibilidade tiveres, melhor poderás criar aplicações e serviços.

Cria um novo ficheiro e guarda-o como `http-app.py` com o seguinte conteúdo:

```
import requests
import logging

logging.basicConfig()

# new logger for this script
logger = logging.getLogger('http-app')

logger.info("About to send a request to example.com")
requests.get('http://example.com')
```

O script configura o recurso de registro com uma configuração básica que emite mensagens para o terminal por padrão. Em seguida, tenta fazer uma solicitação usando a biblioteca `requests`. Executa-o e verifica o resultado. Pode parecer surpreendente que nada apareça no terminal após a execução do script. Antes de explicares exatamente porque é que isto está a acontecer, actualiza o script:

```
import requests
import logging

logging.basicConfig()
root_logger = logging.getLogger()

# Sets logging level for every single app, the "parent" logger
root_logger.setLevel(logging.DEBUG)
```

```
# new logger for this script
logger = logging.getLogger('http-app')

logger.info("About to send a request to example.com")
requests.get('http://example.com')
```

Volta a executar o script e toma nota do resultado:

```
$ python http-app.py
INFO:http-app:About to send a request to example.com
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1):
example.com:80
DEBUG:urllib3.connectionpool:http://example.com:80 "GET /
HTTP/1.1" 200 648
```

O módulo de registo Python pode definir definições de configuração globalmente para cada um dos registadores em todos os pacotes e módulos. Este logger *pai* é chamado de logger *raiz*. A razão pela qual há saída é que as alterações definem o nível do logger raiz para debug. Mas há mais saída do que a única linha do logger *http-app*. Isso acontece porque o pacote *urllib3* também tem seu logger. Como o logger raiz alterou o nível de log global para debug, o pacote *urllib3* agora está emitindo essas mensagens.

É possível configurar vários registadores diferentes e afinar a granularidade e verbosidade dos níveis (bem como qualquer outra configuração de registo). Para demonstrar isto, altera o nível de registo do pacote *urllib3* adicionando estas linhas no final do script *http-app.py*:

```
# fine tune the urllib logger:
urllib_logger = logging.getLogger('urllib3')
urllib_logger.setLevel(logging.ERROR)

logger.info("About to send another request to example.com")
requests.get('http://example.com')
```

A versão atualizada recupera o logger para *urllib3* e altera seu nível de log para erro. O logger do script emite uma nova mensagem antes de chamar `requests.get()`, que por sua vez usa o pacote *urllib3*. Executa novamente o script para verificar a saída:

```
INFO:http-app:About to send a request to example.com
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1):
example.com:80
DEBUG:urllib3.connectionpool:http://example.com:80 "GET /
HTTP/1.1" 200 648
INFO:http-app:About to send another request to example.com
```

Como o nível de registo da *urllib3* foi alterado antes do último pedido, não aparecem mais mensagens de depuração. A mensagem de nível de informação aparece porque esse registrador ainda está configurado em um nível de depuração. Estas combinações de configurações de registo são poderosas porque te permitem selecionar o que é interessante de outras informações que podem causar *ruído* na saída.

Imagina a utilização de uma biblioteca que interage com uma solução de armazenamento Cloud. A aplicação que estás a desenvolver executa milhares de interações, descarregando, listando e carregando conteúdos para o servidor de armazenamento. Supõe que a principal preocupação da aplicação é gerir conjuntos de dados e descarregá-los para o fornecedor Cloud. Achas que seria interessante ver uma mensagem informativa a dizer que está prestes a ser feito um pedido ao fornecedor de Cloud? Na maioria das situações, eu diria que isso está longe de ser útil. Em vez disso, seria fundamental ser alertado quando a aplicação não consegue executar uma determinada operação com o armazenamento. Além disso, pode ser possível que estejas a lidar com timeouts ao pedir o armazenamento e, nesse caso, alterar o nível de registo para indicar quando o pedido acontece é crucial para obter o delta de tempo.

É tudo uma questão de flexibilidade e de adaptação às necessidades do ciclo de vida da tua aplicação. O que é útil hoje pode ser uma sobrecarga de informações amanhã. O registo anda de mãos dadas com a monitorização (abordada na próxima secção), e não é invulgar ouvir falar de observabilidade na mesma conversa. Estes são fundamentais para o DevOps e devem fazer parte do ciclo de vida do ML.

Monitorização e observabilidade

Quando eu era atleta profissional, o meu pai, que também era meu treinador, acrescentou uma tarefa particular que eu desprezava: escrever todos os dias, num diário, sobre o treino que eu tinha acabado de fazer. O diário precisava dos seguintes elementos:

- O exercício planeado. Por exemplo, se forem 10 repetições de 300 metros a um ritmo de 42 segundos.
- Os resultados do exercício. Para os 300 metros, seria o tempo real realizado em cada repetição.
- Como me senti durante e após a sessão.
- Qualquer outra informação relevante, como sentir-se doente ou ter dores devido a uma lesão, por exemplo.

Comecei a treinar profissionalmente aos 11 anos. Quando era adolescente, esta tarefa de escrever um diário parecia pior do que os piores treinos. Não compreendia a importância de escrever um diário e porque é que era vital para mim cobrir os detalhes do treino. Tive a coragem de dizer ao meu pai que isso parecia ser tarefa dele, não minha. Afinal de contas, eu já estava a fazer os exercícios. Esse argumento não me caiu muito bem. O problema é que eu não percebia. Parecia-me uma tarefa inútil, sem qualquer benefício. Não conseguia ver nem sentir os benefícios.

Parecia mais uma tarefa disciplinar no final do dia em vez de um aspeto crucial do treino. Alguns anos depois de ter feito um diário todos os dias (em média, fazia exercício 14 vezes por semana) e de ter tido uma grande época, sentei-me com o meu pai para planear a época seguinte. Em vez de passar duas horas a ouvir o meu pai falar-me do que estava para vir na época seguinte, ele iluminou-me demonstrando o poder do diário. *"Muito bem, Alfredo, deixa-me ir buscar os dois últimos diários, para ver o que fizemos e como te sentiste, para te adaptares, aumentares e teres uma grande época mais uma vez."*

Os diários continham todas as informações de que precisávamos para planear. Ele usou uma frase que me marcou durante anos, e espero que demonstre por que razão a monitorização e as métricas são fundamentais

em qualquer situação: "*Se pudermos medir, então podemos comparar. E se pudermos comparar, só então podemos melhorar.*"

As operações de aprendizagem automática não são diferentes. Quando uma nova iteração de um modelo entra em produção, tens de saber se o seu desempenho é melhor ou pior. Não só tens de saber, como a informação tem de ser acessível e fácil de produzir. Os processos podem criar fricção e fazer com que tudo corra mais devagar do que deveria. A automatização reduz os processos tolos e pode criar facilmente a disponibilidade da informação.

Há alguns anos, trabalhei numa startup onde o chefe de vendas me enviava um ficheiro CSV com novas contas para eu "fazer uns cálculos" e enviar-lhe um PDF com os resultados. Isto é horrível; não é escalável. E não sobrevive ao teste do autocarro.

O teste do autocarro é quando posso ser atropelado por um autocarro hoje, e tudo deve funcionar na mesma, e todos os que trabalham comigo podem compensar. Todo o esforço de automatização e produção de métricas é fundamental para enviar modelos robustos para a produção.

Noções básicas de monitorização de modelos

A monitorização nas operações de ML significa tudo o que tem a ver com a colocação de um modelo em produção - desde a captura de informações sobre os sistemas e serviços até ao desempenho do próprio modelo. Não existe uma única solução mágica para implementar a monitorização que faça com que tudo corra bem. Monitorizar e captar métricas é algo semelhante a conhecer os dados antes de descobrir que características e algoritmos utilizar para treinar um modelo. Quanto mais souberes sobre os dados, melhores decisões poderás tomar sobre o treino do modelo.

Da mesma forma, quanto mais souberes sobre as etapas envolvidas na colocação de um modelo em produção, melhores serão as escolhas que farás ao capturar métricas e definir alertas de monitorização. A resposta a "*que métrica deves captar quando treinas um modelo?*" é uma resposta que não gosto, mas que é tão precisa nesta situação: depende.

Embora existam diferenças entre as métricas e os tipos de alertas que podes definir, existem alguns padrões fundamentais úteis que podes utilizar por defeito. Estes padrões ajudarão a clarificar os dados a recolher, a frequência com que essa recolha deve ocorrer e a melhor forma de os visualizar. Por fim, dependendo da etapa de produção de um modelo, as principais métricas serão diferentes. Por exemplo, ao recolher dados e limpá-los, pode ser substancial detetar o número de valores vazios por coluna e talvez o tempo de conclusão ao processar os dados.

Na semana passada, estive a processar dados de vulnerabilidade do sistema. E tive de fazer algumas alterações à aplicação responsável por ler ficheiros JSON e guardar a informação numa base de dados. Depois de algumas alterações, a base de dados passou de vários gigabytes de tamanho para apenas 100 megabytes. A alteração do código não tinha qualquer intenção de reduzir o tamanho, por isso percebi imediatamente que tinha cometido um erro que precisava de ser corrigido. Encontrar estas situações é uma excelente oportunidade para determinar que métricas (e quando) devem ser capturadas.

Existem alguns tipos de métricas que podes encontrar na maioria dos sistemas de captura de métricas:

Contador

Como o nome indica, este tipo de métrica pode ser útil para contar qualquer tipo de item. É útil ao iterar sobre itens. Por exemplo, pode ser útil para contar valores de células vazias por coluna.

Temporizador

Um temporizador é excelente quando tentas determinar quanto tempo demorará uma ação. Isto é crucial para a monitorização do desempenho, uma vez que a sua funcionalidade de responsabilidade principal é medir o tempo gasto durante uma ação. O tempo gasto é comumente visto em gráficos de monitoramento para APIs HTTP hospedadas. Se tiveres um modelo hospedado, um temporizador ajudaria a capturar quanto tempo o modelo levou para produzir uma previsão sobre HTTP.

Valor

Quando os contadores e os temporizadores não se adequam à métrica a capturar, os valores são úteis. Costumo pensar em valores como partes de uma equação de álgebra: mesmo que eu não saiba o que é X , quero capturar seu valor e persisti-lo. Reutilizando o exemplo do processamento de arquivos JSON no trabalho e salvando informações em um banco de dados, um uso justo para essa métrica seria o tamanho (em gigabytes) do banco de dados resultante.

Há duas operações comuns específicas do ML que os fornecedores de Cloud precisam de monitorizar e capturar métricas úteis. A primeira é o conjunto de dados de destino. Este pode ser o mesmo conjunto de dados utilizado para treinar um modelo, embora seja necessário um cuidado especial para garantir que o número (e a ordem) das características não se altere. A segunda é a linha de base. A linha de base determina quais as diferenças que podem (ou não) ser aceitáveis quando um modelo é treinado. Pensa na linha de base como os limites aceitáveis para determinar a adequação de um modelo à utilização na produção.

Agora que os princípios básicos são claros e que um conjunto de dados alvo com uma linha de base é compreendido, vamos utilizá-los para capturar métricas úteis ao treinar modelos.

Monitoriza a deriva com o AWS SageMaker

Como já mencionei, os fornecedores de Cloud precisam normalmente de um conjunto de dados alvo e de uma linha de base. Isso não é diferente com a AWS. Nesta secção, vamos produzir métricas e capturar violações de dados de um modelo já implementado. O SageMaker é uma ferramenta incrível para inspecionar conjuntos de dados, modelos de treinamento e modelos de provisionamento em ambientes de produção. Como o SageMaker se integra perfeitamente com outras ofertas da AWS, como o armazenamento S3, podes aproveitar para guardar informações de destino que podem ser rapidamente processadas noutro local que tenha acesso.

Uma coisa que gosto particularmente no SageMaker é a sua oferta de Jupyter Notebook. A interface não é tão polida quanto o Colab do Google, mas traz recursos como o AWS SDK pré-instalado e tipos substanciais de kernel para escolher - desde o (agora obsoleto) Python 2.7 até ambientes Conda rodando Python 3.6, como mostrado na [Figura 6-3](#).

NOTA

Esta seção não aborda os detalhes da implantação de um modelo. Se quiseres aprofundar a implantação de modelos na AWS, consulte o [Capítulo 7](#).

Select Kernel

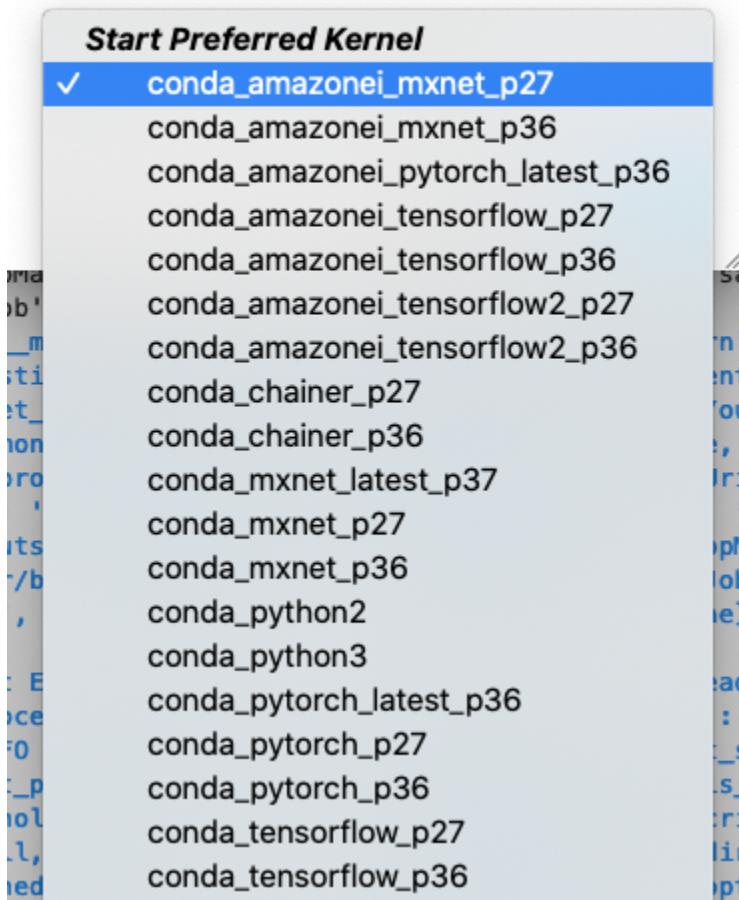


Figura 6-3. Núcleos SageMaker

Utiliza um bloco de notas do SageMaker para esta secção. Faz login no console do AWS e encontra o serviço SageMaker. Depois de carregado, na coluna da esquerda, encontra o link Instâncias de notebook e clica nele. Cria uma nova instância com um nome significativo. Não é necessário alterar nenhuma das predefinições, incluindo o tipo de máquina. Eu chamei o meu notebook de *practical-mlops-monitoring* (veja a Figura 6-4).

Ao implantares o modelo, é importante ativar a captura de dados. Por isso, certifica-te de que utilizas a classe DataCaptureConfig para o fazer. Este é um exemplo rápido que o salva em um bucket S3:

```
from sagemaker.model_monitor import DataCaptureConfig

s3_capture_path = "s3://monitoring/xgb-churn-data"

data_capture_config = DataCaptureConfig(
    enable_capture=True,
    sampling_percentage=100,
    destination_s3_uri=s3_capture_path
)
```

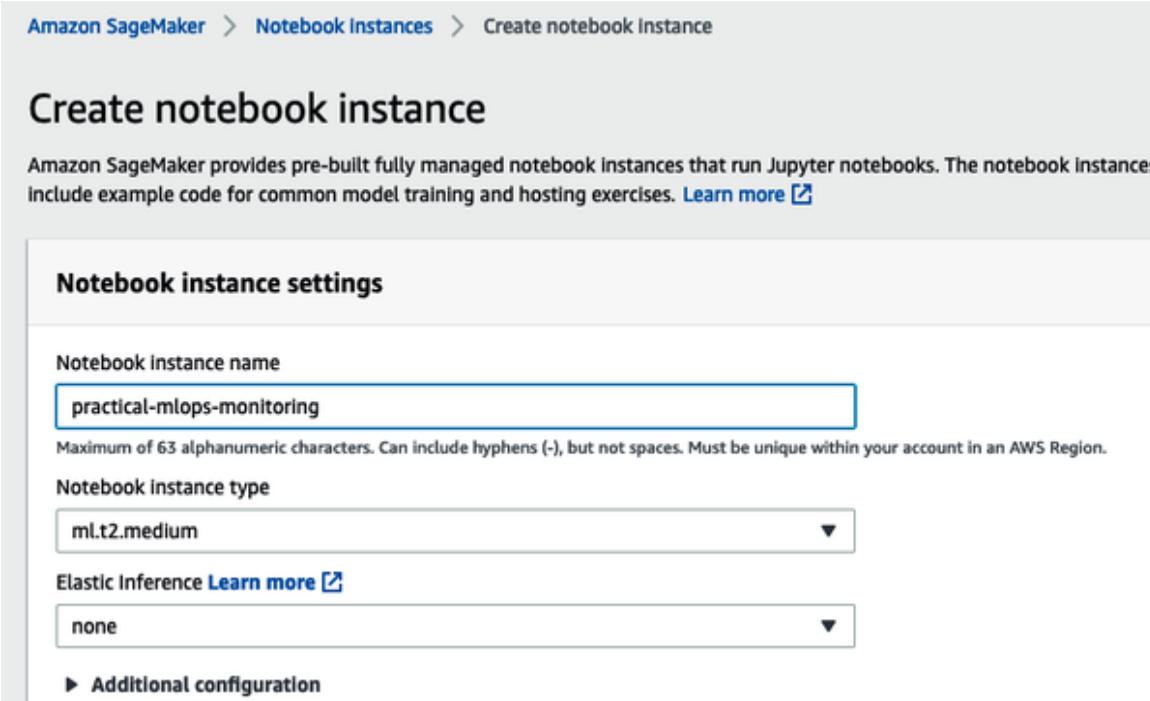


Figura 6-4. Instância do notebook do SageMaker

Utiliza o `data_capture_config` ao chamar `model.deploy()`. Neste exemplo, criei anteriormente um objeto `model` usando a classe `Model()` e atribuí a configuração de captura de dados a ele, para que, quando o modelo for exercitado, os dados sejam salvos no bucket do S3:

```
from sagemaker.deserializers import CSVDeserializer

predictor = model.deploy(
    initial_instance_count=1,
    instance_type="ml.m4.large",
    endpoint_name="xgb-churn-monitor",
    data_capture_config=data_capture_config,
    deserializer=CSVDeserializer(),
)
```

Depois que o modelo for implantado e estiver disponível, envia algumas solicitações para começar a fazer previsões. Ao enviar pedidos para o modelo, estás a fazer com que a configuração de captura guarde os dados críticos necessários para criar a linha de base. Podes enviar pedidos de previsão para o modelo de qualquer forma. Neste exemplo, estou a utilizar o SDK para enviar alguns pedidos utilizando dados CSV de amostra de um ficheiro. Cada linha representa dados que o modelo pode usar para iniciar a previsão. Como a entrada é de dados, é por isso que estou a utilizar um desserializador de CSV, para que o ponto final compreenda como consumir essa entrada:

```
from sagemaker.predictor import Predictor
from sagemaker.serializers import CSVDeserializer, CSVSerializer
import time

predictor = Predictor(
    endpoint_name=endpoint_name,
    deserializer=CSVDeserializer(),
    serializer=CSVSerializer(),
)

# About one hundred requests should be enough from test_data.csv
with open("test_data.csv") as f:
    for row in f:
        payload = row.rstrip("\n")
```

```
response = predictor.predict(data=payload)
time.sleep(0.5)
```

Após a execução, verifica novamente se há saída capturada no bucket S3. Podes listar o conteúdo desse bucket para garantir que existem dados reais. Neste caso, vou utilizar a ferramenta de linha de comandos da AWS, mas podes utilizar a interface Web ou o SDK (o método não importa neste caso):

```
$ aws s3 ls \
  s3://monitoring/xgb-churn-
data/datacapture/AllTraffic/2021/02/03/13/
2021-02-03 08:13:33  61355 12-26-957-d5938b7b-fbd8-4e3c-9dbd-
741f71b.jsonl
2021-02-03 08:14:33    1566 13-27-365-a59180ea-591d-4562-925b-
6472d55.jsonl
2021-02-03 08:33:33    31548 32-24-577-20217dd9-8bfa-4ba2-a7f1-
d9717ef.jsonl
2021-02-03 08:34:33    31373 33-25-476-0b843e95-5fe0-4b79-8369-
b099d0e.jsonl
[...]
```

O bucket lista cerca de 30 itens, confirmando que as solicitações de previsão foram bem-sucedidas e que os dados foram capturados e salvos no bucket S3. Cada arquivo tem uma entrada JSON com algumas informações. Os detalhes de cada entrada são difíceis de entender. Vê o aspetto de uma das entradas:

```
{
  "captureData": {
    "endpointInput": {
      "observedContentType": "text/csv",
      "mode": "INPUT",
      "data": [
        "92,0,176.3,85,93.4,125,207.2,107,9.6,1,2,0,1,00,0,0,1,1,0,1,0",
        "encoding": "CSV"
      ],
      [...]
    }
}
```

Mais uma vez, as entradas fazem referência ao tipo de conteúdo CSV. Isso é crucial para que outros consumidores desses dados possam consumir as

informações corretamente. Até agora, configuramos o modelo para capturar dados e salvá-los em um bucket S3. Tudo isso aconteceu depois de gerar algumas previsões com dados de teste. No entanto, ainda não há uma linha de base. Os dados capturados na etapa anterior são necessários para criar a linha de base. A próxima etapa requer um conjunto de dados de treinamento de destino. Como mencionei anteriormente, o conjunto de dados de treinamento pode ser o mesmo usado para treinar o modelo. Um subconjunto do conjunto de dados pode ser aceitável se o modelo resultante não mudar drasticamente. Este conjunto de *dados alvo* tem de ter as mesmas características (e na mesma ordem) que o conjunto de dados utilizado para treinar o modelo de produção.

NOTA

É comum encontrar documentação online que se refere ao *conjunto de dados de base* de forma intercambiável com o *conjunto de dados alvo*, uma vez que ambos podem inicialmente ser o mesmo. Isto pode tornar confusa a compreensão de alguns destes conceitos. É útil pensar no conjunto de dados de base como os dados utilizados para criar o padrão de ouro (a base de referência) e quaisquer dados mais recentes como o *objetivo*.

O SageMaker facilita o salvamento e a recuperação de dados ao confiar no S3. Já defini localizações ao longo dos exemplos do SDK e, para a linha de base, vou fazer o mesmo. Começa por criar um objeto monitor; este objeto é capaz de produzir uma linha de base e guardá-la no S3:

```
from sagemaker.model_monitor import DefaultModelMonitor

role = get_execution_role()

monitor = DefaultModelMonitor(
    role=role,
    instance_count=1,
    instance_type="ml.m5.xlarge",
    volume_size_in_gb=20,
    max_runtime_in_seconds=3600,
)
```

Agora que o monitor está disponível, utiliza o método `suggest_baseline()` para produzir uma linha de base predefinida para o modelo:

```
from sagemaker.model_monitor.dataset_format import DatasetFormat
from sagemaker import get_execution_role

s3_path = "s3://monitoring/xgb-churn-data"

monitor.suggest_baseline(
    baseline_dataset=s3_path + "/training-dataset.csv",
    dataset_format=DatasetFormat.csv(header=True),
    output_s3_uri=s3_path + "/baseline/",
    wait=True,
)
```

Quando a execução estiver concluída, será produzida uma grande quantidade de resultados. O início da saída deve ser semelhante a isto:

```
Job Name: baseline-suggestion-job-2021-02-03-13-26-09-164
Inputs: [{"InputName": 'baseline_dataset_input', 'AppManaged': False, ...}]
Outputs: [{"OutputName": 'monitoring_output', 'AppManaged': False, ...}]
```

Deves ter dois ficheiros guardados no bucket S3 configurado: `constraints.json` e `estatistics.json`. Podes visualizar as restrições com a biblioteca Pandas:

```
import pandas as pd

baseline_job = monitor.latest_baselining_job
constraints = pd.json_normalize(
    baseline_job.baseline_statistics().body_dict["features"]
)
schema_df.head(10)
```

Este é um pequeno subconjunto da tabela de restrições que o Pandas gera:

name	inferred_type	completeness
num_constraints.is_non_negative		

```

Churn           Integral 1.0      True
Account Length Integral 1.0      True
Day Mins   Fractional 1.0       True
[...]

```

Agora que temos uma linha de base feita com um conjunto de dados muito semelhante ao usado para treinar o modelo de produção e capturamos as restrições relevantes, é hora de analisá-la e monitorar o desvio de dados. Até agora, houve várias etapas envolvidas, mas a maioria delas não mudará muito destes exemplos para outros mais complexos, o que significa que há muitas oportunidades aqui para automatizar e abstrair muito. A recolha inicial de dados ocorre uma vez, quando se define a linha de base, e depois não deve mudar, a menos que sejam necessárias alterações à linha de base. Estas provavelmente não acontecerão com frequência, pelo que o trabalho pesado de definir a linha de base não deve parecer um fardo.

Para analisar os dados coletados, precisamos de um cronograma de monitoramento. O cronograma de exemplo será executado a cada hora, usando a linha de base criada nas etapas anteriores para comparar com o tráfego:

```

from sagemaker.model_monitor import CronExpressionGenerator

schedule_name = "xgb-churn-monitor-schedule"
s3_report_path = "s3://monitoring/xgb-churn-data/report"

monitor.create_monitoring_schedule(
    monitor_schedule_name=schedule_name,
    endpoint_input=predictor.endpoint_name,
    output_s3_uri=s3_report_path,
    statistics=monitor.baseline_statistics(),
    constraints=monitor.suggested_constraints(),
    schedule_cron_expression=CronExpressionGenerator.hourly(),
    enable_cloudwatch_metrics=True,
)

```

Depois de criares o calendário, este necessitará de tráfego para gerar relatórios. Se o modelo já estiver em um ambiente de produção, podemos assumir (e reutilizar) o tráfego existente. Se estiveres a testar a linha de base num modelo de teste, como fiz nestes exemplos, terás de gerar tráfego

invocando um pedido ao modelo implementado. Uma maneira simples de gerar tráfego é reutilizar o conjunto de dados de treinamento para invocar o ponto de extremidade.

O modelo que implementei funcionou durante algumas horas. Usei este script para gerar algumas previsões a partir do modelo implantado anteriormente:

```
import boto3
import time

runtime_client = boto3.client("runtime.sagemaker")

with open("training-dataset.csv") as f:
    for row in f:
        payload = row.rstrip("\n")
        response = runtime_client.invoke_endpoint(
            EndpointName=predictor.endpoint_name,
            ContentType="text/csv",
            Body=payload
        )
        time.sleep(0.5)
```

Como configurei o cronograma de monitoramento para capturar a cada hora, o SageMaker não gerará imediatamente relatórios no bucket S3. Após duas horas, é possível listar o bucket S3 e verificar se os relatórios aparecem lá.

NOTA

Embora o monitor de modelo seja executado de hora em hora, o AWS tem um buffer de 20 minutos que pode causar um atraso de até 20 minutos após a marca da hora. Se já viste outros sistemas de agendamento, este buffer pode ser surpreendente. Isto acontece porque, nos bastidores, a AWS está a equilibrar a carga dos recursos para o agendamento.

Os relatórios são compostos por três ficheiros JSON:

- *constraint_violations.json*

- *constraint.json*
- *estatísticas.json*

As informações interessantes relacionadas ao monitoramento e à captura de desvios estão no arquivo *constraint_violations.json*. No meu caso, a maioria das violações se parece com esta entrada:

```
feature_name: State_MI
constraint_check_type: data_type_check
description:
Data type match requirement is not met. Expected data type:
Integral,
Expected match: 100.0%. Observed: Only 99.71751412429379% of
data is Integral.
```

A linha de base sugerida exigia 100% de integridade dos dados, e aqui estamos a ver que o modelo está suficientemente próximo, com 99,7%. Como a restrição é atender a 100%, a violação é gerada e relatada. A maioria desses números é semelhante no meu caso, exceto por uma linha:

```
feature_name: Churn
constraint_check_type: data_type_check
description:
Data type match requirement is not met. Expected data type:
Integral,
Expected match: 100.0%. Observed: Only 0.0% of data is Integral.
```

Um 0% é uma situação crítica aqui, e é aqui que todo o trabalho árduo de configurar os sistemas para detetar e relatar essas mudanças nas previsões é crucial. Quero enfatizar que, embora tenha sido necessário seguir várias etapas e código padrão com o SDK Python da AWS, não é muito complicado automatizar e começar a gerar esses relatórios automaticamente para conjuntos de dados de destino. Eu criei a linha de base com uma sugestão automatizada, e isso exigirá principalmente um ajuste fino para definir valores aceitáveis para evitar a geração de violações que não são úteis.

Monitorizar a deriva com o Azure ML

O MLOps desempenha um papel significativo nos fornecedores de Cloud. Não é surpreendente que a monitorização e os alertas (pilares fundamentais do DevOps) sejam oferecidos com serviços bem pensados. O Azure pode analisar a deriva de dados e definir alertas para capturar possíveis problemas antes que os modelos entrem em produção. É sempre útil ver como diferentes fornecedores de Cloud resolvem um problema como o desvio de dados - a perspetiva é um ativo inestimável e fará de ti um melhor engenheiro. A quantidade de ideias, documentação e exemplos na plataforma Azure contribui para um processo de integração mais suave. Não é preciso muito esforço para encontrar recursos de aprendizagem para se familiarizar com as ofertas do Azure.

NOTA

No momento em que escrevemos este artigo, a deteção de desvios de dados no Azure ML está em pré-visualização e ainda é necessário resolver alguns problemas menores que impedem a apresentação de exemplos de código sólidos para experimentar.

A deteção de desvio de dados no Azure funciona de forma semelhante à utilização do AWS SageMaker. O objetivo é ser alertado quando ocorre um desvio entre os conjuntos de dados de treinamento e de serviço. Tal como acontece com a maioria das operações de ML, é fundamental ter uma compreensão profunda dos dados (e, portanto, do conjunto de dados). Um exemplo demasiado simplista é um conjunto de dados que capta as vendas de fatos de banho ao longo de um ano: se o número de vendas por semana cair para zero, isso significa que o conjunto de dados se desviou e não deve ser utilizado na produção? Ou que provavelmente estamos a meio do inverno e ninguém está a comprar nada? Estas perguntas abertas são fáceis de responder quando os pormenores dos dados são bem compreendidos.

Existem várias causas para o desvio de dados, muitas das quais podem ser totalmente inaceitáveis. Alguns exemplos destas causas envolvem alterações nos tipos de valores (por exemplo, de Fahrenheit para Celsius),

valores vazios ou nulos ou, no exemplo da venda de fatos de banho, um *desvio natural*, em que as alterações sazonais podem afetar as previsões.

Os padrões para configurar e analisar o desvio no Azure exigem um conjunto de dados de linha de base, um conjunto de dados de destino e um monitor. Esses três requisitos trabalham juntos para produzir métricas e criar alertas sempre que o desvio for detectado. O fluxo de trabalho de monitoramento e análise permite que você detecte e alerte o desvio de dados quando houver novos dados em um conjunto de dados, permitindo a criação de perfis de novos dados ao longo do tempo. Provavelmente, não utilizarás tanto a criação de perfis históricos como a deteção de desvios atuais, porque é mais comum utilizar o ponto de comparação mais atual do que comparar com vários meses atrás. Ainda assim, é útil ter um comparador onde o desempenho do ano anterior é mais significativo do que a comparação com o mês passado. Isto é particularmente verdade com conjuntos de dados que são afectados por eventos sazonais. Não é muito útil comparar as vendas de árvores de Natal com as métricas de agosto.

Para configurar um fluxo de trabalho de desvio de dados no Azure, tens de começar por criar um conjunto de dados de destino. O conjunto de dados de destino requer uma *série temporal* definida nele usando uma coluna de carimbo de data/hora ou uma *coluna virtual*. A coluna virtual é uma funcionalidade interessante porque infere o carimbo de data/hora a partir do caminho onde o conjunto de dados está armazenado. Este atributo de configuração é chamado de *formato de partição*. E se estiveres a configurar um conjunto de dados para utilizar a coluna virtual, verás um formato de partição referenciado no Azure ML Studio e no Python SDK (ver [Figura 6-5](#)).

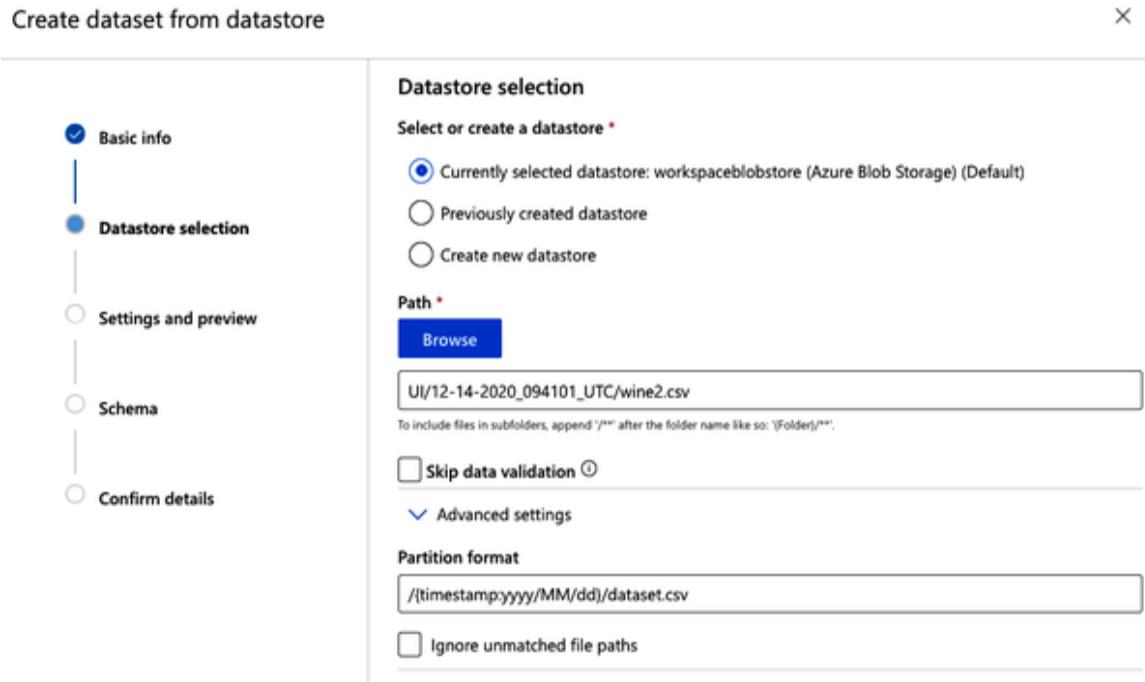


Figura 6-5. Formato da partição do Azure

Neste caso, estou a utilizar um auxiliar de formato de partição para que o Azure possa utilizar o caminho para inferir o carimbo de data/hora. Isso é bom porque instrui o Azure de que a convenção está definindo o padrão. Um caminho como /2021/10/14/dataset.csv significa que o conjunto de dados terminará com uma coluna virtual de 14 de outubro de 2021. A configuração por convenção é uma pepita de ouro da automatização. Sempre que vires uma oportunidade para abstrair (ou remover completamente) a configuração que pode ser inferida por uma convenção (como um caminho, neste caso), deves tirar partido dela. Menos configuração significa menos sobrecarga, o que permite fluxos de trabalho mais rápidos.

Assim que tiveres um conjunto de dados de séries cronológicas, podes continuar a criar um monitor de conjunto de dados. Para que tudo funcione, precisas de um conjunto de dados alvo (neste caso, o conjunto de dados de séries temporais), o conjunto de dados de base e as definições do monitor.

O conjunto de dados de base tem de ter as mesmas características (ou tão semelhantes quanto possível) que as do conjunto de dados alvo. Uma característica interessante é a seleção de um intervalo de tempo para cortar o

conjunto de dados com dados relevantes para a tarefa de monitorização. A configuração do monitor é o que reúne todos os conjuntos de dados.

Permite criar uma agenda para ser executada com um conjunto de características interessantes e definir o limite para a percentagem de desvio de dados tolerável.

Os resultados do desvio estarão disponíveis no separador Monitores de conjunto de dados na secção Ativos no Azure ML Studio. Todos os monitores configurados estão disponíveis com métricas destacadas sobre a magnitude do desvio e uma lista ordenada dos principais recursos com desvio. A simplicidade da exposição do desvio de dados é algo que me agrada no Azure ML Studio, porque pode revelar rapidamente as informações essenciais úteis para tomar decisões corretivas.

Se for necessário aprofundar os detalhes das métricas, pode utilizar o Application Insights para consultar regtos e métricas associados aos monitores. A ativação e a configuração do Application Insights são abordadas em "[Application Insights](#)".

Conclusão

O registo, a monitorização e as métricas são tão importantes que qualquer modelo que entre em produção *tem de os implementar*, sob risco de uma falha catastrófica difícil de recuperar. A alta confiança em processos robustos para resultados reproduzíveis requer todos esses componentes. Como mencionei ao longo deste capítulo, tens de tomar decisões com dados precisos que te possam dizer se a precisão é tão elevada como sempre ou se o número de erros aumentou significativamente.

Muitos exemplos podem ser difíceis de compreender, mas todos eles podem ser automatizados e abstraídos. Ao longo deste livro, lerás constantemente sobre os principais pilares do DevOps e como eles são relevantes para operacionalizar a aprendizagem automática. A automação é o que une pilares como o registo e o monitoramento. A configuração do registo e da monitorização não é normalmente um trabalho excitante, especialmente se a ideia for obter modelos de previsão de última geração que façam um

trabalho excepcional. Mas resultados excepcionais não podem acontecer de forma consistente com uma base quebrada.

Quando comecei a treinar para ser um atleta profissional, sempre duvidei do meu treinador, que não me deixava fazer o salto em altura todos os dias. *"Antes de te tornares um saltador de altura, tens de te tornar um atleta."* Bases fortes permitem resultados fortes, e em DevOps e MLOps, isto não é diferente.

Nos próximos capítulos, terás a oportunidade de te aprofundar em cada um dos três principais fornecedores de Cloud, nos seus conceitos e nas suas ofertas de aprendizagem automática.

Exercícios

- Utiliza um conjunto de dados diferente e cria um relatório de deriva com violações utilizando o AWS SageMaker.
- Adiciona o registo Python a um script que irá registar erros em STDERR, instruções de informação em STDOUT, e todos os níveis num ficheiro.
- Cria um conjunto de dados de séries temporais no Azure ML Studio.
- Configura um monitor de conjunto de dados no Azure ML Studio que enviará um e-mail quando for detectado um desvio para além do limite aceitável.

Questões para discussão sobre pensamento crítico

- Porque é que é desejável registrar em várias fontes ao mesmo tempo?
- Porque é que é fundamental monitorizar o desvio de dados?

- Indica três vantagens da utilização de recursos de registo em relação às declarações `print()` ou `echo`.
- Enumera os cinco níveis de registo mais comuns, do menos para o mais detalhado.
- Quais são os três tipos de métricas comuns encontradas nos sistemas de captura de métricas?

Capítulo 7. MLOps para AWS

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

Toda a gente tinha medo dele [Dr. Abbott (porque gritava com toda a gente)]. Quando eu era assistente, havia um novo residente chamado Harris. O Harris continuava a ter medo dele [Dr. Abbott], apesar de ele ser o chefe dos residentes e estar lá há 5 anos. Mais tarde, [o Dr. Abbott] tem um ataque cardíaco e depois o seu coração pára. Uma enfermeira grita: "Rápido, ele acabou de ter uma paragem, entra!" Então Harris entrou lá... apoiado no esterno e partindo costelas e assim. Então o Harris começou a bombardear o Abbott e ele acordou. Acordou! Olhou para o Harris e disse: "Tu! Pára com isso!" E o Harris parou. E essa foi a última história sobre o Abbott.

—Dr. Joseph Bogen

Uma das perguntas mais comuns que recebo dos alunos é: "Que Cloud devo escolher?" Digo-lhes que a escolha segura é a Amazon. Tem a mais vasta seleção de tecnologia e a maior quota de mercado. Depois de dominares a Cloud AWS, é mais fácil dominar outras ofertas de Cloud, uma vez que também assumem que já conheces a AWS. Este capítulo aborda os fundamentos da AWS para MLOps e explora padrões práticos de MLOps.

Tenho uma longa e rica história de trabalho com a AWS. Numa rede social de desporto que dirigi como CTO e Diretor-Geral, a AWS foi um ingrediente secreto que nos permitiu escalar para milhões de utilizadores em todo o mundo. Também trabalhei como SME (especialista no assunto) na certificação de aprendizagem automática da AWS de raiz nos últimos anos. Fui reconhecido como um [AWS ML Hero](#), também faço parte do programa [AWS Faculty Cloud Ambassador](#) e ensinei certificações de Cloud

a milhares de estudantes na UC Davis, Northwestern, Duke e na Universidade do Tennessee. Por isso, podes dizer que sou um fã da AWS!

Devido ao tamanho da plataforma AWS, é impossível cobrir todos os aspectos do MLOps. Se pretenderes uma cobertura mais exaustiva de todas as opções disponíveis na plataforma AWS, podes também consultar o livro *Data Science on AWS*, de Chris Fregly e Antje Barth (O'Reilly), do qual fui um dos editores técnicos.

Em vez disso, este capítulo se concentra em serviços de nível superior, como o AWS Lambda e o AWS App Runner. Sistemas mais complicados, como o AWS SageMaker, são abordados em outros capítulos deste livro e do livro mencionado anteriormente. A seguir, vamos começar a criar soluções AWS MLOps.

Introdução à AWS

A AWS é líder em Cloud Computing por várias razões, incluindo o seu início precoce e a sua cultura empresarial. Em 2005 e 2006, a Amazon lançou os Amazon Web Services, incluindo o MTurk (Mechanical Turk), o Amazon S3, o Amazon EC2 e o Amazon SQS.

Até hoje, estas ofertas principais não só continuam a existir, como melhoram a cada trimestre e ano. A razão pela qual a AWS continua a melhorar as suas ofertas deve-se à sua cultura. Dizem que na Amazon é sempre "Dia 1", o que significa que a mesma energia e entusiasmo do Dia 1 deve acontecer todos os dias. Também colocam o cliente no "centro de tudo" que fazem. Utilizei estes blocos de construção para construir sistemas escalados para milhares de nós para realizar tarefas de aprendizagem automática, visão computacional e IA. **A Figura 7-1** é um desenho real no quadro branco de uma arquitetura técnica da AWS Cloud mostrada em um local de coworking no centro de São Francisco.

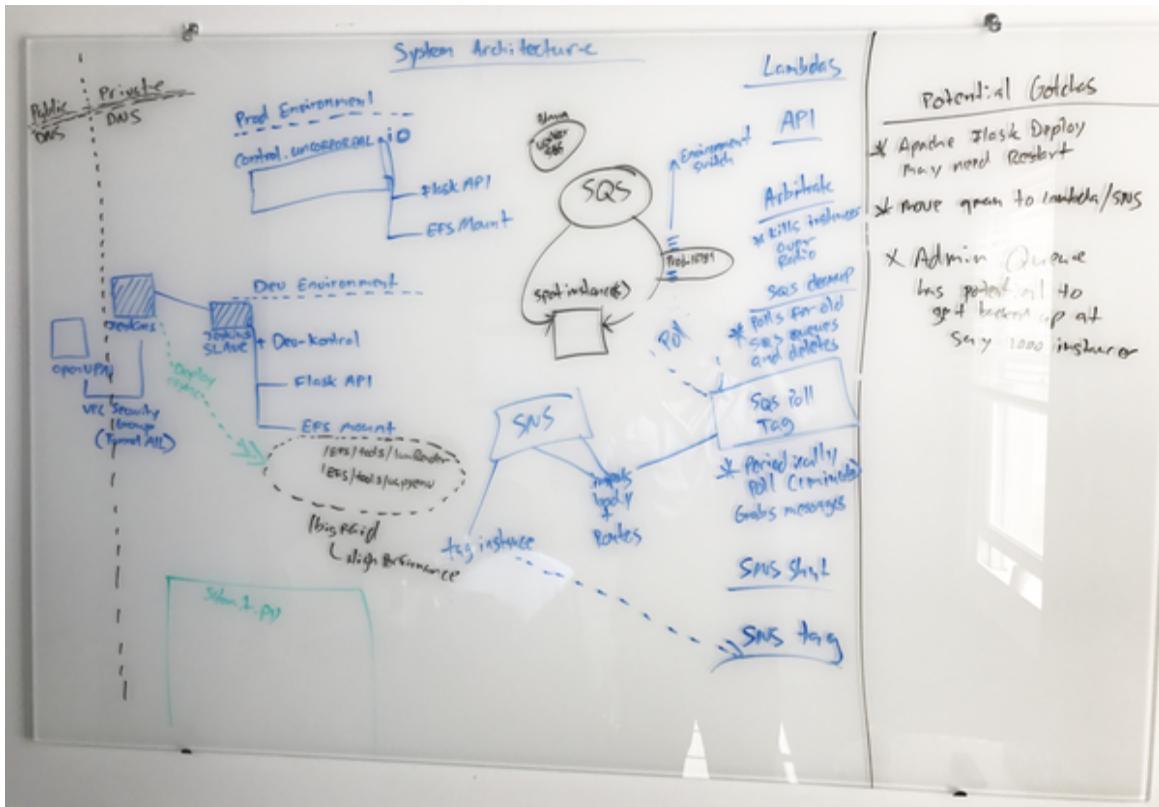


Figura 7-1. Arquitetura da AWS Cloud no quadro branco

Este tipo de cultura é muito diferente da da Google, que tem tido dificuldades com a computação em Cloud. A cultura da Google é orientada para a pesquisa, com muitas contratações académicas. Os benefícios desta cultura são projectos de código aberto como o Kubernetes e o TensorFlow. Ambos são maravilhas complexas da engenharia. A desvantagem é que o cliente não está em primeiro lugar na cultura, o que prejudicou a Google na quota de mercado da computação em Cloud. Muitas organizações recusam-se a comprar serviços profissionais e a apoiar algo tão crítico como a computação em Cloud.

De seguida, vamos ver como começar a utilizar os serviços AWS.

Começa a utilizar os serviços AWS

Para começares a utilizar a AWS, inicialmente apenas precisas de uma **conta de nível gratuito**. Se estiveres na universidade, também podes utilizar a **AWS Academy** e o **AWS Educate**. A AWS Academy fornece material de

certificação prático e laboratórios. O AWS Educate fornece sandboxes para salas de aula.

Assim que tiveres uma conta, o passo seguinte é experimentar as várias ofertas. Uma maneira de pensar na AWS é compará-la a uma loja de atacado a granel. Por exemplo, de acordo com o [statista](#), a Costco tem 795 lojas em todo o mundo, em vários países, e cerca de 1 em cada 5 americanos faz compras lá. Da mesma forma, de acordo com o [whitepaper de 2020 da Amazon Web Services](#), a AWS fornece uma infraestrutura que "alimenta centenas de milhares de empresas em 190 países em todo o mundo".

Na Costco, existem três abordagens consideradas relevantes para uma analogia com a AWS:

- No primeiro cenário, um cliente pode entrar e encomendar uma pizza a granel muito barata, mas de qualidade razoável.
- Num segundo cenário, um cliente novo no Costco precisa de descobrir todos os artigos a granel disponíveis para investigar a melhor forma de utilizar o Costco. Terá de percorrer a loja e olhar para os artigos a granel, como o açúcar, para ver o que pode fazer com estes ingredientes em bruto.
- Num terceiro cenário, quando um cliente da Costco conhece as refeições pré-preparadas (como frango assado, pizza, etc.) e conhece todos os ingredientes crus que pode comprar, pode, em teoria, abrir uma empresa de catering, uma charcutaria local ou um restaurante utilizando o seu conhecimento da Costco e dos serviços que estaoferece.

Um grande retalhista da dimensão da Costco cobra mais dinheiro pela comida preparada e menos dinheiro pelos ingredientes crus. O cliente da Costco que possui um restaurante pode escolher um nível diferente de preparação de alimentos, dependendo da maturidade da sua organização e do problema que pretende resolver. [A Figura 7-2](#) ilustra como estas opções da Costco se comparam com as opções da AWS.

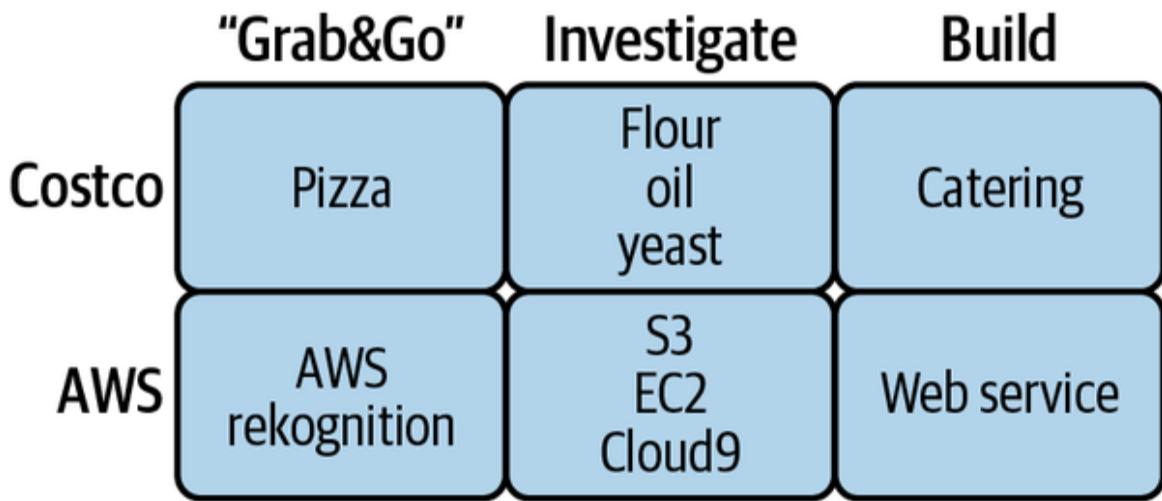


Figura 7-2. Costco versus AWS

Vejamos o caso de uma banca local de Poke Bowl do Havai, perto de uma famosa praia do Havai. O proprietário pode comprar o Poke pré-fabricado da Costco a granel e vendê-lo a um preço que é aproximadamente o dobro do seu custo. Mas, por outro lado, um restaurante de churrasco mais maduro no Havai, com empregados que sabem cozinar e preparar os alimentos, a Costco vende estes alimentos crus a um preço muito inferior ao do Poke totalmente preparado.

Tal como a Costco, a AWS fornece diferentes níveis de produtos e cabe ao cliente decidir a quantidade que vai utilizar. Vamos analisar um pouco essas opções.

Utiliza a solução "No Code/Low Code" do AWS Comprehend

O último exemplo mostrou como a utilização da Costco pode beneficiar diferentes empresas de restauração, desde uma com pouco ou nenhum pessoal - como um stand pop-up - até um estabelecimento mais extenso com mesa. Quanto mais trabalho o Costco fizer na preparação dos alimentos, maior será o benefício para o cliente que os compra e também maior será o custo dos alimentos. O mesmo conceito aplica-se à AWS; quanto mais trabalho a AWS fizer por ti, mais pagas e menos pessoas precisas para manter o serviço.

NOTA

Em economia, a teoria da vantagem comparativa diz que não deves comparar o custo de algo diretamente. Em vez disso, seria útil se comparasses o custo de oportunidade de o fazeres tu mesmo. Todos os fornecedores de Cloud têm este pressuposto incorporado, uma vez que a sua especialização é gerir um centro de dados e criar serviços em cima desse centro de dados. Uma organização que faz MLOps deve concentrar-se na criação de um produto para os seus clientes que gere receitas, e não recriar o que os fornecedores de Cloud fazem mal.

Com a AWS, um excelente ponto de partida é agir como o cliente Costco que encomenda pizza Costco a granel. Da mesma forma, uma empresa da Fortune 500 pode ter requisitos essenciais para adicionar processamento de linguagem natural (PNL) aos seus produtos de serviço ao cliente. Pode passar nove meses a um ano a contratar uma equipa e depois a desenvolver estas capacidades, ou pode começar a utilizar serviços valiosos de alto nível como o [AWS Comprehend for Natural Language Processing](#). O AWS Comprehend também permite que os utilizadores aproveitem a API da Amazon para realizar muitas operações de PNL, incluindo as seguintes:

- Deteção de entidades
- Deteção de frases-chave
- PII
- Deteção de línguas
- Sentimento

Por exemplo, podes cortar e colar texto na consola do Amazon Comprehend, e o AWS Comprehend encontrará todas as entidades do texto. No exemplo da [Figura 7-3](#), peguei o primeiro parágrafo da biografia de LeBron James na Wikipedia, coleei-o no console, cliquei em Analisar e ele destacou as entidades para mim.

Input text

all time.[1] Playing for the Cleveland Cavaliers, Miami Heat, and Los Angeles Lakers, James is the only player in NBA history to have won NBA championships with three franchises as Finals MVP.[2] He has competed in ten NBA Finals, including eight consecutive with the Heat and Cavaliers from 2011 through 2018. His accomplishments include four NBA championships, four NBA Most Valuable Player (MVP) Awards, four Finals MVP Awards, and two Olympic gold medals. During his 17-year career, James holds the record for all-time playoffs points, is third in all-time points, and eighth in career assists. James has been selected to the All-NBA First Team a record 13 times, made the All-Defensive First Team five times, and has been named an All-Star 17 times, including three All-Star MVP selections.

1134 of 5000 characters used.

Clear text Analyze

Insights Info

Entities Key phrases Language PII Sentiment Syntax

Analyzed text

LeBron Raymone James Sr. (/lə'brən/; born December 30, 1984) is an American professional basketball player for the Los Angeles Lakers of the National Basketball Association (NBA). Widely considered one of the greatest NBA players in history, James is frequently compared to Michael Jordan in debates over the greatest basketball player of all time.[1] Playing for the Cleveland Cavaliers, Miami Heat, and Los Angeles Lakers, James is the only player in NBA history to have won NBA championships with three franchises as Finals MVP.[2] He has competed in ten NBA Finals, including eight consecutive with the Heat and Cavaliers from 2011 through 2018. His accomplishments include four NBA championships, four NBA Most Valuable Player (MVP) Awards, four Finals MVP Awards, and two Olympic gold medals. During his 17-year career, James holds the record for all-time playoffs points, is third in all-time points, and eighth in career assists. James has been selected to the All-NBA First Team a record 13 times, made the All-Defensive First Team five times, and has been named an All-Star 17 times, including three

Figura 7-3. AWS Comprehend

Outros casos de utilização, como a revisão de regtos médicos ou a determinação do sentimento de uma resposta de serviço ao cliente, são igualmente simples com o AWS Comprehend e o boto3 Python SDK. Em seguida, vamos cobrir um projeto "hello world" para DevOps na AWS, implantando um site estático usando o Amazon S3.

Utilizar os sítios Web estáticos do Hugo S3

No cenário seguinte, uma excelente forma de explorar a AWS é "passar" pela consola, tal como se passeia pela Costco com admiração na primeira vez que a visita. Faz isso observando primeiro os componentes fundamentais do AWS, ou seja, IaaS (infraestrutura como código). Estes

serviços principais incluem o armazenamento de objectos AWS S3 e as máquinas virtuais AWS EC2.

Aqui, vou mostrar-te como implementar um **site Hugo** no **alojamento de sites estáticos AWS S3**, utilizando o "hello world" como exemplo. A razão para fazer um hello world com o Hugo é que é relativamente simples de configurar e dar-te-á uma boa compreensão dos serviços de alojamento utilizando a infraestrutura principal. Estas competências serão úteis mais tarde, quando aprenderes a implementar aplicações de aprendizagem automática utilizando a entrega contínua.

NOTA

Vale a pena notar que o Amazon S3 é de baixo custo, mas altamente fiável. O preço do S3 aproxima-se de um centímo por GB. Esta infraestrutura de baixo custo, mas altamente fiável, é uma das razões pelas quais a computação em Cloud é tão atraente.

Podes ver todo o projeto no **repositório do GitHub**. Observa como o GitHub é a fonte da verdade para o site, porque todo o projeto consiste em arquivos de texto: arquivos markdown, o modelo Hugo e os comandos de compilação para o servidor de compilação AWS Code. Além disso, é possível percorrer um screencast da implantação contínua lá também. **A Figura 7-4** mostra a arquitetura de alto nível deste projeto.

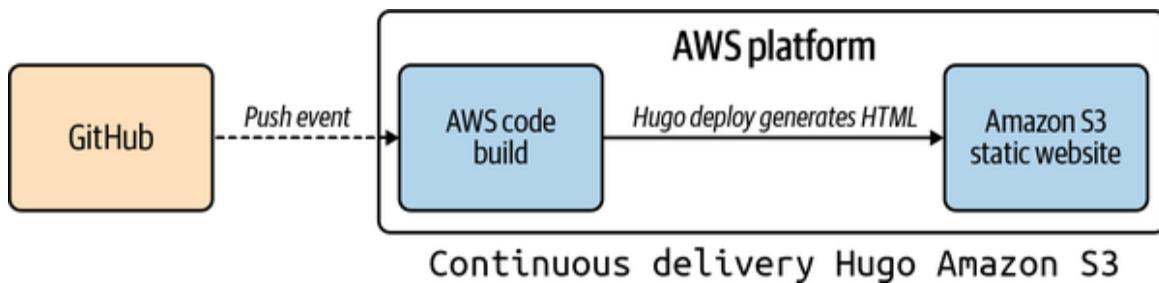


Figura 7-4. Hugo

A versão curta de como este projeto funciona é através da magia do ficheiro `buildspec.yml`. Vamos dar uma olhada em como isso funciona no exemplo a seguir. Primeiro, nota que o binário `hugo` é instalado, depois o comando `hugo` é executado para gerar ficheiros HTML a partir do repositório

verificado. Finalmente, o comando `aws aws s3 sync --delete public s3://dukefeb1` é todo o processo de implantação devido ao poder da hospedagem do bucket S3:

```
version: 0.1

environment_variables:
  plaintext:
    HUGO_VERSION: "0.79.1"

phases:
  install:
    commands:
      - cd /tmp
      - wget https://github.com/gohugoio/hugo/releases/download/v0.80.0/hugo_extended_0.80.0_Linux-64bit.tar.gz
        - tar -xzf hugo_extended_0.80.0_Linux-64bit.tar.gz
        - mv hugo /usr/bin/hugo
        - cd
        - rm -rf /tmp/*
  build:
    commands:
      - rm -rf public
      - hugo
  post_build:
    commands:
      - aws s3 sync --delete public s3://dukefeb1
      - echo Build completed on `date`
```

Outra maneira de descrever um arquivo de sistema de compilação é que ele é uma receita. As informações nos arquivos de configuração de compilação são um "como fazer" para executar as mesmas ações em um ambiente de desenvolvimento do AWS Cloud9.

Conforme discutido no [Capítulo 2](#), o AWS Cloud9 ocupa um lugar especial no meu coração porque resolve um problema específico. Os ambientes de desenvolvimento baseados em Cloud permitem-te desenvolver no local exato onde toda a ação tem lugar. O exemplo mostra como esse conceito époderoso. Verifica o código, testa-o na Cloud e verifica a implementação da mesma ferramenta. Na [Figura 7-5](#), o ambiente AWS Cloud9 invoca um microserviço Python.

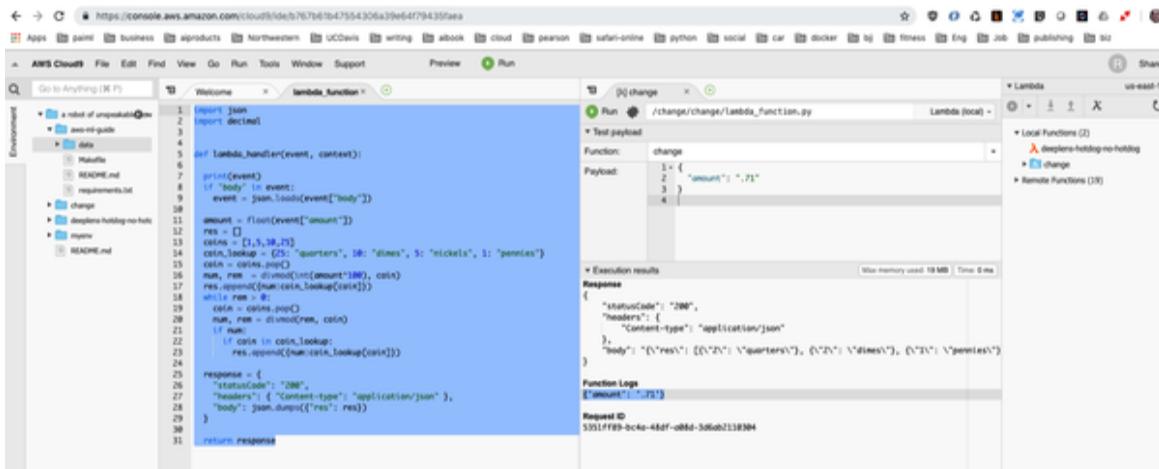


Figura 7-5. Nuvem9

NOTA

Podes ver um passo-a-passo da implementação do Hugo na AWS na [plataforma O'Reilly](#), bem como seguir um guia adicional mais detalhado no [site da Pragmatic AI Labs](#).

Com os fundamentos da entrega contínua para trás, vamos entrar no serverless na plataforma AWS.

Livro de receitas sem servidor

Serverless é uma metodologia crucial para MLOps. No [Capítulo 2](#), mencionei a importância das funções Python. Uma função Python é uma unidade de trabalho que pode receber uma entrada e, opcionalmente, retornar uma saída. Se uma função Python é como uma torradeira, onde colcas um pouco de pão, ela aquece o pão e ejecta a torrada, então o serverless é a fonte de eletricidade.

Uma função Python precisa de ser executada algures, tal como uma torradeira precisa de ser ligada a algo para funcionar. Este conceito é o que o serverless faz; permite que o código seja executado na Cloud. A definição mais genérica de serverless é código que roda sem servidores. Os próprios servidores são abstraídos para permitir que o desenvolvedor se concentre em escrever funções. Essas funções realizam tarefas específicas, e essas

tarefas podem ser encadeadas para construir sistemas mais complexos, como servidores que respondem a eventos.

A função é o centro do universo com a computação em Cloud. Na prática, isso significa que qualquer coisa que seja uma função pode ser mapeada em uma tecnologia que resolve um problema: containers, Kubernetes, GPUs ou AWS Lambda. Como podes ver na [Figura 7-6](#), existe um rico ecossistema de soluções em Python que mapeiam diretamente para uma função.

O serviço de nível mais baixo que executa serverless na plataforma AWS é o AWS Lambda. Vamos dar uma olhada em [alguns exemplos deste repositório](#).

Primeiro, uma das funções Lambda mais simples de escrever na AWS é uma função Marco Polo. Uma função Marco Polo recebe um evento com um nome nele. Então, por exemplo, se o nome do evento for "Marco", retorna "Polo". Se o nome do evento for outra coisa, retorna "Não!"

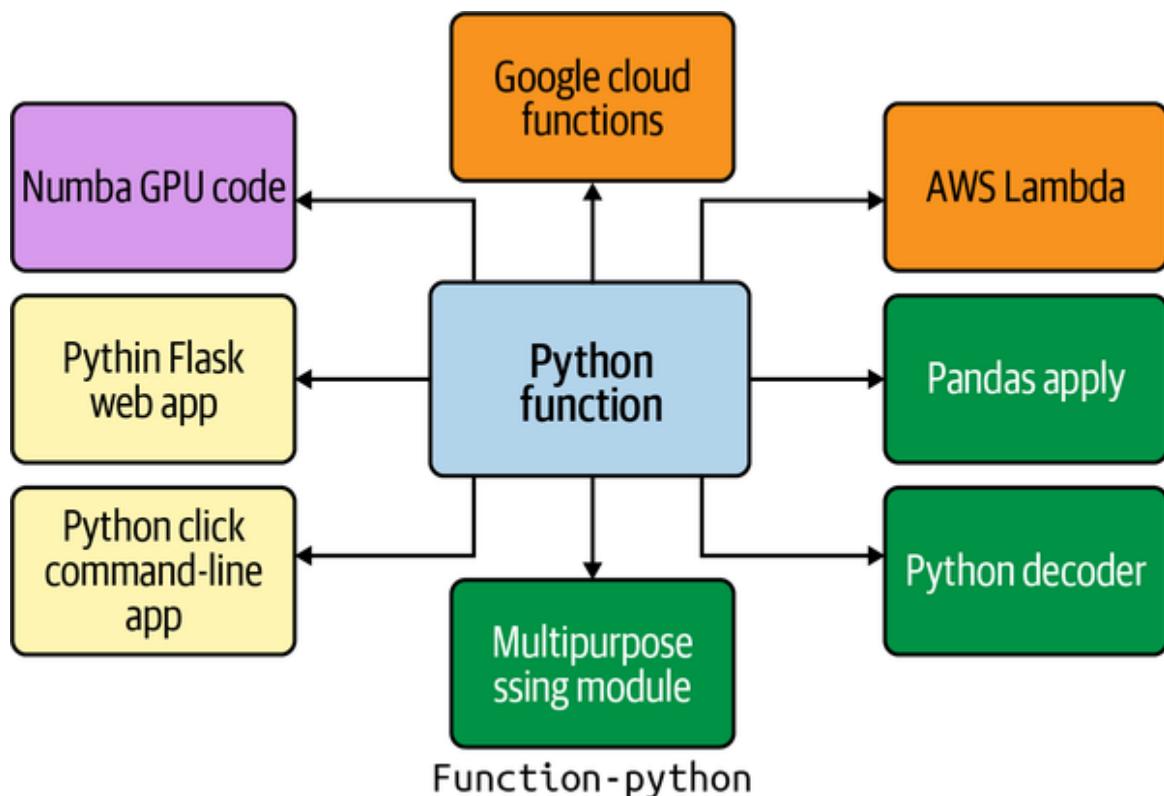


Figura 7-6. Funções Python

NOTA

Quando era adolescente, nos anos 80 e 90, o Marco Polo era um jogo típico para jogar na piscina no verão. Quando trabalhei como conselheiro num acampamento numa piscina perto da minha casa, era um dos jogos preferidos dos miúdos que supervisionava. O jogo funciona da seguinte forma: todos entram na piscina e uma pessoa fecha os olhos e grita "Marco". Depois, os outros jogadores na piscina têm de dizer "Polo". A pessoa com os olhos fechados utiliza o som para localizar alguém para marcar. Quando alguém é apanhado, então é "It".

Aqui está o código Marco Polo do AWS Lambda; repara que um event passa para o lambda_handler:

```
def lambda_handler(event, context):
    print(f"This was the raw event: {event}")
    if event["name"] == "Marco":
        print(f"This event was 'Marco'")
        return "Polo"
    print(f"This event was not 'Marco'")
    return "No!"
```

Com a computação em Cloud sem servidor, pensa numa lâmpada na tua garagem. Uma lâmpada pode ser ligada de várias maneiras, como manualmente através do interruptor de luz ou automaticamente através do evento de abertura da porta da garagem. Da mesma forma, um AWS Lambda também responde a muitos sinais.

Vamos enumerar as formas como as lâmpadas e os lambdas podem ser activados:

- Lâmpadas
 - Liga manualmente o interruptor
 - Através do comando de abertura da porta da garagem
 - Temporizador de segurança noturno que acende a luz da meia-noite às 6 da manhã.
- AWS Lambda

- Invoca manualmente através da consola, da linha de comandos da AWS ou do AWS Boto3 SDK
 - Responde a eventos S3, como o carregamento de um ficheiro para um contentor
 - Invoca o temporizador todas as noites para descarregar dados

Que tal um exemplo mais complexo? Com o AWS Lambda, é simples integrar um S3 Trigger com rotulagem de visão computacional em todas as novas imagens colocadas numa pasta, com uma quantidade trivial de código:

Finalmente, podes encadear várias funções AWS Lambda através do AWS StepFunctions:

```
{  
    "Comment": "This is Marco Polo",  
    "StartAt": "Marco",  
    "States": {  
        "Marco": {  
            "Type": "Task",  
            "Resource": "arn:aws:lambda:us-east-  
1:561744971673:function:marco20",  
            "Next": "Polo"  
        },  
        "Polo": {  
            "Type": "Task",  
            "Resource": "arn:aws:lambda:us-east-  
1:561744971673:function:polo",  
            "Next": "Finish"  
        },  
        "Finish": {  
            "Type": "Pass",  
            "Result": "Finished",  
            "End": true  
        }  
    }  
}
```

Podes ver este fluxo de trabalho em ação na [Figura 7-7](#).

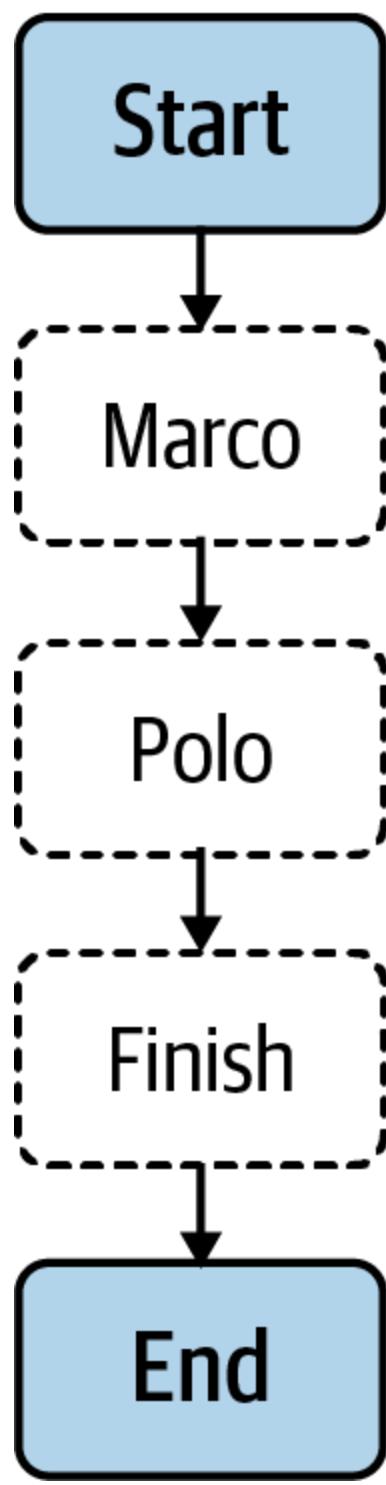


Figura 7-7. Funções de passo

Ainda mais divertido é o facto de poderes chamar as funções AWS Lambda através de um CLI. Aqui tens um exemplo:

```
aws lambda invoke \
    --cli-binary-format raw-in-base64-out \
    --function-name marcopython \
    --payload '{"name": "Marco"}' \
    response.json
```

NOTA

É importante consultar sempre a documentação mais recente da AWS para a CLI, pois ela é um alvo em constante movimento. No momento em que este livro foi escrito, a versão atual da CLI era a V2, mas talvez seja necessário ajustar os exemplos de linha de comando à medida que as coisas mudarem no futuro. Podes encontrar a documentação mais recente no [site AWS CLI Command Reference](#).

A resposta da carga útil é a seguinte:

```
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
(.venv) [cloudshell-user@ip-10-1-14-160 ~]$ cat response.json
"Polo"(.venv) [cloudshell-user@ip-10-1-14-160 ~]$
```

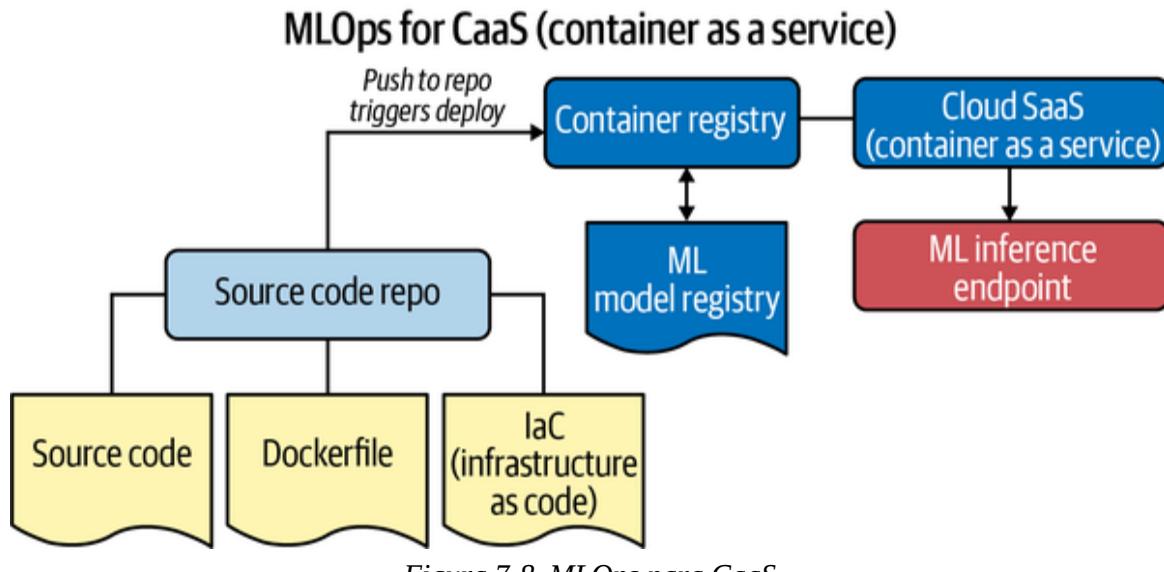
NOTA

Para um passo a passo mais avançado do AWS Lambda utilizando o Cloud9 e o AWS SAM (Serverless Application Model), podes ver um passo a passo de um pequeno microsserviço da Wikipédia no [canal do YouTube da Pragmatic AI Labs](#) ou na [plataforma de aprendizagem da O'Reilly](#).

O AWS Lambda é talvez o tipo de computação mais valioso e flexível que podes utilizar para fornecer previsões para pipelines de aprendizagem automática ou para gerir eventos ao serviço de um processo MLOps. A razão para isso é a velocidade de desenvolvimento e teste. Em seguida, vamos falar sobre algumas ofertas de CaaS, ou contentor como um serviço.

AWS CaaS

Fargate é um contêiner como uma oferta de serviço da AWS que permite que os desenvolvedores se concentrem na criação de microsserviços em contêineres. Por exemplo, na [Figura 7-8](#), quando esse microsserviço funciona no contêiner, todo o tempo de execução, incluindo os pacotes necessários para a implantação, funcionará em um novo ambiente. A plataforma Cloud lida com o resto da implantação.



NOTA

Os contentores resolvem muitos problemas que têm atormentado a indústria do software. Portanto, como regra geral, é uma boa ideia utilizá-los em projectos MLOps. Segue-se uma lista parcial das vantagens dos contentores nos projectos:

- Permite que o programador imite o serviço de produção localmente no seu ambiente de trabalho
- Permite a distribuição fácil do tempo de execução do software aos clientes através de registos de contentores públicos como o Docker Hub, o GitHub Container Registry e o Amazon Elastic Container Registry
- Permite que o GitHub ou um repositório de código-fonte seja a "fonte da verdade" e contenha todos os aspectos do microsserviço: modelo, código, IaC e tempo de execução
- Permite uma fácil implementação na produção através de serviços CaaS

Vamos ver como podes construir um microsserviço que devolve a alteração correta usando o Flask. A Figura 7-9 mostra um fluxo de trabalho de desenvolvimento no AWS Cloud9. O Cloud9 é o ambiente de desenvolvimento; um contêiner é criado e enviado para o ECR. Mais tarde, esse contêiner é executado no ECS.

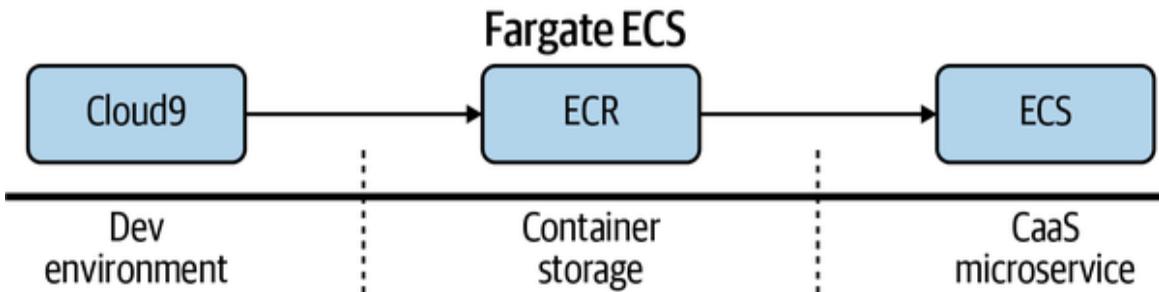


Figura 7-9. Fluxo de trabalho do ECS

Segue-se o código Python para *app.py*:

```

from flask import Flask
from flask import jsonify
app = Flask(__name__)

def change(amount):
    # calculate the resultant change and store the result (res)
    res = []
    coins = [1,5,10,25] # value of pennies, nickels, dimes,
quarters
    coin_lookup = {25: "quarters", 10: "dimes", 5: "nickels", 1:
"pennies"}

    # divide the amount*100 (the amount in cents) by a coin value
    # record the number of coins that evenly divide and the
remainder
    coin = coins.pop()
    num, rem = divmod(int(amount*100), coin)
    # append the coin type and number of coins that had no
remainder
    res.append({num:coin_lookup[coin]})

    # while there is still some remainder, continue adding coins
    # to the result
    while rem > 0:
        coin = coins.pop()
        num, rem = divmod(rem, coin)

```

```

    if num:
        if coin in coin_lookup:
            res.append({num:coin_lookup[coin]})

return res

@app.route('/')
def hello():
    """Return a friendly HTTP greeting."""
    print("I am inside hello world")
    return 'Hello World! I can make change at route: /change'

@app.route('/change/<dollar>/<cents>')
def changeroute(dollar, cents):
    print(f"Make Change for {dollar}.{cents}")
    amount = f"{dollar}.{cents}"
    result = change(float(amount))
    return jsonify(result)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)

```

Repara que o microsserviço Web Flask responde a pedidos de alteração através de pedidos Web para o padrão URL `/change/<dollar>/<cents>`. Podes ver o código-fonte completo para este exemplo Fargate no seguinte repositório GitHub. As etapas são as seguintes:

1. Configura a aplicação: virtualenv + `make all`
2. Testa a aplicação local: `python app.py`
3. Enrola-o para testar: `curl localhost:8080/change/1/34`
4. Cria o ECR (Amazon Container Registry)

Na **Figura 7-10**, um repositório ECR permite a implementação posterior do Fargate.

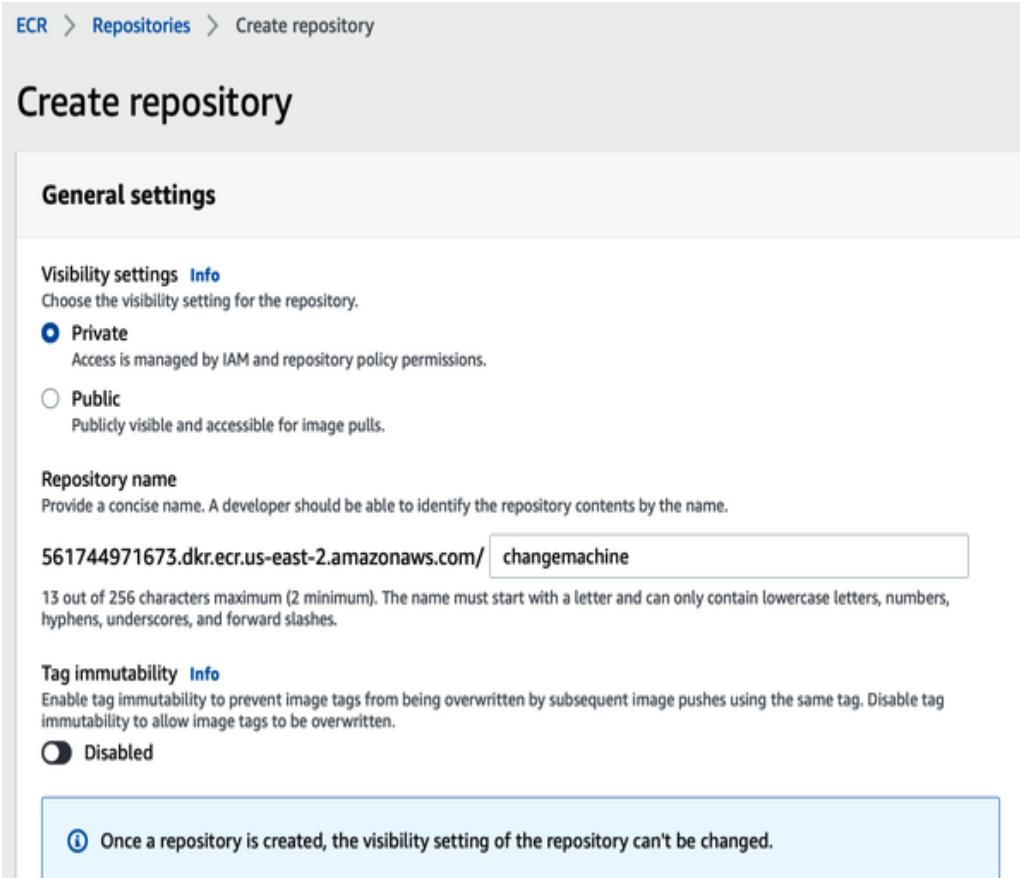


Figura 7-10. ECR

5. Constrói um contentor
6. Empurra o contentor
7. Executa o docker local: `docker run -p 8080:8080 changemachine`
8. Desloca-te para Fargate
9. Testa o serviço público

NOTA

Qualquer serviço Cloud terá mudanças rápidas e constantes na funcionalidade, por isso é melhor ler a documentação atual. A [documentação atual do Fargate](#) é um excelente local para ler mais sobre as formas mais recentes de implementar o serviço.

Também podes, opcionalmente, assistir a uma apresentação completa de uma implementação Fargate na [plataforma O'Reilly](#).

Outra opção para CaaS é o AWS App Runner, que simplifica ainda mais as coisas. Por exemplo, podes implementar diretamente a partir do código fonte ou apontar para um contentor. Na [Figura 7-11](#), o AWS App Runner cria um fluxo de trabalho simplificado que conecta um repositório de origem, um ambiente de implantação e um URL seguro resultante.

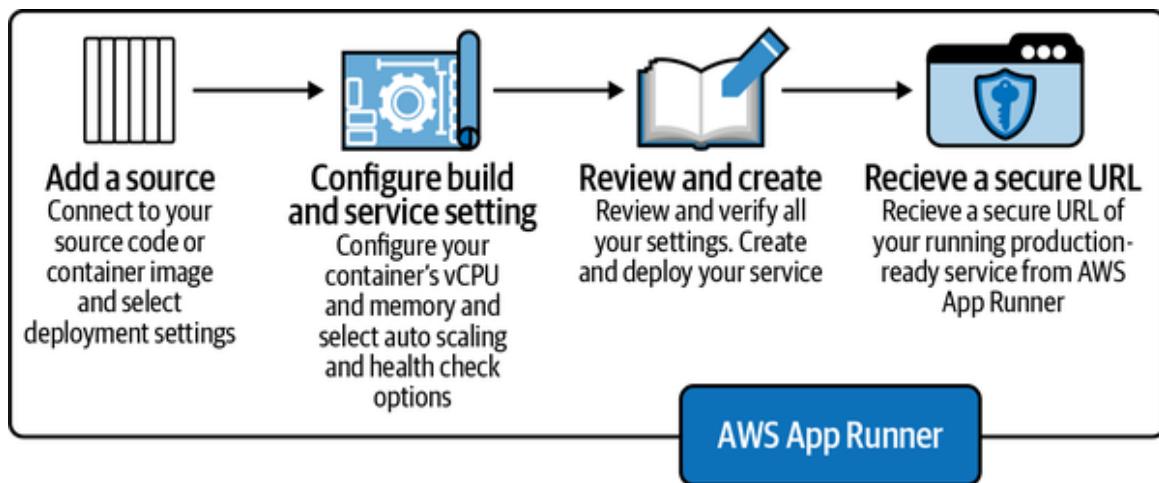


Figura 7-11. AWS App Runner

Este repositório pode ser facilmente convertido para um método AWS App Runner no assistente AWS com os seguintes passos:

1. Para construir o projeto, utiliza o comando: `pip install -r requirements.txt`.
2. Para executar o projeto, utiliza: `python app.py`.
3. Finalmente, configura a porta a utilizar: `8080`.

Uma inovação importante, mostrada na [Figura 7-12](#), é a capacidade de conectar muitos serviços diferentes no AWS, ou seja, a infraestrutura principal, como o AWS CloudWatch,平衡adores de carga, serviços de contêineres e gateways de API em uma oferta completa.

The screenshot shows the AWS App Runner service creation interface. At the top, a green banner displays the message "Create service succeeded." Below this, the service details are listed:

- Status: Running
- Default domain: <https://wkgnu5im6u.us-east-1.awssapprunner.com>
- Service ARN: arn:aws:apprunner:us-east-1:561744971673:service/CaaS2/7d89107ab3b64568931459a27b34ea21
- Source: <https://github.com/noahgift/eks-fargate-tutorial/main>

Below the details, there are tabs for Activity, Logs (which is selected), Configuration, Metrics, and Custom domains. The Logs section contains an Event log tab with a list of log entries:

```
1 05-23-2021 08:55 AM [AppRunner] Service status is set to RUNNING.  
2 05-23-2021 08:55 AM [AppRunner] Service creation completed successfully.  
3 05-23-2021 08:55 AM [AppRunner] Successfully routed incoming traffic to application.  
4 05-23-2021 08:55 AM [AppRunner] Health check is successful. Routing traffic to application.  
5 05-23-2021 08:54 AM [AppRunner] Performing health check on path '/' and port '8080'.  
6 05-23-2021 08:54 AM [AppRunner] Provisioning instance and deploying image.  
7 05-23-2021 08:54 AM [AppRunner] Successfully built source code.  
8 05-23-2021 08:52 AM [AppRunner] Starting source code build.  
9 05-23-2021 08:52 AM [AppRunner] Successfully pulled source code.
```

Figura 7-12. Cria o serviço AWS App Runner

O serviço final implantado na [Figura 7-13](#) mostra um URL seguro e a capacidade de invocar o ponto de extremidade e retornar a alteração correta.

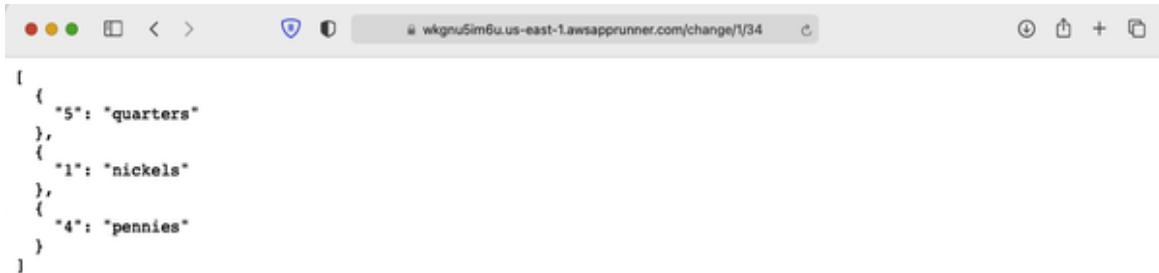


Figura 7-13. AWS App Runner implantado

Porque é que esta magia é útil? Em poucas palavras, toda a lógica, incluindo talvez um modelo de aprendizado de máquina, está em um único repositório. Portanto, este é um estilo atraente para a construção de produtos compatíveis com MLOps. Além disso, um dos aspectos mais complexos da entrega de um aplicativo de aprendizado de máquina para produção é a implantação de microsserviços. O AWS App Runner faz com que a maior parte da complexidade desapareça, capturando tempo para outras partes do problema de MLOps. Em seguida, vamos discutir como a AWS lida com a visão computacional.

Visão por computador

Dou um curso de visão computacional aplicada no [programa de pós-graduação em ciência de dados da Northwestern](#). Esta aula é muito agradável de lecionar porque fazemos o seguinte:

- Demonstrações de vídeo semanais
- Concentra-te na resolução de problemas e não na codificação ou modelação
- Usa ferramentas de alto nível como o AWS DeepLens, uma câmara de vídeo com aprendizagem profunda

Na prática, isto permite um ciclo de feedback rápido que se concentra no problema versus a tecnologia para resolver o problema. Uma das tecnologias em uso é o dispositivo AWS Deep Lens, como mostrado na [Figura 7-14](#). O dispositivo é um kit completo de desenvolvimento de hardware de visão computacional, na medida em que contém uma câmara de 1080p, sistema operativo e capacidades sem fios. Em particular, isto resolve o problema de prototipagem da visão por computador.

Tech Specs

CPU	OS	GRAPHICS
Intel Atom® Processor	Ubuntu OS-16.04 LTS	Intel Gen9 Graphics Engine
MEMORY	BUILT-IN STORAGE	
8GB RAM	16GB Memory (expandable)	

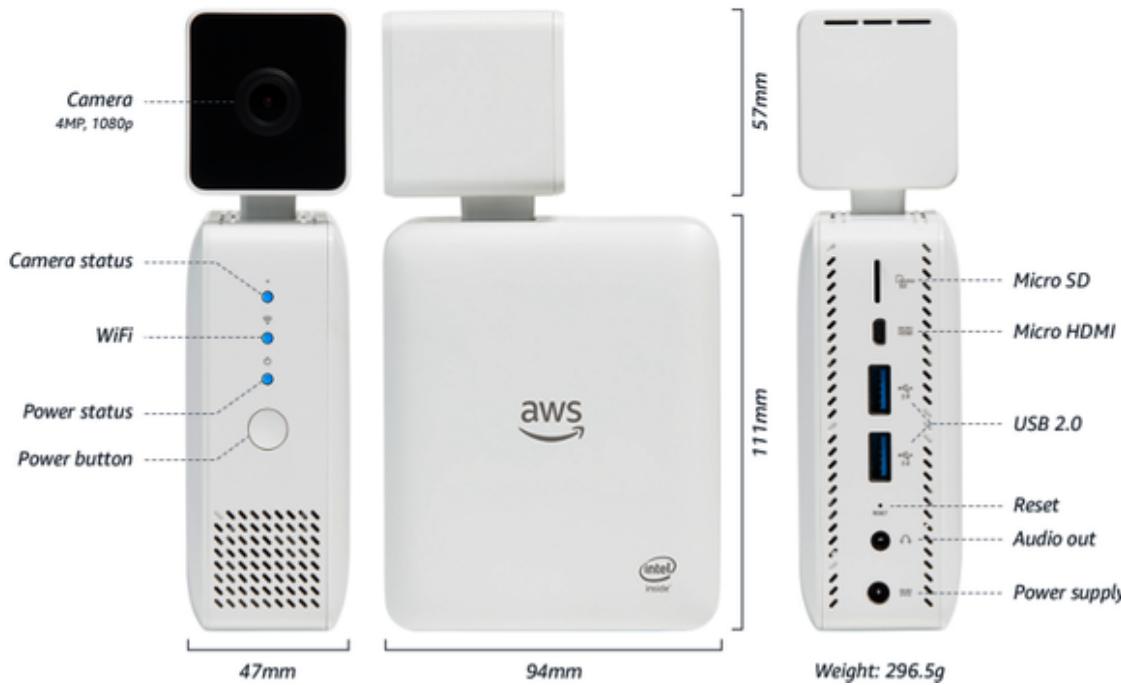


Figura 7-14. DeepLens

Quando o AWS DeepLens começa a capturar vídeo, divide o vídeo em dois fluxos. O fluxo do projeto mostrado na [Figura 7-15](#) adiciona anotações em tempo real e envia os pacotes para o serviço MQ Telemetry Transport (MQTT), que é um protocolo de rede de publicação e assinatura.

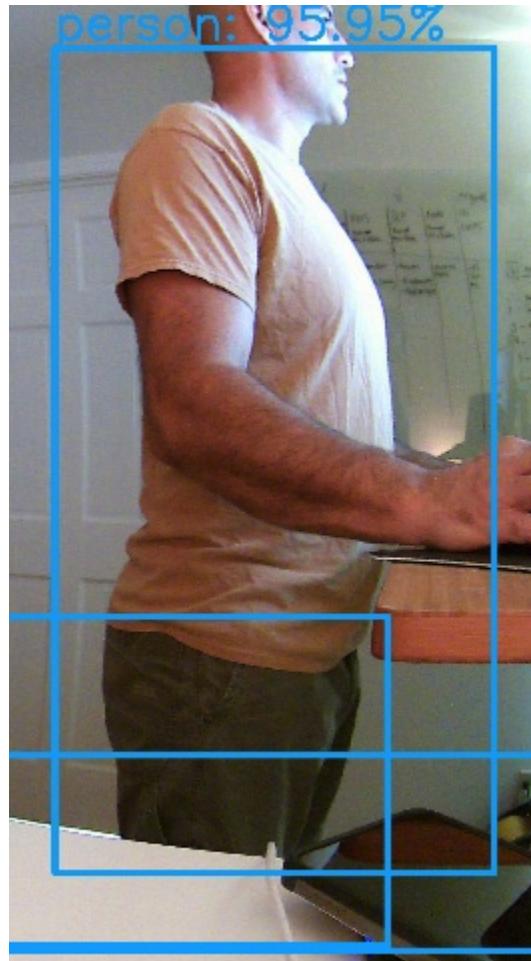


Figura 7-15. Detecta

Na [Figura 7-16](#), os pacotes MQTT chegam em tempo real à medida que os objectos são detectados no fluxo.

O DeepLens é uma tecnologia "plug and play", pois aborda talvez a parte mais desafiadora do problema de criar um sistema de prototipagem de visão computacional em tempo real - capturar os dados e enviá-los para algum lugar. O fator "divertido" disso é como é fácil ir de zero a um criando soluções com o AWS DeepLens. Em seguida, vamos ser mais específicos e ir além da construção de microsserviços e passar a construir microsserviços que implantam código de aprendizado de máquina.

The screenshot shows the MQTT client interface with the following details:

- Subscriptions:** \$aws/things/deeplens... (highlighted in red)
- Publish:** Publish to topic \$aws/things/deeplens... (highlighted in red)
- Message Content:**

```

1 [
2   "message": "Hello from AWS IoT console"
3 ]

```
- Published Messages:**
 - \$aws/things/deeplens... Sep 10, 2018 8:33:58 AM -0700 (highlighted in red)

```
{
  "chair": 0.311279296875,
  "dinning table": 0.3017578125,
  "bottle": 0.449951171875
}
```
 - \$aws/things/deeplens... Sep 10, 2018 8:33:57 AM -0700 (highlighted in red)

```
{
  "chair": 0.31787109375,
  "dinning table": 0.3330078125,
  "bottle": 0.50634765625
}
```

Figura 7-16. MQTT

MLOps no AWS

Uma forma de começar a utilizar os MLOps na AWS é considerar a seguinte questão. Quando te é dado um problema de aprendizagem automática com três restrições - precisão da previsão, explicabilidade e operações -, em que duas te concentras para alcançar o sucesso e por que ordem?

Muitos cientistas de dados com foco académico saltam imediatamente para a precisão das previsões. Construir modelos de previsão cada vez melhores é um desafio divertido, como jogar Tetris. Além disso, a modelação é glamorosa e um aspeto cobiçado do trabalho. Os cientistas de dados gostam de mostrar a precisão com que conseguem treinar um modelo académico utilizando técnicas cada vez mais sofisticadas.

Toda a plataforma Kaggle trabalha para aumentar a precisão da previsão, e há recompensas monetárias para o modelo preciso. Uma abordagem diferente é concentrar-se na operacionalização do modelo. A vantagem desta abordagem é que, mais tarde, a precisão do modelo pode melhorar juntamente com as melhorias no sistema de software. Tal como a indústria automóvel japonesa se concentrou no Kaizen ou na melhoria contínua, um sistema de ML pode concentrar-se numa precisão de previsão inicial razoável e melhorar rapidamente.

A cultura da AWS apoia este conceito nos princípios de liderança de "Preconceito para a Ação" e "Entregar Resultados". O "Bias for Action" refere-se à velocidade e à entrega de resultados, concentrando-se nas entradas críticas para um negócio e entregando resultados rapidamente. Como resultado, os produtos da AWS em torno da aprendizagem automática, como o AWS SageMaker, mostram o espírito desta cultura de ação e resultados.

A entrega contínua (CD) é um componente essencial do MLOps. Antes de poderes automatizar a entrega para a aprendizagem automática, o próprio microsserviço precisa de ser automatizado. As especificidades mudam dependendo do tipo de serviço da AWS envolvido. Vamos começar com um exemplo de ponta a ponta.

No exemplo a seguir, um aplicativo Flask do Elastic Beanstalk é implantado continuamente usando toda a tecnologia AWS, desde o AWS Code Build até o AWS Elastic Beanstalk. Esta "pilha" também é ideal para implantar modelos de ML. O Elastic Beanstalk é uma tecnologia de plataforma como serviço oferecida pela AWS que simplifica grande parte do trabalho de implementação de uma aplicação.

Na [Figura 7-17](#), observa que o AWS Cloud9 é um ponto de partida recomendado para o desenvolvimento. Em seguida, um repositório do GitHub mantém o código-fonte do projeto e, à medida que ocorrem eventos de alteração, aciona o servidor de compilação nativo da Cloud, o AWS CodeBuild. Finalmente, o processo do AWS Code Build executa a

integração contínua, testa o código e fornece entrega contínua ao AWS Elastic Beanstalk.

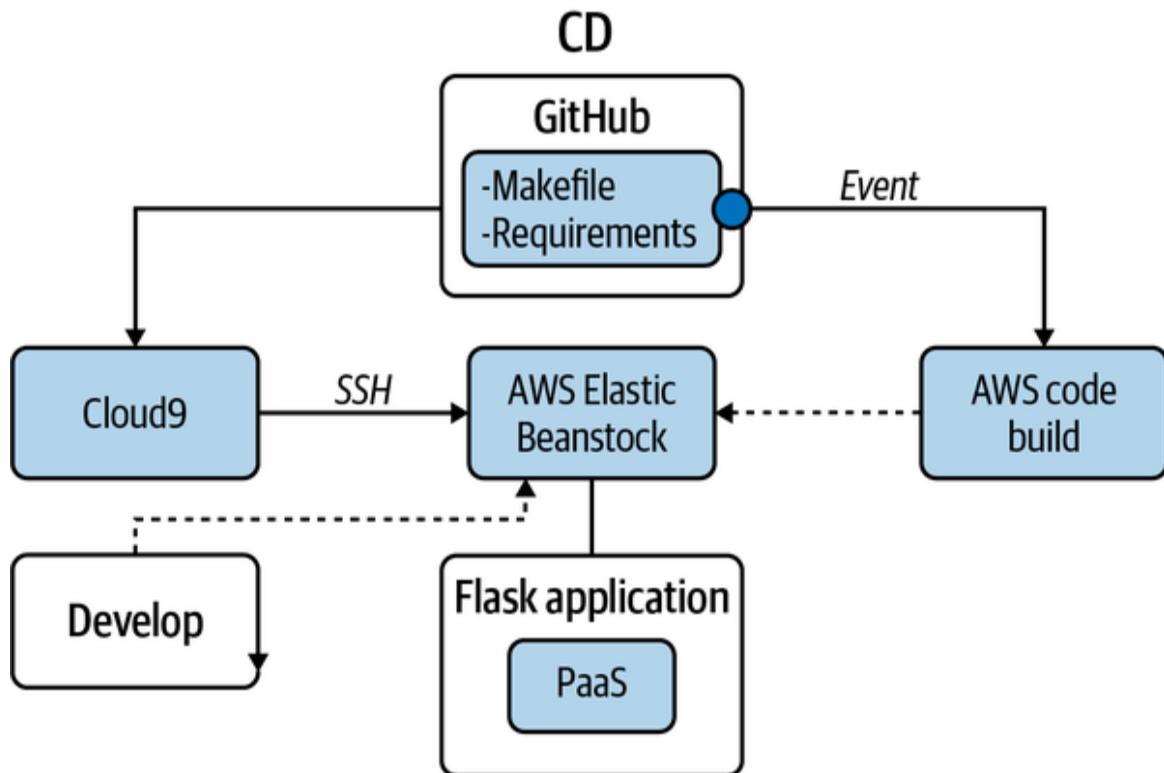


Figura 7-17. Elastic Beanstalk

NOTA

O código fonte e um passo-a-passo deste exemplo estão nas seguintes ligações:

- [Código fonte](#)
- [Vídeo de apresentação da plataforma O'Reilly](#)

Para replicar este projeto exato, podes seguir os seguintes passos:

1. Verifica o repositório no AWS Cloud9 ou no AWS CloudShell se tiveres conhecimentos sólidos de linha de comandos.
2. Cria um virtualenv Python, faz o source e corre `make all`:

```
python3 -m venv ~/.eb
source ~/.eb/bin/activate
make all
```

Observa que `awsebcli` instala através de requisitos, e esta ferramenta controla o Elastic Beanstalk a partir do CLI.

3. Inicializa a nova aplicação eb:

```
eb init -p python-3.7 flask-continuous-delivery --region
us-east-1
```

Opcionalmente, usa `eb init` para criar chaves SSH para entrar nas instâncias em execução.

4. Cria uma instância eb remota:

```
eb create flask-continuous-delivery-env
```

5. Configura o projeto de compilação de código da AWS. Observe que seu Makefile precisa refletir os nomes do seu projeto:

```
version: 0.2

phases:
  install:
    runtime-versions:
      python: 3.7
  pre_build:
    commands:
      - python3.7 -m venv ~/.venv
      - source ~/.venv/bin/activate
      - make install
      - make lint
```

```
build:  
  commands:  
    - make deploy
```

Depois de teres o projeto a funcionar com a implementação contínua, estás pronto para passar ao passo seguinte, implementar um modelo de ML. Recomendo vivamente que faças funcionar um projeto do tipo "hello world" com a implementação contínua, como este, antes de avançares diretamente para um projeto de ML complexo quando estiveres a aprender uma nova tecnologia . Em seguida, vamos analisar um livro de receitas de MLOps intencionalmente simples que é a base para muitas novas implantações de serviços da AWS.

Livro de receitas do MLOps no AWS

Com os componentes fundamentais fora do caminho, vamos analisar uma receita básica de aprendizagem automática e aplicá-la a vários cenários. Repara que esta receita básica é implementável em muitos serviços na AWS e em muitos outros ambientes Cloud. Este projeto do Livro de Receitas do MLOps a seguir é intencionalmente espartano, portanto, o foco está na implantação do aprendizado de máquina. Por exemplo, este projeto prevê a altura a partir de uma entrada de peso para os jogadores da Major League Baseball.

Na [Figura 7-18](#), GitHub é a fonte da verdade e contém o andaime do projeto. Em seguida, o serviço de construção é o GitHub Actions, e o serviço de contêiner é o GitHub Container Registry. Ambos os serviços podem substituir facilmente qualquer oferta semelhante na Cloud. Em particular, na Cloud da AWS, podes utilizar o AWS CodeBuild para CI/CD e o AWS ECR (Elastic Container Registry). Finalmente, uma vez que um projeto tenha sido "contentorizado", abre o projeto para muitos alvos de implantação. Na AWS, isso inclui o AWS Lambda, o AWS Elastic Beanstalk e o AWS App Runner.

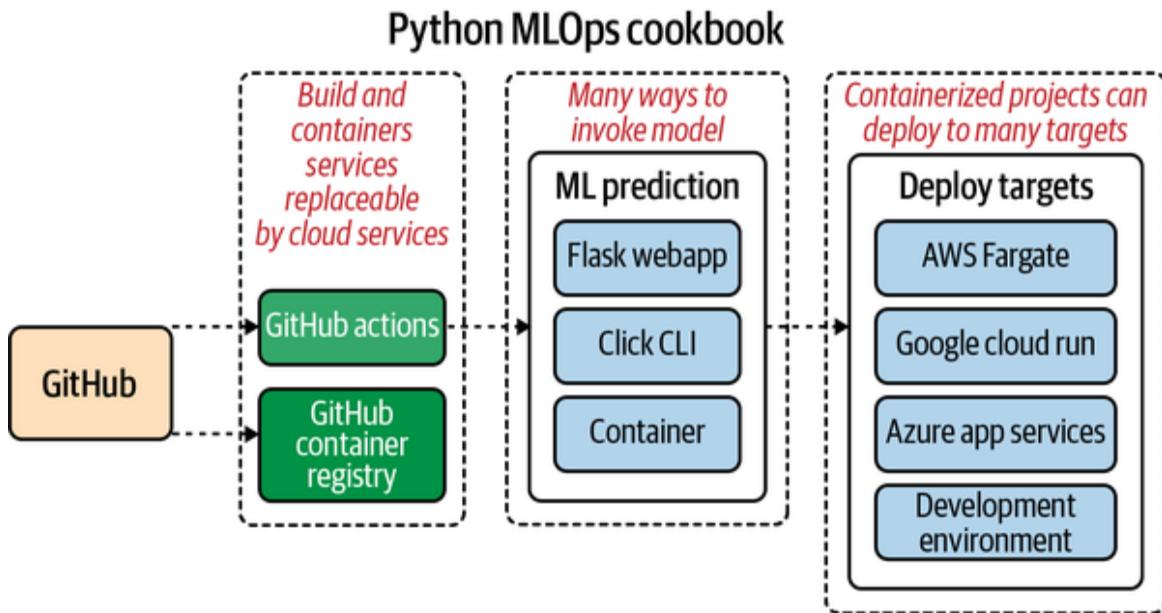


Figura 7-18. Livro de receitas do MLOps

Os ficheiros seguintes são todos úteis para construir soluções em muitas receitas diferentes:

Makefile

O Makefile é tanto uma lista de receitas quanto uma maneira de invocar essas receitas. Vê o [Makefile no projeto de exemplo do GitHub](#).

requisitos.txt

O ficheiro de requisitos contém a lista de pacotes Python para o projeto. Normalmente, esses pacotes são "fixados" a um número de versão, o que limita dependências inesperadas de pacotes. Vê o [requirements.txt no projeto de exemplo do GitHub](#).

cli.py

Esta linha de comando mostra como uma biblioteca ML também pode ser invocada a partir da CLI, não apenas através de uma aplicação web. Vê o [cli.py no projeto de exemplo do GitHub](#).

utilscli.py

O utilscli.py é um utilitário que permite ao utilizador invocar diferentes pontos finais, ou seja, AWS, GCP, Azure ou qualquer ambiente de produção. A maioria dos algoritmos de aprendizagem automática requer que os dados sejam escalados. Esta ferramenta simplifica o escalonamento da entrada e o escalonamento de volta da saída. Vê o utilscli.py no projeto de exemplo do GitHub.

app.py

O arquivo de aplicativo é o microsserviço da Web do Flask que aceita e retorna um resultado de previsão JSON por meio do ponto de extremidade de URL /predict. Visualiza o app.py no projeto de exemplo do GitHub.

mlib.py

A biblioteca de manipulação de modelos faz a maior parte do trabalho pesado em um local centralizado. Essa biblioteca é intencionalmente muito básica e não resolve problemas mais complicados, como o carregamento em cache do modelo ou outros problemas de produção exclusivos da implantação de produção. Visualiza o mlib.py no projeto de exemplo do GitHub.

htwtmlb.csv

Um arquivo CSV é útil para o dimensionamento de entrada. Visualiza o htwtmlb.csv no projeto de exemplo do GitHub.

modelo.joblib

Esse modelo é exportado do sklearn, mas poderia facilmente estar em outro formato, como ONNX ou TensorFlow. Outras considerações de produção do mundo real podem ser manter esse modelo em um local diferente, como o Amazon S3, em um contêiner ou até mesmo hospedado pelo AWS SageMaker. Visualiza o modelo.joblib no projeto de exemplo do GitHub.

Dockerfile

Esse arquivo permite a conteinerização do projeto e, como resultado, abre muitas novas opções de implantação, tanto na plataforma AWS quanto em outras Clouds. Visualiza o [Dockerfile no projeto de exemplo do GitHub](#).

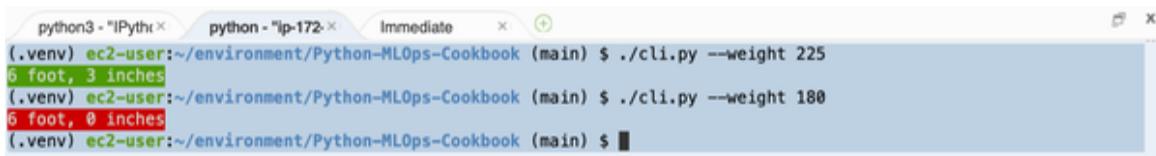
Baseball_Predictions_Export_Model.ipynb

O bloco de notas Jupyter é um artefacto crucial a incluir num projeto de aprendizagem automática. Mostra a outro desenvolvedor o pensamento por trás da criação do modelo e fornece um contexto valioso para manter o projeto em produção. Vê o [Baseball_Predictions_Export_Model.ipynb no projeto de exemplo do GitHub](#).

Estes artefactos de projeto são úteis como ferramenta educacional para explicar os MLOps, mas podem ser diferentes ou mais complexos num cenário de produção único. Em seguida, vamos discutir como as ferramentas CLI (interface de linha de comando) ajudam a operacionalizar um projeto de aprendizagem automática.

Ferramentas CLI

Neste projeto, existem duas ferramentas CLI. Primeiro, o *cli.py* principal é o ponto de extremidade que fornece as previsões. Por exemplo, para prever a altura de um jogador da MLB, usa o seguinte comando para criar uma previsão: `./cli.py --weight 180`. Observe na [Figura 7-19](#) que a opção de linha de comando `--weight` permite que o usuário teste rapidamente muitas novas entradas de previsão.

A screenshot of a terminal window titled "python3 - IPython" and "python - ip-172". It shows three command-line entries: ".venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)\$./cli.py --weight 225", which outputs "6 foot, 3 inches"; ".venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)\$./cli.py --weight 180", which outputs "6 foot, 0 inches"; and ".venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)\$".

```
python3 - IPython python - ip-172 Immediate 
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)$ ./cli.py --weight 225
6 foot, 3 inches
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)$ ./cli.py --weight 180
6 foot, 0 inches
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main)$
```

Figura 7-19. Previsão da CLI

Então, como é que isto funciona? A maior parte da "magia" é feita através de uma biblioteca que faz o trabalho pesado de escalar os dados, fazer a previsão e depois fazer uma transformação inversa:

```
"""MLOps Library"""

import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge
import joblib
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import logging

logging.basicConfig(level=logging.INFO)

import warnings

warnings.filterwarnings("ignore", category=UserWarning)

def load_model(model="model.joblib"):
    """Grabs model from disk"""

    clf = joblib.load(model)
    return clf

def data():
    df = pd.read_csv("htwtmlb.csv")
    return df

def retrain(tsize=0.1, model_name="model.joblib"):
    """Retrains the model

    See this notebook: Baseball_Predictions_Export_Model.ipynb
    """

    df = data()
    y = df["Height"].values # Target
    y = y.reshape(-1, 1)
    X = df["Weight"].values # Feature(s)
    X = X.reshape(-1, 1)
    scaler = StandardScaler()
    X_scaler = scaler.fit(X)
    X = X_scaler.transform(X)
```

```

y_scaler = scaler.fit(y)
y = y_scaler.transform(y)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=tsize, random_state=3
)
clf = Ridge()
model = clf.fit(X_train, y_train)
accuracy = model.score(X_test, y_test)
logging.debug(f"Model Accuracy: {accuracy}")
joblib.dump(model, model_name)
return accuracy, model_name

def format_input(x):
    """Takes int and converts to numpy array"""

    val = np.array(x)
    feature = val.reshape(-1, 1)
    return feature

def scale_input(val):
    """Scales input to training feature values"""

    df = data()
    features = df["Weight"].values
    features = features.reshape(-1, 1)
    input_scaler = StandardScaler().fit(features)
    scaled_input = input_scaler.transform(val)
    return scaled_input

def scale_target(target):
    """Scales Target 'y' Value"""

    df = data()
    y = df["Height"].values # Target
    y = y.reshape(-1, 1) # Reshape
    scaler = StandardScaler()
    y_scaler = scaler.fit(y)
    scaled_target = y_scaler.inverse_transform(target)
    return scaled_target

def height_human(float_inches):
    """Takes float inches and converts to human height in
    ft/inches"""

```

```

feet = int(round(float_inches / 12, 2)) # round down
inches_left = round(float_inches - feet * 12)
result = f"{feet} foot, {inches_left} inches"
return result

def human_readable_payload(predict_value):
    """Takes numpy array and returns back human readable
    dictionary"""
    height_inches = float(np.round(predict_value, 2))
    result = {
        "height_inches": height_inches,
        "height_human_readable": height_human(height_inches),
    }
    return result

def predict(weight):
    """Takes weight and predicts height"""

    clf = load_model() # loadmodel
    np_array_weight = format_input(weight)
    scaled_input_result = scale_input(np_array_weight)
    scaled_height_prediction = clf.predict(scaled_input_result)
    height_predict = scale_target(scaled_height_prediction)
    payload = human_readable_payload(height_predict)
    predict_log_data = {
        "weight": weight,
        "scaled_input_result": scaled_input_result,
        "scaled_height_prediction": scaled_height_prediction,
        "height_predict": height_predict,
        "human_readable_payload": payload,
    }
    logging.debug(f"Prediction: {predict_log_data}")
    return payload

```

Em seguida, a estrutura Click envolve as chamadas da biblioteca para *mlib.py* e cria uma interface limpa para servir as previsões. Há muitas vantagens em usar ferramentas de linha de comando como interface principal para interagir com modelos de aprendizado de máquina. A velocidade de desenvolvimento e implantação de uma ferramenta de aprendizado de máquina de linha de comando é talvez a mais importante:

```

#!/usr/bin/env python
import click
from mlib import predict

@click.command()
@click.option(
    "--weight",
    prompt="MLB Player Weight",
    help="Pass in the weight of a MLB player to predict the height",
)
def predictcli(weight):
    """Predicts Height of an MLB player based on weight"""

    result = predict(weight)
    inches = result["height_inches"]
    human_readable = result["height_human_readable"]
    if int(inches) > 72:
        click.echo(click.style(human_readable, bg="green",
                               fg="white"))
    else:
        click.echo(click.style(human_readable, bg="red",
                               fg="white"))

if __name__ == "__main__":
    # pylint: disable=no-value-for-parameter
    predictcli()

```

A segunda ferramenta CLI é *utilscli.py*, que executa o retreinamento do modelo e pode servir como ponto de entrada para fazer mais tarefas. Por exemplo, esta versão não altera a predefinição `model_name`, mas podes adicioná-la como uma opção ao [bifurcar este repositório](#):

```
./utilscli.py retrain --tsize 0.4
```

Repara que o *mlib.py* faz novamente o trabalho pesado, mas o CLI fornece uma forma conveniente de fazer uma prototipagem rápida de um modelo ML:

```

#!/usr/bin/env python
import click

```

```

import mlib
import requests

@click.group()
@click.version_option("1.0")
def cli():
    """Machine Learning Utility Belt"""

@cli.command("retrain")
@click.option("--tsize", default=0.1, help="Test Size")
def retrain(tsize):
    """Retrain Model
    You may want to extend this with more options, such as
    setting model_name
    """
    click.echo(click.style("Retraining Model", bg="green",
                           fg="white"))
    accuracy, model_name = mlib.retrain(tsize=tsize)
    click.echo(
        click.style(f"Retrained Model Accuracy: {accuracy}",
                   bg="blue",
                   fg="white"))
    click.echo(click.style(f"Retrained Model Name: {model_name}",
                           bg="red",
                           fg="white"))

@cli.command("predict")
@click.option("--weight", default=225, help="Weight to Pass In")
@click.option("--host", default="http://localhost:8080/predict",
              help="Host to query")
def mkrequest(weight, host):
    """Sends prediction to ML Endpoint"""

    click.echo(click.style(f"Querying host {host} with weight:
{weight}",
                           bg="green", fg="white"))
    payload = {"Weight": weight}
    result = requests.post(url=host, json=payload)
    click.echo(click.style(f"result: {result.text}", bg="red",
                           fg="white"))

if __name__ == "__main__":
    cli()

```

A Figura 7-20 é um exemplo de reciclagem do modelo.

```
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py retrain --tsize 0.1
Retraining Model
Retrained Model Accuracy: 0.18137638458541205
Retrained Model Name: model.joblib
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py retrain --tsize 0.2
Retraining Model
Retrained Model Accuracy: 0.2802199932746626
Retrained Model Name: model.joblib
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py retrain --tsize 0.3
Retraining Model
Retrained Model Accuracy: 0.2752994698858813
Retrained Model Name: model.joblib
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py retrain --tsize 0.4
Retraining Model
Retrained Model Accuracy: 0.24905929566719742
Retrained Model Name: model.joblib
```

Figura 7-20. Reciclagem do modelo

Também podes consultar uma API implementada, que em breve abordará o CLI, permitindo-te alterar tanto o anfitrião como o valor passado para a API. Esta etapa usa a biblioteca `requests`. Pode ajudar a construir um exemplo Python "puro" de uma ferramenta de previsão versus apenas prever com um comando `curl`. Podes ver um exemplo da saída na Figura 7-21:

```
./utilscli.py predict --weight 400
```

```
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py predict
Querying host http://localhost:8080/predict with weight: 225
result: {
    "prediction": {
        "height_human_readable": "6 foot, 3 inches",
        "height_inches": 75.09
    }
}

(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $ ./utilscli.py predict --weight 400
Querying host http://localhost:8080/predict with weight: 400
result: {
    "prediction": {
        "height_human_readable": "7 foot, 1 inches",
        "height_inches": 85.45
    }
}
```

Figura 7-21. Prever pedidos

Talvez estejas convencido de que as ferramentas CLI são a forma ideal de implementar rapidamente modelos ML, sendo assim uma verdadeira

organização orientada para MLOps. Que mais poderias fazer? Aqui tens mais duas ideias.

Primeiro, podes construir um cliente mais sofisticado que faça pedidos HTTP assíncronos a um serviço web implementado. Esta funcionalidade é uma das vantagens de construir ferramentas utilitárias em Python puro. Uma biblioteca a considerar para HTTPS assíncrono é a [Fast API](#).

Em segundo lugar, podes implementar continuamente a própria CLI. Para muitas empresas de SaaS, laboratórios universitários e muitos outros cenários, esse pode ser o fluxo de trabalho ideal para adaptar a velocidade e o Agile como um objetivo principal. Neste exemplo de projeto do GitHub, há um exemplo simples de [como conteinerizar uma ferramenta de linha de comando](#). Os três arquivos críticos são o Makefile, o *cli.py* e o Dockerfile.

Repara que o Makefile facilita o "lint" da sintaxe do Dockerfile utilizando [hadolint](#):

```
install:  
    pip install --upgrade pip &&\n    pip install -r requirements.txt  
  
lint:  
    docker run --rm -i hadolint/hadolint < Dockerfile
```

O CLI em si é bastante pequeno, um dos aspectos valiosos da estrutura Click:

```
#!/usr/bin/env python  
import click  
  
@click.command()  
@click.option("--name")  
def hello(name):  
    click.echo(f'Hello {name}!')  
  
if __name__ == '__main__':  
    #pylint: disable=no-value-for-parameter  
    hello()
```

Finalmente, o Dockerfile constrói o contentor:

```
FROM python:3.7.3-stretch

# Working Directory
WORKDIR /app

# Copy source code to working directory
COPY . app.py /app/

# Install packages from requirements.txt
# hadolint ignore=DL3013
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir --trusted-host pypi.python.org -r
requirements.txt
```

Para executar este contentor exato, podes fazer o seguinte:

```
docker run -it noahgift/cloudapp python app.py --name "Big John"
```

O resultado é o seguinte:

```
Hello Big John!
```

Este fluxo de trabalho é ideal para ferramentas CLI baseadas em ML! Por exemplo, para construir este contentor e enviá-lo, podes fazer o seguinte fluxo de trabalho:

```
docker build --tag=<tagname> .
docker push <repo>/<name>:<tagname>
```

Esta secção abordou ideias sobre como criar andaimes para projectos de aprendizagem automática. Em particular, vale a pena considerar três ideias principais: usar contentores, construir um microsserviço web e usar ferramentas de linha de comandos. De seguida, vamos abordar os microsserviços Flask com mais detalhe.

Microsserviço Flask

Ao lidar com fluxos de trabalho MLOps, é essencial observar que um microsserviço Flask ML pode operar de várias maneiras. Esta secção cobre

muitos desses exemplos.

Vamos primeiro dar uma olhada no núcleo da aplicação de microsserviço de aprendizado de máquina Flask no exemplo a seguir. Nota, mais uma vez, que a maior parte do trabalho pesado é feito através da biblioteca *mlib.py*. O único código "real" é a rota do Flask que faz o seguinte
@app.route("/predict", methods=['POST']) post request.
Aceita um payload JSON semelhante a {"Weight": 200}, depois devolve um resultado JSON:

```
from flask import Flask, request, jsonify
from flask.logging import create_logger
import logging

from flask import Flask, request, jsonify
from flask.logging import create_logger
import logging

import mlib

app = Flask(__name__)
LOG = create_logger(app)
LOG.setLevel(logging.INFO)

@app.route("/")
def home():
    html = f"<h3>Predict the Height From Weight of MLB
Players</h3>"
    return html.format(format)

@app.route("/predict", methods=['POST'])
def predict():
    """Predicts the Height of MLB Players"""

    json_payload = request.json
    LOG.info(f"JSON payload: {json_payload}")
    prediction = mlib.predict(json_payload['Weight'])
    return jsonify({'prediction': prediction})

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080, debug=True)
```

Este serviço Web Flask é executado de uma forma simples utilizando `python app.py`. Por exemplo, executa o microsserviço Flask da seguinte forma com o comando `python app.py`:

```
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $  
python app.py  
* Serving Flask app "app" (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a  
production...  
Use a production WSGI server instead.  
* Debug mode: on  
INFO:werkzeug: * Running on http://127.0.0.1:8080/ (Press CTRL+C  
to quit)  
INFO:werkzeug: * Restarting with stat  
WARNING:werkzeug: * Debugger is active!  
INFO:werkzeug: * Debugger PIN: 251-481-511
```

Para fazer uma previsão contra a aplicação, executa `predict.sh`. Repara que um pequeno script bash pode ajudar a depurar a tua aplicação sem teres de escrever todo o jargão de `curl`, o que pode fazer com que cometas erros de sintaxe:

```
#!/usr/bin/env bash  
  
PORT=8080  
echo "Port: $PORT"  
  
# POST method predict  
curl -d '{  
    "Weight":200  
}'\\"  
    -H "Content-Type: application/json" \  
    -X POST http://localhost:$PORT/predict
```

Os resultados da previsão mostram que o ponto de extremidade do Flask retorna uma carga útil JSON:

```
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook (main) $  
./predict.sh  
Port: 8080  
{
```

```
        "prediction": {
            "height_human_readable": "6 foot, 2 inches",
            "height_inches": 73.61
        }
    }
```

Nota que a ferramenta anterior *utilscli.py* também podia fazer pedidos web a este endpoint. Também podes usar o [httpie](#) ou a ferramenta [postman](#). A seguir, vamos discutir como uma estratégia de contentorização funciona para este microsserviço.

Microsserviço Flask em contentor

Segue-se um exemplo de como construir o contentor e executá-lo localmente (podes encontrar o conteúdo de *predict.sh* no [GitHub](#)).

```
#!/usr/bin/env bash

# Build image
#change tag for new container registry, gcr.io/bob
docker build --tag=noahgift/mllops-cookbook .

# List docker images
docker image ls

# Run flask app
docker run -p 127.0.0.1:8080:8080 noahgift/mllops-cookbook
```

A adição de um fluxo de trabalho de contentor é simples e permite um método de desenvolvimento mais fácil, uma vez que pode partilhar um contentor com outra pessoa da sua equipa. Também abre a opção de implantar seu aplicativo de aprendizado de máquina em muitas outras plataformas. Em seguida, vamos falar sobre como criar e implantar contêineres automaticamente.

Cria automaticamente um contentor através de GitHub Actions e faz push para GitHub Container Registry

Conforme abordado anteriormente no livro, o fluxo de trabalho do contêiner de GitHub Actions é um ingrediente valioso para muitas receitas. Pode

fazer sentido construir um contêiner programaticamente com GitHub Actions e enviá-lo para o Registro de contêiner do GitHub. Esta etapa pode servir tanto como um teste do processo de construção do contêiner quanto como alvo de implantação - digamos que você esteja implantando a ferramenta CLI discutida anteriormente. Este exemplo é o que esse código parece na prática. Observe que você precisaria alterar o `tags` para o seu Registro de contêiner (mostrado na [Figura 7-22](#)):

```
build-container:
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2
    - name: Logging to GitHub registry
      uses: docker/login-action@v1
      with:
        registry: ghcr.io
        username: ${github.repository_owner}
        password: ${secrets.BUILDCONTAINERS}
    - name: build flask app
      uses: docker/build-push-action@v2
      with:
        context: ./
        #tags: alfredodeza/flask-roberta:latest
        tags: ghcr.io/noahgift/python-mlops-cookbook:latest
        push: true
```

```
> ✓ Logging to Github registry
  ↴
  ✓ build flask app
    ↴
    1 ► Run docker/build-push-action@v2
    10 🚀 Buildx version: 0.5.1
    11 ⏳ Starting build...
    12 /usr/bin/docker buildx build --tag ghcr.io/noahgift/python-mlops-cookbook:latest --iidfile /tmp/docker-build-push-BV5GHC/iidfile --push .
    13 #1 [internal] load build definition from Dockerfile
    14 #1 sha256:fc6ccee5f6431d295e5b88dbde8228b6191df5c6473c60e26796d38328fa5b50
    15 #1 transferring dockerfile: 417B 0.0s done
    16 #1 DONE 0.0s
    17
    18 #2 [internal] load .dockerignore
    19 #2 sha256:b102fa427a5175dbb7d96426d01d8385230d10c0b5b2dc6a07e6c8bc28621ae4
    20 #2 transferring context: 2B done
    21 #2 DONE 0.0s
    22
    23 #3 [internal] load metadata for docker.io/library/python:3.8.8-slim-buster
    24 #3 sha256:0b88c6e6145102e0abd0b86aa5c1e180712e674f1ef65e77bdc4e8b3d32797e1
    25 #3 ...
```

Figura 7-22. Registro de contentores do GitHub

Os sistemas de construção SaaS (software as a service) e os registos de contentores SaaS são úteis fora do ambiente Cloud central. Eles verificam para os desenvolvedores, dentro e fora da sua empresa, que o fluxo de trabalho do contêiner é válido. A seguir, vamos unir muitos dos conceitos deste capítulo e usar uma oferta de PaaS de alto nível.

AWS App Runner Microsserviço Flask

O AWS App Runner é um serviço de alto nível que simplifica drasticamente os MLOps. Por exemplo, as receitas anteriores do livro de receitas do Python MLOps são fáceis de integrar com apenas alguns cliques no serviço AWS App Runner. Além disso, como mostra a [Figura 7-23](#), o AWS App Runner aponta para um repositório de código-fonte e fará a implantação automática em cada alteração no GitHub.

Depois de implantado, é possível abrir o AWS Cloud9 ou o AWS CloudShell, clonar o repositório do Python MLOps Cookbook e usar o *utilscli.py* para consultar o endpoint fornecido pelo serviço App Runner. A consulta bem-sucedida no AWS CloudShell é exibida na [Figura 7-24](#).

Repository type

Container registry
 Deploy your service from a container image stored in a container registry.

Source code repository
 Deploy your service from code hosted in a source code repository.

Connect to GitHub [Info](#)

App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.

[Add new](#)

Repository

[C](#)

Branch

[C](#)

Deployment settings

Deployment trigger

Manual
 Start each deployment yourself using the App Runner console or AWS CLI.

Automatic
 Every push to this branch deploys a new version of your service.

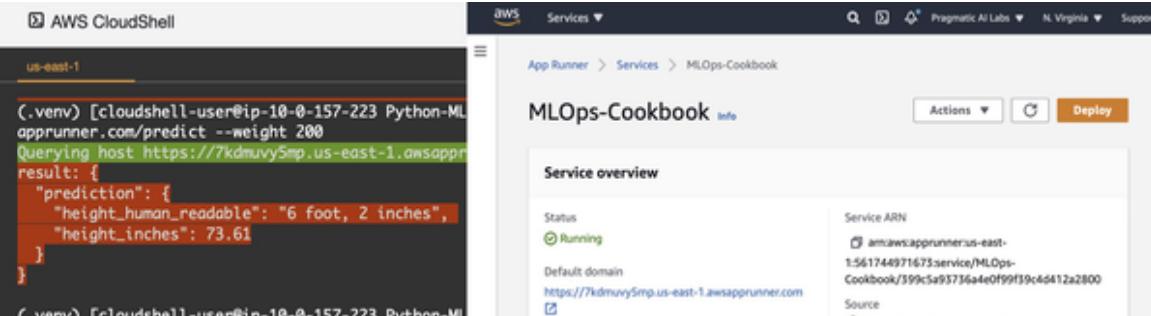


Figura 7-23. AWS App Runner

AWS CloudShell

```
us-east-1
.cvenv) [cloudshell-user@ip-10-0-157-223 Python-ML]
apprunner.com/predict --weight 200
Querying host https://7kdmuvy5mp.us-east-1.awssapp
result: {
  "prediction": {
    "height_human_readable": "6 foot, 2 inches",
    "height_inches": 73.61
  }
}
(.venv) [cloudshell-user@ip-10-0-157-223 Python-ML]
```

Services

App Runner > Services > MLOps-Cookbook

MLOps-Cookbook [Info](#)

Actions [C](#) [Deploy](#)

Service overview

Status	Running
Default domain	https://7kdmuvy5mp.us-east-1.awssapprunner.com
Source	Amazon ECR

Figura 7-24. Resultado da previsão do AWS App Runner

Em suma, os serviços AWS de alto nível permitem-te fazer MLOps de forma mais eficiente, uma vez que menos esforço vai para a construção DevOps processos . Então, a seguir, vamos passar para outra opção de computador para AWS, o AWS Lambda.

Receitas Lambda da AWS

Instala o SAM (AWS Serverless Application Model) de acordo com as instruções da documentação da AWS. O AWS Cloud9 já o tem instalado. Podes encontrar as receitas no GitHub. O AWS Lambda é essencial devido à sua profunda integração com a AWS. A seguir, vamos explorar como usar as práticas recomendadas modernas para implantar um modelo de ML sem servidor.

Um método eficiente e recomendado para implantar software na produção é através do SAM. A inovação desta abordagem combina funções Lambda, fontes de eventos e outros recursos como um processo de implementação e um conjunto de ferramentas de desenvolvimento.

Em particular, os principais benefícios do SAM, de acordo com a AWS, incluem configuração de implantação única, uma extensão do AWS CloudFormation, práticas recomendadas incorporadas, depuração e teste locais e integração profunda com ferramentas de desenvolvimento, incluindo a minha favorita, Cloud9.

Para começar, primeiro tens de instalar o AWS SAM CLI. Depois disso, podes consultar o guia oficial para obteres os melhores resultados.

AWS Lambda-SAM Local

Para começar a utilizar o SAM Local, pode experimentar o seguinte fluxo de trabalho para um novo projeto:

- Instala o SAM (como mostrado anteriormente)
- `sam init`
- `sam local invoke`

NOTA

Se estiveres a construir em Cloud9, é provavelmente uma boa ideia redimensionar usando [utils/resize.sh](#):

```
utils/resize.sh 30
```

Este truque dá-te mais tamanho de disco para criares vários contentores com o SAM local ou qualquer outro fluxo de trabalho do AWS Container.

Segue-se um esquema típico de arranque SAM, que é apenas ligeiramente diferente para um projeto ML:

```
└── sam-app/
    ├── README.md
    ├── app.py
    ├── requirements.txt
    ├── template.yaml
    └── tests
        └── unit
            └── __init__.py
                └── test_handler.py
```

Com este conhecimento básico, vamos passar a fazer mais com o AWS Lambda e o SAM.

Implantação em contêineres do AWS Lambda-SAM

Agora vamos mergulhar em um fluxo de trabalho em contêiner para SAM, já que ele suporta o registro de um contêiner que o AWS Lambda usa. Podes ver o [SAM-Lambda contentorizado a implementar o projeto no seguinte repositório](#). Primeiro, vamos abordar os componentes principais. As etapas críticas para implantar no SAM incluem os seguintes arquivos:

- *App.py* (o ponto de entrada do AWS Lambda)
- O *Dockerfile* (o que é construído e enviado para o Amazon ECR)
- *Template.yaml* (utilizado pelo SAM para implementar a aplicação)

O manipulador lambda faz muito pouco porque o trabalho duro ainda está acontecendo na biblioteca *mlib.py*. Uma "pegadinha" a ter em conta com o AWS Lambda é que precisas de usar um tratamento lógico diferente nas funções lambda, dependendo de como são invocadas. Por exemplo, se o Lambda for invocado através da consola ou Python, então não existe um corpo de pedido Web, mas no caso de integração com o API Gateway, o payload tem de ser extraído do **body** do evento:

```
import json
import mlib

def lambda_handler(event, context):
    """Sample pure Lambda function"""

    #Toggle Between Lambda function calls and API Gateway
    Requests
    print(f"RAW LAMBDA EVENT BODY: {event}")
    if 'body' in event:
        event = json.loads(event["body"])
        print("API Gateway Request event")
    else:
        print("Function Request")

    #If the payload is correct predict it
    if event and "Weight" in event:
        weight = event["Weight"]
        prediction = mlib.predict(weight)
        print(f"Prediction: {prediction}")
        return {
            "statusCode": 200,
            "body": json.dumps(prediction),
        }
    else:
        payload = {"Message": "Incorrect or Empty Payload"}
        return {
            "statusCode": 200,
            "body": json.dumps(payload),
        }
```

O projeto contém um Dockerfile que constrói o Lambda para a localização do ECR. Observa que ele pode ser empacotado no contêiner do Docker no

caso de um modelo de ML menor e até mesmo ser retrainado e empacotado programaticamente por meio do AWS CodeBuild:

```
FROM public.ecr.aws/lambda/python:3.8

COPY model.joblib mlib.py htwtmlb.csv app.py requirements.txt .

RUN python3.8 -m pip install -r requirements.txt -t .

# Command can be overwritten by providing a different command
CMD ["app.lambda_handler"]
```

O modelo SAM controla a camada IaC (Infraestrutura como código), permitindo um processo de implementação fácil:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
    python3.8

Sample SAM Template for sam-ml-predict

Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldMLFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /predict
            Method: post
      Metadata:
        Dockerfile: Dockerfile
        DockerContext: ./ml_hello_world
        DockerTag: python3.8-v1

Outputs:
  HelloWorldMLApi:
    Description: "API Gateway endpoint URL for Prod stage for"
```

```

Hello World ML
    function"
    Value: !Sub "https://${ServerlessRestApi}.execute-
api.${AWS::Region}.\amazonaws.com/Prod/predict/"
HelloWorldMLFunction:
    Description: "Hello World ML Predict Lambda Function ARN"
    Value: !GetAtt HelloWorldMLFunction.Arn
HelloWorldMLFunctionIamRole:
    Description: "Implicit IAM Role created for Hello World ML
function"
    Value: !GetAtt HelloWorldMLFunctionRole.Arn

```

Os únicos passos que restam a fazer são executar dois comandos: `sam build` e `sam deploy --guided`, o que lhe permite percorrer o processo de implantação. Por exemplo, na [Figura 7-25](#), o aviso `sam build` constrói o contêiner e, em seguida, prompt para testar localmente via `sam local invoke` ou fazer o deploy guiado.

```

(.venv) ec2-user:~/environment/Python-MLOps-Cookbook/recipes/aws-lambda-sam/sam-ml-predict (main) $ sam build
Building codeuri: . runtime: None metadata: {'Dockerfile': 'Dockerfile', 'DockerContext': './ml_hello_world', 'DockerTag': 'python3.8-v1'} functions: ['HelloWorldMLFunction']
Building image for HelloWorldMLFunction function
Setting DockerBuildArgs: {} for HelloWorldMLFunction function
Step 1/4 : FROM public.ecr.aws/lambda/python:3.8
--> cd95409d1c53
Step 2/4 : COPY model.joblib mlib.py htwtmbl.csv app.py requirements.txt .
--> Using cache
--> 5529737bad2f
Step 3/4 : RUN python3.8 -m pip install -r requirements.txt -t .
--> Using cache
--> bd1b0eb99988
Step 4/4 : CMD ["app.lambda_handler"]
--> Using cache
--> ab29da98e728
Successfully built ab29da98e728
Successfully tagged helloworldmlfunction:python3.8-v1

Build Succeeded

Built Artifacts : .aws-sam/build
Built Template : .aws-sam/build/template.yaml

Commands you can use next
=====
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided

```

Figura 7-25. Construção SAM

Podes invocar um `sam local invoke -e payload.json` com o seguinte corpo para testar localmente:

```
{
    "Weight": 200
}
```

Observa que o contêiner gira e uma carga útil é enviada e devolvida. Este processo de teste é valioso antes de implantares a aplicação no ECR:

```
(.venv) ec2-user:~/environment/Python-MLOps-Cookbook/recipes/aws-lambda-sam/sam-ml-predict
Building image.....
Skip pulling image and use local one: helloworldmlfunction:rapid-1.20.0.
START RequestId: d104cf8a-ce6b-4f50-9f2b-c82e99b8016f Version:
$LATEST
RAW LAMBDA EVENT BODY: {"Weight": 200}
Function Request
Prediction: {"height_inches": 73.61, "height_human_readable": "6 foot, 2 inches"}
END RequestId: d104cf8a-ce6b-4f50-9f2b-c82e99b8016f
REPORT RequestId: d104cf8a-ce6b-4f50-9f2b-c82e99b8016f Init
Duration: 0.08 ms Duration: 2187.82
{"statusCode": 200, "body": "{\"height_inches\": 73.61, \"height_human_readable\": \"6 foot, 2 inches\"}"}
```

É possível fazer uma implantação guiada com os testes fora do caminho, como mostrado na [Figura 7-26](#). Observa especialmente que os prompts guiam cada etapa do processo de implantação se você selecionar esta opção.



The screenshot shows a terminal window with the AWS SAM CLI command 'sam deploy' running. The output is as follows:

```
Setting default arguments for 'sam deploy'
=====
Stack Name [sam-app]: sam-ml-hello
AWS Region [us-east-1]:
Image Repository for HelloWorldMLFunction: 561744971673.dkr.ecr.us-east-1.amazonaws.com/sam-ml-hello
    helloworldmlfunction:python3.8-v1 to be pushed to 561744971673.dkr.ecr.us-east-1.amazonaws.com/sam-ml-hello:helloworldmlfunction-adea6346d767-python3.8-v1

#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [y/N]: y
#SAM needs permission to be able to create roles to connect to the resources in your template
Allow SAM CLI IAM role creation [Y/n]: y
HelloWorldMLFunction may not have authorization defined, Is this okay? [y/N]: Y
Save arguments to configuration file [Y/n]: Y
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:

Looking for resources needed for deployment: Found!
```

Figura 7-26. Destacamento guiado por SAM

Depois de implantar o lambda, há várias maneiras de usá-lo e testá-lo. Talvez uma das mais fáceis seja verificar se a imagem está no AWS Lambda Console (veja a [Figura 7-27](#)).

The screenshot shows the AWS Lambda execution results page. At the top, there's a green header bar with a checkmark icon and the text "Execution result: succeeded (logs)". Below this, a "Details" section is expanded, showing a JSON log entry:

```
{  
  "statusCode": 200,  
  "body": "{\"height_inches\": 73.61, \"height_human_readable\": \"6 foot, 2 inches\"}"  
}
```

Below the log, there's a "Summary" section with the following data:

Code SHA-256	Request ID
5a18474df6bef7f95b3d2c601a1b921814ef13eb90f3970db148578351dc09e8	a4262a19-60f9-409b-8421-084ee5981aa0
Duration	Billed duration
260.73 ms	261 ms
Resources configured	Max memory used
128 MB	104 MB

Under "Log output", it says: "The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group." followed by the log entries:

```
START RequestId: a4262a19-60f9-409b-8421-084ee5981aa0 Version: $LATEST  
Event Body {'Weight': 200}  
Prediction: {'height_inches': 73.61, 'height_human_readable': '6 foot, 2 inches'}  
END RequestId: a4262a19-60f9-409b-8421-084ee5981aa0  
REPORT RequestId: a4262a19-60f9-409b-8421-084ee5981aa0 Duration: 260.73 ms Billed Duration: 261 ms Memory Size: 128 MB  
Max Memory Used: 104 MB
```

Figura 7-27. Testa o AWS Lambda no Console

Duas maneiras de testar a API em si incluem o Console do AWS Cloud9 e também a ferramenta Postman. As Figuras 7-29 e 7-30 mostram exemplos de ambas.

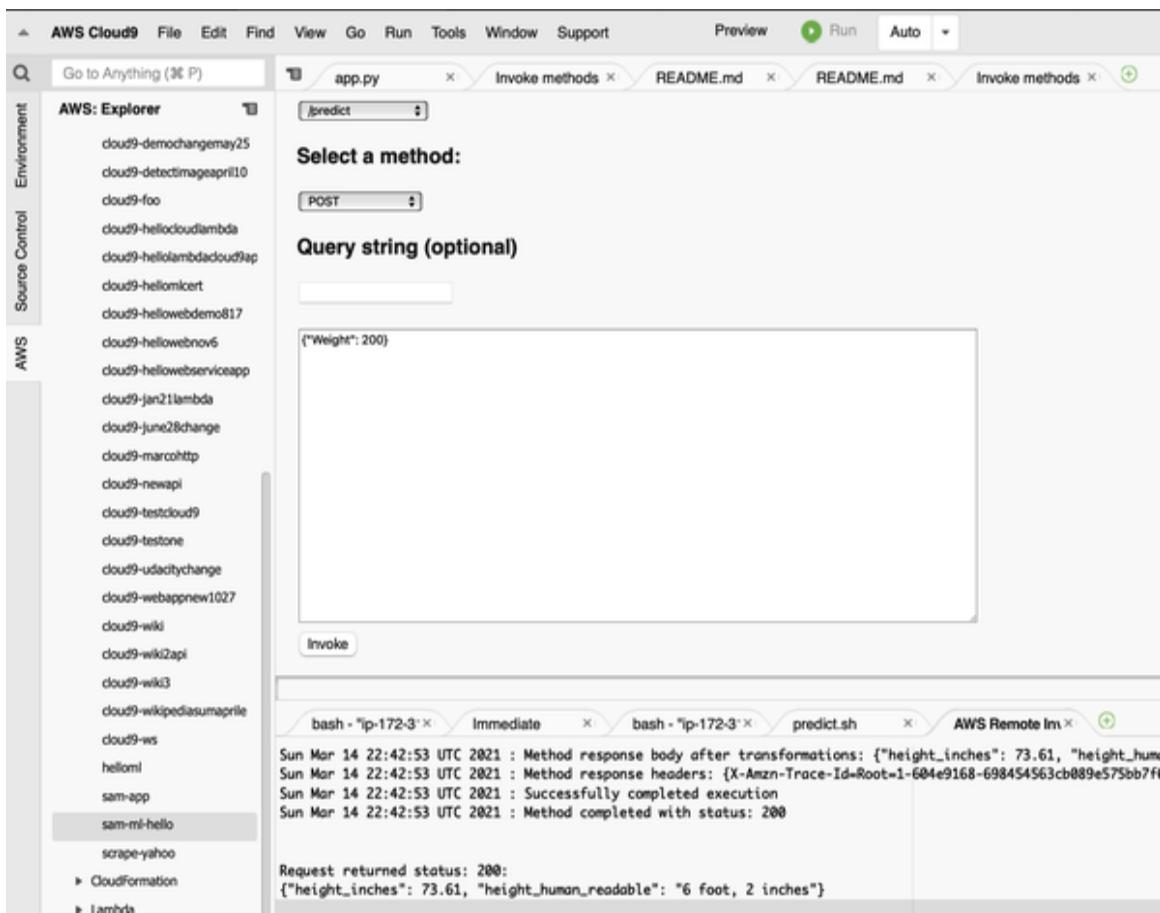


Figura 7-28. Invoca o Lambda Cloud9

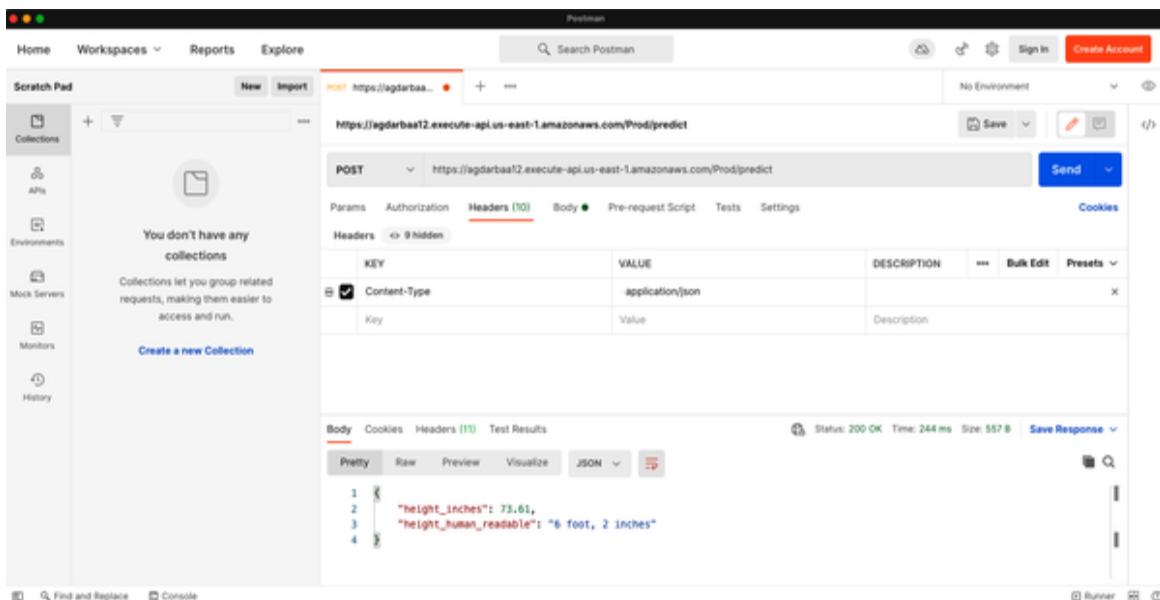


Figura 7-29. Testa o AWS Lambda com o Postman

Outros destinos de implantação para considerar a implantação dessa receita básica de aprendizado de máquina incluem o Flask Elastic Beanstalk e o AWS Fargate. Diferentes variações da receita incluem outros serviços HTTP, como o [fastapi](#), ou o uso de um modelo pré-treinado disponível por meio da API AWS Boto3 em vez de treinar seu modelo.

O AWS Lambda é uma das tecnologias mais interessantes para criar sistemas distribuídos que incorporam engenharia de dados e engenharia de aprendizagem automática. Vamos falar de um estudo de caso do mundo real a seguir.

Aplicar a aprendizagem automática da AWS ao mundo real

Vamos mergulhar em exemplos de como usar os recursos de aprendizado de máquina da AWS no mundo real. Nesta secção, vários MLOps abordam o que está envolvido em empresas reais que utilizam a aprendizagem automática.

NOTA

Há muitas formas de utilizar a AWS para MLOps e um número quase infinito de combinações de serviços no mundo real. Aqui tens uma lista parcial de padrões de engenharia de aprendizagem automática recomendados na AWS:

Contentor como um serviço (CaaS)

Para as organizações que estão a lutar para começar algo, o CaaS é um excelente local para iniciar a jornada MLOps. Um serviço recomendado é o AWS App Runner.

SageMaker

Para organizações de maior dimensão, com diferentes equipas e grandes volumes de dados, a SageMaker é uma excelente plataforma, uma vez que permite uma segurança refinada e uma implementação e formação de nível empresarial.

Sem servidor com APIs de IA

Para pequenas startups que precisam de se mover muito rapidamente, uma excelente abordagem inicial para fazer MLOps é usar modelos pré-treinados através de uma API juntamente com uma tecnologia sem servidor como AWS Lambda.

ESTUDO DE CASO: REDE SOCIAL DE DESPORTO

Esta secção é um estudo de caso sobre uma rede social de desporto criada com base na aprendizagem automática da AWS, com o antigo diretor de gestão de produtos, Rob Loranger. A entrada de Rob no mundo dos dados e da sua importância para a tomada de decisões de arquitetura começou em 2005, quando se tornou engenheiro de vendas especializado em ferramentas de software utilizadas para conceber, desenvolver e gerir bases de dados relacionais, acabando por se tornar gestor de produto de uma plataforma de arquitetura de dados empresariais. Desde então, Rob manteve-se como gestor de produtos, ganhando experiência em plataformas de software orientadas para a aprendizagem automática, desde redes sociais a centros de dados e análise e controlo de redes de telecomunicações.

Rob obteve um MBA na Universidade da Califórnia, Davis, onde estudou análise de dados e aprendizagem automática. Ganhou experiência em todas as atividades do pipeline de ML, desde a formulação do problema de negócios até a escolha, avaliação e implantação de um modelo de aprendizado de máquina. As ferramentas que utilizou ao longo do caminho incluem Amazon SageMaker, Amazon Forecast, Jupyter Notebooks, Minitab, Stata, Weka, Python e R.

P: Quais são 3-5 dos aspectos mais importantes a ter em conta na implementação e manutenção de sistemas de aprendizagem automática em escala?

R: A aprendizagem automática não é um processo único. Tens de estar empenhado em avaliar continuamente os dados e a precisão do modelo à medida que estes mudam ao longo do tempo. Tal como acontece com todos os projectos de impacto, envolve o SME (especialista na matéria) desde o início e com frequência, para que possas construir a melhor compreensão do problema empresarial e dos dados a utilizar para testar e validar o teu modelo de aprendizagem automática.

Certifica-te de que o problema comercial que estás a tentar resolver com a aprendizagem automática e o impacto da resolução desse problema são tão simples quanto possível para obter o máximo apoio dos intervenientes críticos ao longo da vida do projeto.

P: Explica como é que um gestor de produtos de IA/ML pensa na criação de produtos de ML.

R: Enquanto gestor de produto da plataforma social Sqor Sports, o KPI que mais me interessava, por ser um forte indicador da aderência da plataforma, era o de utilizadores activos mensais (MAU). Além disso, para determinar se os utilizadores valorizavam realmente a plataforma, era essencial ver esta métrica crescer significativamente ao longo do tempo. Mesmo que a métrica mantivesse um estado estacionário aparentemente bom, a plataforma poderia estar a falhar. O estado estacionário poderia ser atribuído a situações como um afluxo de novos utilizadores mês após mês que estivessem activos apenas durante o primeiro mês antes de saírem ou um núcleo sólido de utilizadores de nicho que considerassem a plataforma valiosa e continuassem a utilizá-la ao longo do tempo. Qualquer uma destas situações ficaria muito aquém da missão da Sqor Sports de ser a plataforma social número um para atletas, equipas e respectivos fãs se relacionarem de forma significativa uns com os outros.

Era essencial construir uma plataforma rica que atraísse novos utilizadores e proporcionasse uma experiência excitante e única que não se encontra em mais lado nenhum. Por conseguinte, a nossa estratégia melhorou a aquisição de fãs, a retenção de fãs e o crescimento viral através dos fãs (ou seja, fãs que ajudam a adquirir novos fãs). As melhorias na plataforma nestas áreas refletir-se-iam no nosso KPI principal e o MAU não cresceria de forma satisfatória até que todas estas áreas estivessem corretas.

Tirando partido das relações existentes com atletas e equipas mantidas pela nossa equipa executiva, conselho de administração e

equipa comercial, aumentámos rapidamente o número de atletas e grupos na plataforma. Por sua vez, aproveitando o conteúdo social dos atletas e das equipas criado no Sqor e distribuído por outras redes sociais, adquirimos rapidamente fãs desses atletas e equipas. Além disso, o crescimento viral é um tema vasto que não vamos explorar completamente aqui. No entanto, foi também uma parte essencial da nossa estratégia, reflectida no roteiro do produto como funcionalidades que proporcionaram aos fãs conteúdos e experiências únicas e melhoraram a capacidade dos fãs para criar e promover relações e partilhar conteúdos dentro e fora da plataforma.

No entanto, no início, a retenção de fãs revelou-se um desafio, uma vez que, embora tivéssemos muitos conteúdos criados pelos nossos atletas e equipas, os fãs tinham dificuldade em descobrir e seguir novos atletas/equipas. Por conseguinte, os fãs tinham normalmente uma pequena rede de atletas e equipas e não recebiam conteúdo suficiente da plataforma. Como resultado, a maioria dos fãs tornou-se inativa após o primeiro mês.

Um aspeto fundamental para resolver este problema de retenção estava enraizado no nosso motor de recomendação baseado na aprendizagem automática. Infelizmente, as primeiras iterações do motor sofreram com a baixa entrada de dados, o que levou a recomendações de feeds de fãs fracas. No entanto, à medida que melhorámos os dados e o algoritmo do motor de recomendação, começámos a ver as melhorias na retenção de fãs que pretendíamos, como se pode ver através da análise de coortes.

Por último, só com um MAU considerável e em crescimento é que poderíamos apoiar a nossa estratégia de receitas que, no início, consistia apenas numa partilha de receitas com o merchandise dos atletas vendido na plataforma, mas com o objetivo final de fornecer uma plataforma empresarial para gerir a presença social (por exemplo, atividade nas redes sociais, principais fãs, vendas de merchandise, etc.).

P: Como é que as pessoas podem entrar em contacto contigo e o que gostarias de partilhar com os leitores sobre o que estás a trabalhar?

R: Podes contactar-me por e-mail(rtloranger@gmail.com) ou pelo [Linkedin](#).

ESTUDO DE CASO: CONSELHOS DE CARREIRA COM JULIEN SIMON, EVANGELISTA DE APRENDIZAGEM AUTOMÁTICA DA AWS

Julien Simon é um criador de conteúdos prolífico e divulgador da plataforma de aprendizagem automática da AWS. Tive a sorte de o encontrar para obter algumas informações sobre a aprendizagem automática na plataforma AWS.

P: Qual é a tua formação e como é que te envolveste na operacionalização da aprendizagem automática?

R: Sou um engenheiro de software que acabou por passar 10 anos a liderar grandes equipas de software e infra-estruturas em várias startups. Recolher, armazenar e processar dados foi sempre uma atividade central nas minhas equipas. Com o tempo, evoluiu de bases de dados relacionais para business intelligence, análise em tempo real e aprendizagem automática. Esta última era, de facto, o núcleo de uma das empresas em que trabalhei. Dada a nossa escala na altura (cerca de 500 mil visitas por segundo), a formação e a implementação de modelos criaram todo o tipo de desafios interessantes, desde o armazenamento de dados à implementação (e, por vezes, à reversão) de modelos em centenas de servidores.

P: Quais são 3-5 dos aspectos mais importantes a ter em conta na implementação e manutenção de sistemas de aprendizagem automática em escala?

R: Primeiro, podemos concordar com o facto de que a aprendizagem automática é engenharia de software? Se concordarmos, então vamos também concordar que as melhores práticas como a rastreabilidade, o controlo de versões, os testes, a automatização e a documentação não são opcionais. Ainda me surpreende como os fluxos de trabalho de aprendizagem automática podem ser tão diferentes, porque "as coisas são diferentes". Não me parece que sejam de todo. É claro que a ciência dos dados é uma

disciplina muito específica, mas no final do dia, produz artefactos de código e dados, por isso não vamos reinventar a roda. Muitas das coisas que funcionaram bem durante décadas continuam a funcionar com a aprendizagem automática.

O código e os conjuntos de dados devem ser versionados e rastreados. O código e os modelos devem ser testados. Os modelos devem ser implantados com segurança na produção, com portas de qualidade e estratégias bem compreendidas (canário, azul-verde, etc.). A monitorização também é fundamental, tanto do ponto de vista da infraestrutura (taxa de transferência, latência) como do ponto de vista da aprendizagem automática (qualidade da previsão, desvio de dados). Por fim, os teus modelos também devem ser escalados para cima e para baixo de acordo com o tráfego de entrada. E sim, todos estes processos devem ser tão automatizados quanto possível.

Mais uma vez, nada de novo, mas vamos começar a fazê-lo também para a aprendizagem automática!

P: O que mais te entusiasma neste momento com a aprendizagem automática e porquê?

R: O preguiçoso que há em mim adora o facto de podermos transferir e implementar modelos com apenas algumas linhas de código. Iniciativas como o Hugging Face tornam extremamente fácil adicionar modelos de última geração à tua aplicação, mesmo que não sejas um especialista em aprendizagem automática. Em geral, tudo o que torna a aprendizagem automática mais acessível é um grande passo em frente: aprendizagem por transferência, AutoML e serviços de IA de alto nível que escondem completamente toda a complexidade. Costumo dizer que devemos tornar a aprendizagem automática tão simples como o Amazon S3, pelo que qualquer passo nessa direção é bem-vindo.

Do ponto de vista tecnológico, estou atento ao novo hardware que pode acelerar enormemente a formação e a inferência. As GPUs são

boas, mas, na maioria das vezes, parecem uma opção de força bruta. Estou definitivamente ansioso por alternativas mais elegantes, mais económicas e que consumam menos energia!

P: Quais são as 3-5 principais coisas que uma pessoa que esteja a ler este livro pode fazer para ter sucesso na sua carreira nos MLOps?

R: O meu conselho número um é aprenderes o máximo que puderdes sobre o problema comercial que estás a tentar resolver. A não ser que o conheças profundamente, não serás capaz de dar os passos certos. Com o que é que os utilizadores realmente se preocupam? Quais são as principais métricas a que deves estar atento? Qual é a melhor solução de compromisso entre custo e desempenho e tempo de colocação no mercado? Tudo isto é fundamental para alcançar o sucesso em qualquer projeto de aprendizagem automática. A tecnologia é apenas uma ferramenta. Dominá-la por si só ou para melhorar o teu currículo é inútil.

Além disso, também deves ser obcecado pela automatização, sem a qual não podes escalar as tuas operações. As tuas equipas de ciência de dados devem ser capazes de trabalhar completamente sozinhas, treinando, implementando e testando os seus modelos na sua própria conta. A equipa de operações só deve estar envolvida na gestão da infraestrutura de produção e não na infraestrutura de desenvolvimento e teste. Naturalmente, é necessário definir as portas de qualidade adequadas para evitar que os maus modelos entrem em produção. Se a aprovação manual funcionar melhor para ti, tudo bem, mas tudo o resto deve ser automatizado.

Por último, mas não menos importante, certifica-te de que tens a tua segurança coberta. Também aqui se devem aplicar as melhores práticas (princípio do menor privilégio, encriptação, auditoria, etc.). A AWS tem muitos serviços relacionados com a segurança (IAM, KMS, CloudTrail, políticas de balde S3, etc.), e deves definitely olhar para eles para construir uma plataforma segura de ciência de dados e aprendizagem automática.

P: Como é que as pessoas podem entrar em contacto contigo e o que gostarias de partilhar sobre o que estás a fazer?

R: Tenho todo o gosto em estabelecer uma ligação no LinkedIn, no Twitter [@julsimon](#) ou no [YouTube](#), onde construo e partilho o máximo de conteúdo possível, para ajudar os programadores a serem bem sucedidos com a aprendizagem automática na AWS!

Conclusão

Este capítulo aborda tanto as estradas mais percorridas quanto os cantos exclusivos da AWS. Uma das principais conclusões é que a AWS é a maior plataforma Cloud. Há muitas maneiras diferentes de abordar um problema usando a tecnologia AWS, desde a criação de uma empresa inteira que faz aprendizado de máquina, como os estudos de caso discutem, até o uso de APIs de visão computacional de alto nível.

Em particular, para os líderes empresariais e técnicos, recomendo as seguintes práticas recomendadas para Bootstrap das capacidades MLOps o mais rapidamente possível:

- Envolve-te com o Suporte Empresarial da AWS.
- Obtém a certificação da tua equipa em AWS, começando com o exame AWS Solutions Architect ou Cloud Practitioner e AWS Certified Machine Learning Specialty.
- Obtém ganhos rápidos utilizando APIs de IA como o AWS Comprehend, o AWS Rekognition e ofertas de PaaS de alto nível como o AWS App Runner ou o AWS Lambda.
- Concentra-te na automatização e garante que automatizas tudo o que puderdes, desde a ingestão de dados e o armazenamento de funcionalidades até à modelação e à implementação do modelo de aprendizagem automática.

- Começa a utilizar o SageMaker como um investimento a longo prazo da MLOps e utiliza-o para projectos a longo prazo ou mais complexos, juntamente com soluções mais acessíveis.

Por fim, se queres mesmo usar a AWS como indivíduo, incluindo iniciar uma carreira como especialista em aprendizagem automática da AWS, pode ser benéfico e lucrativo obter uma certificação. O [Apêndice B](#) dá-te uma rápida introdução sobre como te preparamos para os exames de certificação da AWS. Uma tarefa final recomendada seria fazeres alguns exercícios e perguntas de pensamento crítico para praticares mais a AWS.

O nosso próximo capítulo afasta-se da AWS e aprofunda-se no Azure.

Exercícios

- Constrói um pipeline de entrega contínua de aprendizagem automática para um serviço Web Flask utilizando o Elastic Beanstalk. Podes consultar o [repositório do GitHub](#) como ponto de partida.
- Inicia uma instância do Amazon SageMaker e cria e implementa os [dados do censo dos EUA para o exemplo de segmentação da população](#).
- Constrói um serviço de previsão de aprendizagem automática CaaS utilizando o AWS Fargate. Podes usar este [repositório do GitHub](#) como ponto de partida.
- Constrói um protótipo de engenharia de dados sem servidor utilizando este [repositório GitHub](#) como ponto de partida.
- Constrói um acionador de visão computacional que detecta etiquetas; utiliza este [repositório GitHub](#) como ponto de partida.
- Usa o projeto base do MLOps Cookbook e implanta em tantos alvos diferentes quanto possível: CLI em contentor, EKS, Elastic

Beanstalk, Instâncias Spot e qualquer outra coisa que possas imaginar.

Questões para discussão sobre pensamento crítico

- Porque é que as organizações que utilizam a aprendizagem automática utilizam um lago de dados? Qual é o principal problema que resolvem?
- Qual é o caso de utilização de modelos pré-construídos como o AWS Comprehend em vez de treinar o teu modelo de análise de sentimentos?
- Por que uma organização usaria o AWS SageMaker em vez de Pandas, sklearn, Flask e Elastic Beanstalk? Quais são os casos de uso para ambos?
- Quais são as vantagens de um processo de implementação de modelos de aprendizagem automática em contentores?
- Um colega diz que está confuso sobre por onde começar com a aprendizagem automática na AWS devido à variedade de ofertas. Como recomendadas que eles abordem a sua pesquisa?

Capítulo 8. MLOps para Azure

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

Uma terceira razão para a nossa família se mudar foi o facto de nunca termos tido uma casa própria desde 1933, quando, em plena Grande Depressão, fomos desalojados. Só anos mais tarde é que comprehendi como é que o paraíso rural que eu adorava aos 6 anos desapareceu. Os meus pais não conseguiam pagar as prestações. A minha mãe abandonou o seu consultório em dificuldades e arranjou um emprego como médica recepcionista num hospital estatal que lhe dava alguns dólares e um apartamento demasiado apertado para todos nós, pelo que o meu irmão e eu fomos enviados para aquilo a que mais tarde chamámos "O Exílio".

—Dr. Joseph Bogen

Os investimentos contínuos da Microsoft no Azure para a aprendizagem automática estão a dar frutos. O número de funcionalidades oferecidas atualmente faz com que a plataforma como um todo seja uma excelente oferta. Há alguns anos, nem sequer era claro que o Azure iria receber um tal influxo de engenharia de alto nível e um interesse crescente nos seus serviços.

Se ainda não experimentaste o Azure ou se ainda não viste nada relacionado com a aprendizagem automática na oferta de cloud da Microsoft, recomendo vivamente que lhe dês uma oportunidade. Tal como a maioria dos fornecedores de Cloud, está disponível um período experimental com créditos suficientes para experimentares e julgares por ti próprio.

Um dos exemplos que costumo dar é o da utilização de Kubernetes. Instalar, configurar e implementar um cluster Kubernetes não é *uma* tarefa simples. É ainda mais complicado se tiveres em conta a associação de um modelo de aprendizagem automática e o escalonamento das suas interações com potenciais consumidores. Este é um problema difícil de resolver corretamente. Se tiveres oportunidade de percorrer as definições de implementação de um modelo treinado, utilizando um cluster Kubernetes como alvo para o modelo, resume-se a selecionar o cluster a partir de um menu pendente.

Para além de todas as funcionalidades e abstrações de problemas complexos como a implementação de modelos em produção, é *muito refrescante* ver uma enorme quantidade de documentação detalhada. Embora este capítulo se concentre na realização de operações de aprendizagem automática no Azure, não consegue captar todos os pormenores interessantes. O **recurso de documentação principal** é um excelente local para marcar e aprofundar mais informações não abordadas aqui.

Neste capítulo, analiso algumas das opções interessantes no Azure, desde o treino de um modelo até à sua implementação em contentores ou num cluster Kubernetes. Uma vez que estão a tornar-se comuns nas ofertas de aprendizagem automática, vou mergulhar nos pipelines. Os pipelines podem melhorar ainda mais a automatização, mesmo quando a automatização tem origem fora da Cloud do Azure.

Existem muitas formas de executar tarefas de aprendizagem automática na plataforma: Azure ML Studio Designer, Jupyter Notebooks e AutoML, onde pode carregar um CSV e começar a treinar modelos imediatamente. Por fim, a maioria das funcionalidades (se não todas) tem suporte correspondente no SDK (abordado na secção seguinte). Essa flexibilidade é vital, pois permite-te escolher a solução que funciona melhor para ti. Portanto, não há necessidade de seguir uma maneira opinativa de operacionalizar modelos.

Por fim, abordarei recomendações práticas sobre a aplicação de alguns dos princípios fundamentais do DevOps, como a monitorização e o registo

utilizando funcionalidades do Azure MachineLearning.

NOTA

Embora este capítulo seja sobre o Azure Machine Learning, não abordará conceitos básicos como a criação e configuração de uma nova conta. Se ainda não experimentaste o serviço, [podes começar aqui](#).

CLI do Azure e Python SDK

Os vários exemplos e snippets de código neste capítulo pressupõem que tem a ferramenta de linha de comandos do Azure e o Python SDK (Software Development Kit) instalados e disponíveis no seu ambiente. Certifica-te de que [instalas a versão mais recente da CLI](#) e que a extensão de aprendizagem automática está disponível após a instalação:

```
$ az extension add -n azure-cli-ml
```

Na maioria das vezes, terás de autenticar a partir do teu sistema local de volta para o Azure. Este fluxo de trabalho é semelhante ao de outros fornecedores de Cloud. Para associar a tua conta do Azure ao ambiente atual, executa o seguinte comando:

```
$ az login
```

A maioria dos exemplos Python que utilizam o Python SDK do Azure requerem uma conta no Azure e o ficheiro *config.json* associado ao teu espaço de trabalho descarregado localmente. Este ficheiro tem todas as informações necessárias para associar o código Python ao espaço de trabalho. Todas as interações com o Python SDK devem utilizar este ficheiro para configurar o tempo de execução:

```
import azureml.core  
from azureml.core import Workspace
```

```
ws = Workspace.from_config()
```

Para recuperar o ficheiro *config.json*, inicia sessão no estúdio ML do Azure e clica no menu superior direito (denominado Alterar Subscrição). O submenu terá um link para baixar o arquivo *config.json* (consulte a Figura 8-1).

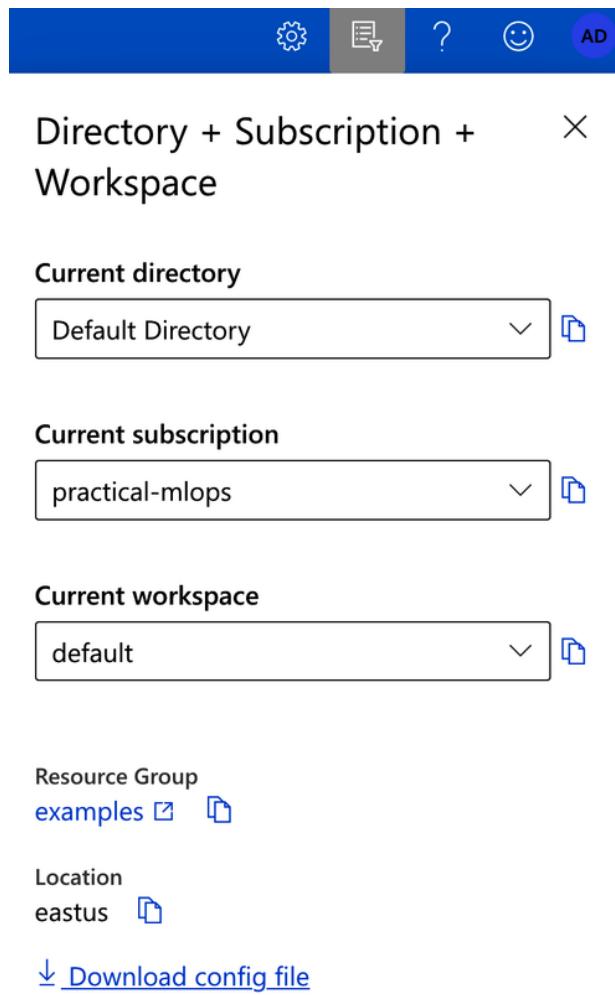


Figura 8-1. Configuração JSON do Azure

Se *config.json* não estiver presente no diretório de trabalho atual, a utilização do SDK falha com um traceback:

```
In [2]: ws = Workspace.from_config()
```

```
-----
```

```
-----
```

```
UserErrorException
```

```
Traceback (most recent)
```

```
call last)
<ipython-input-2-e469111f639c> in <module>
----> 1 ws = Workspace.from_config()

~/python3.8/site-packages/azureml/core/workspace.py in
from_config
    269
    270     if not found_path:
--> 271         raise UserErrorException(
    272             'We could not find config.json in: {} or in its parent
directories.'
    273             'Please provide the full path to the config file or
ensure that '
```

Agora que já abordei algumas das noções básicas de utilização do Azure CLI e do SDK, vou entrar em mais detalhes sobre a autenticação e algumas variantes que podes utilizar no Azure.

Autenticação

A autenticação deve ser uma das peças principais da automatização quando lida com serviços. O Azure tem uma *entidade de serviço* para acesso (e controlo de acesso) a recursos. Ao automatizar serviços e fluxos de trabalho, as pessoas tendem a ignorar ou mesmo a tentar simplificar a autenticação em geral. É muito comum ouvir recomendações como: "*utiliza apenas o utilizador root*" ou "*altera as permissões dos ficheiros para que qualquer pessoa possa escrever e executar*". Um engenheiro experiente saberá o que é melhor, mas apenas porque experimentou a dor depois de aceitar essas sugestões de segurança fraca e restrições de implantação. Compreendo muito bem o problema de tentar resolver uma situação como esta.

Quando trabalhava como Administrador de Sistemas numa agência de meios de comunicação, o Chefe de Engenharia entrou diretamente no ambiente de produção para tornar um ficheiro legível para qualquer pessoa (necessário para a aplicação PHP onde corria). Acabámos por descobrir o que se passava. O resultado da alteração significava que qualquer pedido HTTP (e qualquer pessoa na Internet) tinha permissões de leitura, escrita e

execução para esse ficheiro. Às vezes, uma simples correção pode ser tentadora e parecer o melhor caminho a seguir, mas nem sempre é o caso. Particularmente com segurança (e autenticação neste caso), tens de ser muito cético em relação a correcções simples que removem uma restrição de segurança.

Certifica-te sempre de que a autenticação é feita corretamente e não tentes ignorar ou contornar estas restrições, mesmo que seja uma opção.

Diretor de Serviços

No Azure, a criação de uma entidade de serviço envolve várias etapas. Dependendo das restrições de que necessitas para a conta e o acesso aos recursos, o processo varia de um exemplo para outro. Adapta o seguinte para se adequar a um cenário específico. Primeiro, depois de iniciar sessão com o CLI, executa o seguinte comando:

```
$ az ad sp create-for-rbac --sdk-auth --name ml-auth
```

Esse comando cria a entidade de serviço com o nome `ml-auth`. És livre de escolher o nome que quiseres, mas é sempre bom ter alguma convenção para te lembrares a que é que estes nomes estão associados. Em seguida, anota a saída e verifica o valor "`clientId`", que os próximos passos exigem:

```
[...]
Changing "ml-auth" to a valid URI of "http://ml-auth", which is
the required
format used for service principal names
Creating a role assignment under the scope of:
    "/subscriptions/xxxxxxxx-2cb7-4cc5-90b4-xxxxxxxxx24c6"
    Retrying role assignment creation: 1/36
    Retrying role assignment creation: 2/36
{
[...]
    "clientId": "xxxxxxxx-3af0-4065-8e14-xxxxxxxxxxxx",
[...]
    "sqlManagementEndpointUrl":
    "https://management.core.windows.net:8443/",
```

```
        "galleryEndpointUrl": "https://gallery.azure.com/",
        "managementEndpointUrl": "https://management.core.windows.net/"
    }
```

Agora utiliza o "`clientId`" para obter metadados da entidade de serviço recém-criada:

```
$ az ad sp show --id xxxxxxxx-3af0-4065-8e14-xxxxxxxxxxx
{
    "accountEnabled": "True",
    "appDisplayName": "ml-auth",
    ...
    ...
    ...
    "objectId": "4386304e-3af1-4066-8e14-475091b01502",
    "objectType": "ServicePrincipal"
}
```

Para permitir que a entidade de serviço aceda ao teu espaço de trabalho do Azure Machine Learning, tens de a associar ao espaço de trabalho e ao grupo de recursos:

```
$ az ml workspace share -w example-workspace \
    -g alfredodeza_rg_linux_centralus \
    --user 4386304e-3af1-4066-8e14-475091b01502 --role owner
```

Esse comando é o último necessário para concluir o processo de criação de uma entidade de serviço e associá-la à conta de aprendizado de máquina usando a função `owner`. O `example-workspace` é o nome do espaço de trabalho que utilizei no Azure e o `alfredodeza_rg_linux_centralus` é o grupo de recursos nesse espaço de trabalho. É uma pena que não haja saída da execução desse comando após uma chamada bem-sucedida.

Uma vez criada esta conta, podes utilizá-la para ativar a automatização com autenticação, o que evita os prompts e a autenticação constante. Certifica-te de que restringes o acesso e a função ao menor número de permissões necessárias.

NOTA

Estes exemplos utilizam o valor de função "owner" para a entidade de serviço, que tem permissões alargadas. O valor padrão para o sinalizador `--role` é "contributor", que é mais restrito. Adapta este valor ao que melhor se adequa ao teu ambiente (e utilização).

Autenticação de serviços API

Outros ambientes podem não beneficiar de uma conta de entidade de serviço, dependendo do fluxo de trabalho em causa. Esses serviços podem ser expostos à Internet, fornecendo interação com modelos implantados por meio de solicitações HTTP. Nesses casos, tens de decidir o tipo de autenticação a ativar.

Antes de implementar um modelo ou mesmo de configurar quaisquer definições para colocar um modelo em produção, tens de ter uma boa ideia sobre as diferentes funcionalidades de segurança disponíveis.

O Azure oferece diferentes formas de autenticação, dependendo do serviço com o qual tens de interagir. Os padrões para esses serviços também mudam, dependendo do tipo de implantação. Essencialmente, há suporte para dois tipos: chaves e tokens. O Serviço de Kubernetes do Azure (AKS) e as Instâncias de Contentor do Azure (ACI) têm suporte variável para estes tipos de autenticação.

Autenticação baseada em chaves:

- O AKS tem a autenticação baseada em chave activada por predefinição.
- A ACI tem a autenticação baseada em chave desactivada por predefinição (mas pode ser activada). Nenhuma autenticação é ativada por padrão.

Autenticação baseada em token:

- O AKS tem a autenticação baseada em token desactivada por predefinição.

- A ACI não suporta autenticação baseada em token.

Estes tipos de clusters para implementações são suficientemente significativos para serem compreendidos antes de implementares o modelo em produção. Mesmo para ambientes de teste, ativa sempre a autenticação para evitar uma incompatibilidade entre o desenvolvimento e a produção.

Computa as instâncias

O Azure tem uma definição para instâncias de computação que as descreve como uma *estaçao de trabalho gerida com base na cloud para cientistas*. Essencialmente, permite-te começar *muito rapidamente* com tudo o que podes precisar para realizar operações de aprendizagem automática na Cloud. Ao desenvolver provas de conceitos ou ao tentar algo novo a partir de um tutorial, podes aproveitar o excelente suporte para **Jupyter Notebooks** com uma enorme quantidade de dependências pré-instaladas e prontas a utilizar. Embora possas carregar os teus próprios notebooks, recomendo que comeceis por navegar pelas amostras completas que existem, como mostrado na [Figura 8-2](#).

Notebooks

The screenshot shows the Azure Notebooks interface. At the top, there are two tabs: "Files" and "Samples", with "Samples" being the active tab. Below the tabs is a search bar with the placeholder text "Search to filter notebooks". To the right of the search bar are two icons: a blue circular arrow icon and a double-left arrow icon. The main content area displays a hierarchical list of notebooks. It starts with a folder named "Samples", which contains a folder "1.28.0", which in turn contains several notebooks represented by brown folder icons: "how-to-use-azureml", "tutorials", "compute-instance-quickstarts", "create-first-ml-experiment", "image-classification-mnist-data", "machine-learning-pipelines-advanced", and "regression-automl-nyc-taxi-data".

Figura 8-2. Amostras do bloco de notas do Azure

Um dos problemas mais incómodos a resolver quando se tenta começar a utilizar notebooks e a mexer em modelos é a criação de um ambiente. Ao oferecer algo pronto a utilizar e pré-configurado especificamente para o ML, permite-te realizar as tuas tarefas rapidamente, passando das ideias num bloco de notas para a produção.

Quando estiveres pronto para implementar um modelo treinado a partir de uma instância de computação, podes utilizar a instância de computação como um cluster de treino, uma vez que suporta uma fila de trabalhos, vários trabalhos em paralelo e treino distribuído multi-GPU. Essa é uma combinação perfeita para depuração e teste, pois o ambiente é reproduzível.

Já alguma vez ouviste dizer "*Mas funciona no meu computador!*"? De certeza que já ouvi! Mesmo que estejas a testar ou a explorar novas ideias, os ambientes reproduutíveis são uma excelente forma de normalizar o desenvolvimento, para que haja menos surpresas e a colaboração entre outros engenheiros seja mais simplificada. A utilização de ambientes

reprodutíveis é uma parte fundamental do DevOps que se aplica ao ML. A consistência, juntamente com a normalização, exige muito esforço para ser bem feita e, sempre que encontrares ferramentas ou serviços imediatamente disponíveis, deves aproveitá-los imediatamente.

Como administrador de sistemas, esforcei-me imenso para normalizar os ambientes de produção para desenvolvimento, e é um problema difícil de resolver. Utiliza as instâncias de computação do Azure tanto quanto possível!

Podes criar uma instância de computação no Azure ML Studio a partir dos vários fluxos de trabalho que requerem uma, como quando crias um Notebook Jupyter. É mais fácil encontrar o link Computação na seção Gerenciar e criar uma instância lá. Depois de carregado, são apresentadas várias opções (consulte a [Figura 8-3](#)). Uma boa regra geral é escolher uma máquina virtual de baixo custo para começar, o que evita custos mais altos no futuro para algo que pode não ser necessário apenas para executar um notebook.

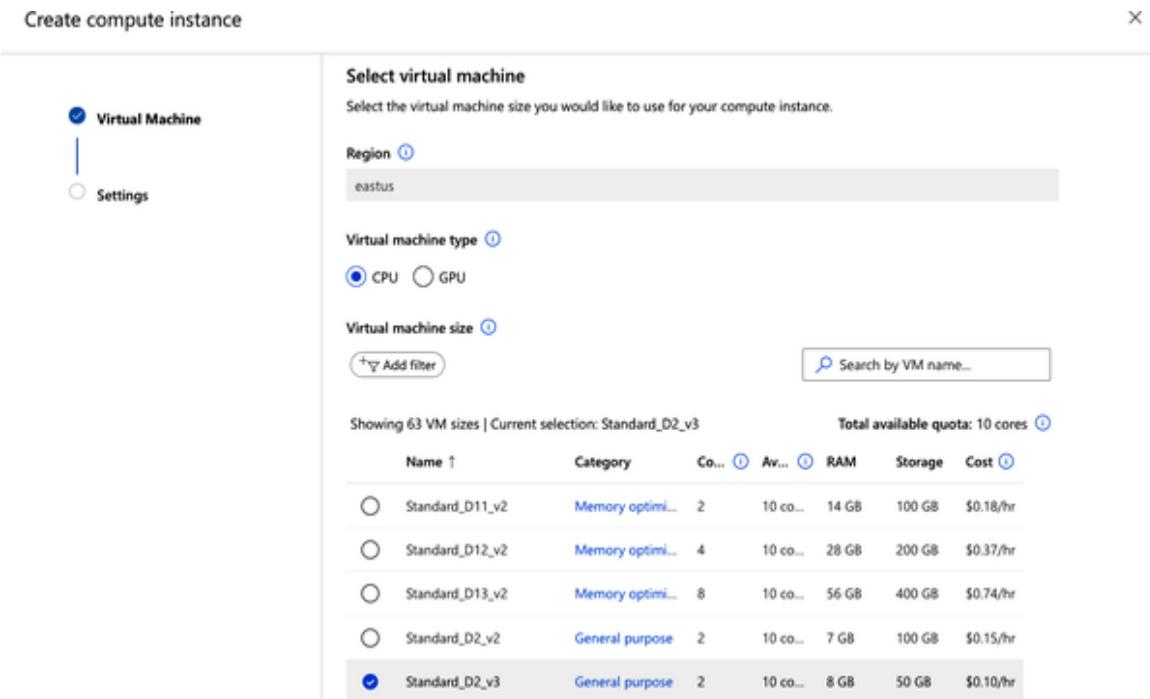


Figura 8-3. Cria uma instância de computação no Azure

Neste exemplo, estou a escolher uma `Standard_D2_v3`. Uma vez que os nomes das várias máquinas oferecidas estão sempre a mudar, as escolhas que te são dadas podem ser diferentes.

Implantação

Existem várias formas de implementar no Azure para interagir com um modelo. Se estiveres a lidar com grandes quantidades de dados para além do que é razoável lidar na memória, a inferência em lote é mais adequada. O Azure fornece muitas ferramentas úteis (a disponibilidade geral deste serviço foi anunciada em 2020), ajudando os utilizadores a lidar com terabytes de dados estruturados e não estruturados e a obter inferências a partir deles.

Outra forma de implementação é para inferência online (por vezes referida como *instantânea*). Ao lidar com conjuntos de dados mais pequenos (não da ordem dos terabytes!), é útil ter um modelo implementado rapidamente, acedido através de uma API HTTP que o Azure pode criar para si programaticamente. Uma API HTTP que é criada para ti automaticamente para um modelo treinado é outra destas funcionalidades que deves aproveitar.

Criar APIs HTTP não é assim tão trabalhoso, mas transferir esse trabalho para um serviço significa que tu (e a tua equipa) têm mais tempo para trabalhar em partes mais substanciais, como a qualidade dos teus dados ou a robustez do processo de implementação.

Registo de modelos

A documentação do Azure descreve o registo como algo opcional. É de facto opcional, na medida em que não precisa dele para implantar um modelo. No entanto, à medida que se habitua ao processo de implementação de modelos e à sua libertação em ambientes de produção, torna-se evidente que poupar em funcionalidades e restrições como a autenticação (ou, neste

caso, o registo de modelos) facilita as coisas no início, mas pode criar problemas mais tarde.

Recomendo vivamente que registe os seus modelos e que siga um processo para o fazer, sendo ainda melhor se este processo for totalmente automatizado. É certo que podes não precisar de registar todos os modelos com que trabalhas, mas deves definitivamente registar os modelos selecionados que entram em produção. Se estiveres familiarizado com o **Git** (**o sistema de controlo de versões**), o controlo de versões dos modelos será uma forma muito natural de pensar nas alterações e modificações dos modelos de nível de produção.

Estes são alguns dos principais aspectos do controlo de versões de modelos que fazem com que seja uma funcionalidade interessante de utilizar:

- Podes identificar a versão do modelo que estás a utilizar
- Podes selecionar rapidamente entre as várias versões, com a clareza das descrições
- Podes voltar atrás e selecionar um modelo diferente sem esforço

Existem algumas formas de registar um modelo. Se estiveres a treinar o modelo no Azure, então podes utilizar o **Python SDK** com o objeto de resultado de uma classe Run:

```
description = "AutoML trained model"
model = run.register_model(description=description)

# A model ID is now accessible
print(run.model_id)
```

Mas não é obrigado a utilizar o Azure para treinar o modelo. Talvez já tenhas vários modelos treinados e estejas a considerar mudar para o Azure para os colocar em produção. O Python SDK também te permite registar estes modelos. Aqui está um exemplo de como fazer isso com um modelo **ONNX** que está disponível localmente no teusistema:

```
import os
from azureml.core.model import Model

# assumes `models/` is a relative directory that contains
# uncompressed
# ONNX models
model = Model.register(
    workspace=ws,
    model_path ="models/world_wines.onnx",
    model_name = "world_wines",
    tags = {"onnx": "world-wines"},
    description = "Image classification of world-wide wine
labels"
)
```

Uma das coisas mais interessantes do Azure é a sua flexibilidade. O Python SDK não é a única forma de registar modelos. Vê aqui como o fazer com o Azure CLI:

```
$ az ml model register --name world_wines --model-path
mnist/model.onnx
```

Podes iterar sobre vários modelos e registá-los no Azure rapidamente com algumas modificações aos exemplos anteriores. Se tiveres modelos disponíveis via HTTP, poderás descarregá-los programaticamente e enviá-los. Utiliza metadados adicionais para preencher as etiquetas e as descrições; boas descrições facilitam a identificação posterior. Quanto mais automatizado for o processo, melhor! Na [Figura 8-4](#), estou a utilizar o Azure ML Studio para carregar e registar diretamente um modelo ONNX, o que também é útil para lidar com um único modelo.

Register a model X

Name *

Description

Model framework *

 *

Framework version *

Model file or folder *

Upload file Upload folder

*

Figura 8-4. Registar um modelo no Azure

Atribuição de versões a conjuntos de dados

À semelhança dos modelos de registo, a capacidade de versionar um conjunto de dados resolve um dos maiores problemas actuais do ML: conjuntos de dados enormes que diferem ligeiramente são difíceis (e, até há pouco tempo, impossíveis) de versionar corretamente. Os sistemas de controlo de versões, como o Git, não são adequados para esta tarefa, embora os sistemas de controlo de versões devam ajudar. Esta incompatibilidade entre os sistemas de controlo de versões que visam as alterações ao código-fonte e os enormes conjuntos de dados tem sido um espinho na produção de modelos de produção fiáveis e reproduzíveis.

Este é outro exemplo de como os fornecedores Cloud, como o Azure, melhoraram os fluxos de trabalho com funcionalidades como o controlo de versões de conjuntos de dados. Os dados são uma das peças mais importantes na elaboração de um pipeline de ML e, uma vez que os dados podem passar por muitas rondas de transformações e limpeza, é fundamental que os versiones ao longo dos passos, desde o bruto ao limpo.

Recupera o conjunto de dados primeiro. Neste exemplo, o conjunto de dados é hospedado por HTTP:

```
from azureml.core import Dataset

csv_url =
("https://automlsamplenotebookdata.blob.core.windows.net"
 "/automl-sample-notebook-data/bankmarketing_train.csv")
dataset = Dataset.Tabular.from_delimited_files(path=csv_url)
```

Em seguida, regista-o utilizando o objeto `dataset`:

```
dataset = dataset.register(
    workspace=workspace,
    name="bankmarketing_dataset",
    description="Bankmarketing training data",
    create_new_version=True)
```

O `create_new_version` definirá incrementalmente uma versão mais recente dos dados, mesmo que não exista uma versão anterior (as versões começam em 1). Depois de registar e criar uma nova versão do conjunto de dados, recupera-o pelo nome e pela versão:

```
from azureml.core import Dataset

# Get a dataset by name and version number
bankmarketing_dataset = Dataset.get_by_name(
    workspace=workspace,
    name="bankmarketing_dataset",
    version=1)
```

NOTA

Embora possa parecer, criar uma nova versão de conjunto de dados *não significa que* o Azure faça uma cópia de todo o conjunto de dados com o espaço de trabalho. Os conjuntos de dados usam referências aos dados no serviço de armazenamento.

Implantação de modelos em um cluster de computação

Nesta seção, configurarás e implantarás um modelo em um cluster de computação. Embora existam algumas etapas envolvidas, é útil repetir o processo algumas vezes para te habituares a elas.

Vai ao [Azure ML Studio](#) e cria uma nova execução de ML automatizada clicando em "New Automated ML run" na secção Automated ML, ou diretamente na página principal (home) clicando na caixa Create New com o menu pendente. Para esta parte do processo, precisas de um conjunto de dados disponível. Se não tiveres registado um conjunto de dados, podes descarregar um e selecionar A partir de um ficheiro local para o registar. Segue os passos para carregá-lo e torná-lo disponível no Azure.

Configuração de um cluster

De volta à secção "Automated ML run", seleciona o conjunto de dados disponível para configurar uma nova execução. Parte da configuração requer um nome e uma descrição significativos. Até este ponto, tudo o que configuraste e disponibilizaste foi o conjunto de dados e como ele deve ser usado e armazenado. Mas um dos componentes críticos de uma estratégia de implantação robusta é garantir que o cluster seja sólido o suficiente para treinar o modelo (veja a [Figura 8-5](#)).

Configure run

Configure the experiment. Select from existing experiments or define a new name, select the target column and the training compute to use. [Learn more on how to configure the experiment](#)

Dataset
bankmarketing_trainig ([View dataset](#))

Experiment name *
 Create new
 New experiment name
 bankmarketing-test-run

Target column * [?](#)
 Column9

Select compute cluster * [?](#)
 Select a compute...

[Create a new compute](#) [Refresh compute](#)

Figura 8-5. Configura uma execução do AutoML

Neste ponto, nenhum cluster deve estar disponível na conta. Selecciona "Criar um novo cálculo" na parte inferior do formulário. A criação de um novo cluster pode ser feita durante a configuração de uma execução para treinar um modelo, ou pode ser feita diretamente na secção Gerir, na ligação Computação. O objetivo final é criar um cluster robusto para treinar o teu modelo.

Vale a pena enfatizar a utilização de nomes e descrições significativos sempre que possível em todos os muitos recursos e produtos no Azure ML. A adição dessas peças informativas é fundamental para capturar (e posteriormente identificar) o que é o recurso subjacente. Uma forma útil de resolver este problema é pensar nestes elementos como campos, tal como quando escreves um e-mail: o *assunto do e-mail* deve captar a ideia geral sobre o que estará no corpo. As descrições são incrivelmente poderosas quando estás a lidar com centenas de modelos (ou mais!). Se fores organizado e claro nas convenções de nomenclatura e nas descrições, vais longe.

Existem dois tipos de clusters importantes: o *Cluster de Inferência* e o *Cluster de Computação*. É fácil sentires-te confuso até conheceres os sistemas subjacentes: os clusters de inferência utilizam **Kubernetes** nos bastidores e os Clusters de computação utilizam máquinas virtuais. Criar ambos no Azure ML Studio é simples. Tudo o que precisas de fazer é

preencher um formulário e ter um cluster a funcionar, pronto para treinar alguns modelos. Após a criação, todos eles (independentemente do tipo) estão disponíveis como uma opção para treinar o teu modelo.

Embora o cluster Kubernetes (Inference) possa ser usado para fins de teste, eu costumo usar um Compute Cluster para tentar estratégias diferentes. Independentemente do backend escolhido, é *crucial* combinar a carga de trabalho com o tamanho (e a quantidade) de máquinas que fazem o trabalho. Por exemplo, num Compute Cluster, tens *de* determinar o número mínimo de nós, bem como o número máximo de nós. Ao paralelizar o treinamento de um modelo, o número de execuções paralelas não pode ser maior do que o número máximo de nós. Determinar corretamente o número de nós, a quantidade de RAM e quantos núcleos de CPU são adequados é mais um fluxo de trabalho de tentativa e erro. Para a execução de teste, é melhor começar com menos e criar clusters com mais potência conforme necessário (consulte a [Figura 8-6](#)).

Create compute cluster ⓘ

Virtual Machine

Settings

Configure Settings

Configure compute cluster settings for your selected virtual machine size.

Name	Category	Cores	Available quota	RAM	Storage	Cost/Hour
Standard_D1	General purpose	1	24 cores	3.5 GB	50 GB	\$0.02/hr

Compute name * ⓘ
test-compute

Minimum number of nodes * ⓘ
0

Maximum number of nodes * ⓘ
2

Idle seconds before scale down * ⓘ
120

Enable SSH access ⓘ

Advanced settings

Name	Category	Cores	Available quota	RAM	Storage	Cost/Hour
Standard_D1	General purpose	1	24 cores	3.5 GB	50 GB	\$0.02/hr

Figura 8-6. Cria um cluster de computação

Escolhi 0 como o número mínimo de nós nesta figura porque evita ser cobrado por nós ociosos. Isso permitirá que o sistema diminua para 0 quando o cluster não estiver a ser utilizado. O tipo de máquina que selecionei não tem muito desempenho, mas como estou a testar a execução,

não tem grande importância; posso sempre voltar atrás e criar um cluster mais recente.

NOTA

Os clusters para *treinar* um modelo não são os mesmos que usarás para implantar um modelo para inferência ao vivo (ou em lote). Isso pode ser confuso porque ambas as estratégias (treinamento e inferência) usam o termo "cluster" para se referir ao grupo de nós que fazem o trabalho.

Implantação de um modelo

Tal como acontece com o treino de um modelo, quando chega a altura de implementar um modelo em produção, tens de estar ciente das opções de cluster disponíveis. Embora existam algumas maneiras de fazer isso, há duas maneiras principais de implantar. Dependendo do tipo de caso de utilização da implementação (produção ou teste), tens de escolher a que melhor se adequa. A seguir, apresentamos uma excelente maneira de pensar sobre esses recursos e onde eles funcionam melhor:

Instância de Contendor do Azure (ACI)

Ideal para testes e ambientes de teste em geral, especialmente se os modelos forem pequenos (menos de 1 GB).

Serviço de Kubernetes do Azure (AKS)

Todas as vantagens do Kubernetes (especialmente o escalonamento) e para modelos com mais de 1 GB.

Ambas as opções são relativamente simples de configurar para uma implementação. No Azure ML Studio, vai a Models (Modelos) na secção Assets (Activos) e seleciona um modelo previamente treinado. Na [Figura 8-7](#), escolhi um modelo ONNX que registei.

Há algumas partes essenciais no formulário; escolhi o ACS como o tipo de computação e habilitei a autenticação. Isso é crucial porque, quando a implantação for concluída, não será possível interagir com o contêiner, a

menos que você use chaves para autenticar a solicitação HTTP. A ativação da autenticação não é necessária, mas é altamente recomendada para manter a paridade entre os ambientes de produção e de teste.

Depois de preencheres o formulário e o submeteres, inicia-se o processo de implementação do modelo. No meu caso de exemplo, activei a autenticação e utilizei o ACS. Estas opções não afectam muito a interação com o modelo, por isso, assim que a implementação estiver concluída, posso começar a interagir com o modelo através de HTTP, garantindo que os pedidos estão a utilizar as chaves.

Deploy a model X

Name * i ✖

mobelinetv-deploy

Description i

Testing out the ONNX model

Compute type * i *

Azure Container Instance

Models: mobilenetv1

Enable authentication

i Keys can be found on the endpoint details page.

Figura 8-7. Implementa um modelo

Podes encontrar todos os detalhes sobre o modelo implementado na secção Pontos finais. O nome usado para o desdobramento é listado, o que leva a um painel de controle dos detalhes do desdobramento. Três guias fazem parte desse painel: Detalhes, Consumir e Logs de implantação. Todas elas estão repletas de informações úteis. Se a implantação for concluída com êxito, os logs provavelmente não serão tão interessantes. Na guia Detalhes,

uma seção mostrará a API HTTP (exibida como "ponto de extremidade REST"). Como eu habilitei a autenticação, a página mostrará o valor de "Key-based authentication" como `true`, conforme mostrado na [Figura 8-8](#).



Figura 8-8. Ponto de extremidade REST

Qualquer coisa que possa falar com um serviço HTTP com a autenticação activada funcionará. Este é um exemplo que utiliza Python (não é necessário o Azure SDK para isto); a entrada está a utilizar **JSON** (JavaScript Object Notation), e a entrada para este modelo específico segue um esquema rigoroso que irá variar dependendo do modelo com o qual estás a interagir. Este exemplo utiliza a biblioteca **Requests** Python:

```
import requests
import json

# URL for the web service
scoring_uri = 'http://676fac5d-5232-adc2-
3032c3.eastus.azurecontainer.io/score'

# If the service is authenticated, set the key or token
key = 'q8szMDoNlxCpiGI8tnqax1yDiy'

# Sample data to score, strictly tied to the input of the trained
model
data = {"data": [
    {
        "age": 47,
        "campaign": 3,
        "contact": "home",
        "day_of_week": "fri",
        "default": "yes",
        "duration": 95,
        "education": "high.school",
        "nr.employed": 4.967,
        "poutcome": "failure",
        "previous": 1
    }
]}  
Content-Type: application/json  
Authorization: Bearer q8szMDoNlxCpiGI8tnqax1yDiy
```

```

        }
    ]
}

# Convert to JSON
input_data = json.dumps(data)

# Set the content type
headers = {'Content-Type': 'application/json'}

# Authentication is enabled, so set the authorization header
headers['Authorization'] = f'Bearer {key}'

# Make the request and display the response
resp = requests.post(scoring_uri, input_data, headers=headers)
print(resp.json())

```

Uma vez que o serviço é exposto com HTTP, permite-me interagir com a API da forma que eu quiser (como utilizando Python no exemplo anterior). Interagir com uma API HTTP para interagir com o modelo é interessante porque oferece uma excelente flexibilidade e obtém resultados imediatos, uma vez que se trata de um serviço de inferência em tempo real. No meu caso, não demorou muito tempo a criar a implementação para obter respostas da API. Isto significa que estou a tirar partido da infraestrutura e dos serviços Cloud para experimentar rapidamente uma prova de conceito que pode acabar em produção. Experimentar modelos com dados de amostra e interagir com eles em ambientes semelhantes aos de produção é fundamental para preparar o caminho para uma automação robusta mais tarde.

Quando tudo corre bem, recebe uma resposta JSON com dados úteis da previsão. Mas o que acontece quando as coisas falham? É útil estar ciente das diferentes formas em que isto pode falhar e o que significam. Neste exemplo, utilizo uma entrada que é inesperada para o modelo ONNX:

```
{
  "error_code": 500,
  "error_message": "ONNX Runtime Status Code: 6. Non-zero status
code returned
while running Conv node. Name:'mobilenetv20_features_conv0_fwd'
```

```
    Missing Input: data\nStacktrace:\n"
}
```

Um código de estado HTTP 500 indica que o serviço teve um erro causado por uma entrada inválida. Certifica-te de que utilizas as chaves e os métodos de autenticação corretos. A maioria dos erros do Azure é fácil de compreender e corrigir; eis alguns exemplos do seu aspeto:

Missing or unknown Content-Type header field in the request

Certifica-te de que o tipo de conteúdo correto (por exemplo, JSON) é utilizado e declarado no pedido.

Method Not Allowed. For HTTP method: GET and request path: /score

Estás a tentar fazer um pedido GET quando provavelmente precisas de um pedido POST que envia dados.

Authorization header is malformed. Header should be in the form: "Authorization: Bearer <token>"

Certifica-te de que o cabeçalho está corretamente construído e que tem um token válido .

Resolução de problemas de implementação

Um dos muitos factores cruciais para um MLOps eficaz (obviamente herdado das melhores práticas DevOps, abordadas em "[DevOps e MLOps](#)") é uma boa capacidade de resolução de problemas. A depuração ou resolução de problemas não é uma competência que nasce contigo; requer prática e perseverança. Uma maneira de explicar essa habilidade é compará-la a andar em uma nova cidade e encontrar o caminho. Alguns dirão que se eu me oriento com (aparente) facilidade, então devo ter alguma habilidade natural. Mas não é bem assim. Presta atenção aos detalhes, revê esses detalhes mentais regularmente e questiona tudo. Nunca presumas.

Quando me oriento, identifico imediatamente a posição do sol ou do mar (é Este? ou Oeste?), e tomo notas mentais de pontos de referência ou edifícios notáveis. Depois, vai do princípio ao fim, revendo mentalmente cada passo: No hotel, fui para a esquerda até à grande igreja, onde virei à direita, atravessei o belo parque e agora estou na praça. A noite instala-se, e depois não há posição do sol, e não me lembro para onde ir. Viraste aqui à esquerda? Ou à direita? Faço perguntas, os meus amigos dizem-me todos que é à esquerda, e eu não assumo que estão certos. Confia mas valida. "*Se me disserem que é à esquerda, isso significa que não teria chegado ao parque, vou na direção deles, espero uns quarteirões e, se não houver parque, volto atrás e viro para o outro lado.*"

Nunca assumas as coisas. Questiona tudo. Presta atenção aos pormenores. Confia, mas verifica. Se praticares estes conselhos, a tua depuração parecerá uma competência natural para os teus colegas. Nesta secção, entro em alguns detalhes relacionados com os contentores e a implementação em contentores no Azure e algumas das coisas que podes ver surgir quando são problemas. Mas podes aplicar os conceitos principais em qualquer lugar.

Recuperar registos

Depois de implantares um contentor, tens algumas formas de recuperar os registos. Podes aceder a eles no Azure ML Studio, bem como através da linha de comandos e do Python SDK. Usando o Python SDK significa apenas algumas linhas após a inicialização do espaço de trabalho:

```
from azureml.core import Workspace
from azureml.core.webservice import Webservice

# requires `config.json` in the current directory
ws = Workspace.from_config()

service = Webservice(ws, "mobelinetv-deploy")
logs = service.get_logs()

for line in logs.split('\n'):
    print(line)
```

Executa esse código de exemplo com o nome de um serviço que já implantou anteriormente e a saída é gerada com muitas informações. Por vezes, os registos não são assim tão bons e são, na sua maioria, repetitivos. Tens de ver através do ruído e apanhar os destaques com informações úteis. Em uma implantação bem-sucedida, a maior parte da saída não terá muito significado.

Estes são alguns dos registos da implementação de um modelo ONNX (os carimbos de data/hora foram removidos por questões de brevidade):

```
WARNING - Warning: Falling back to use azure cli login credentials.  
Version: local_build  
Commit ID: default  
  
[info] Model path: /var/azureml-models/mobilenetv/1/mobilenetv2-  
7.onnx  
[info][onnxruntime inference_session.cc:545 Initialize]:  
Initializing session.  
[onnxruntime inference_session Initialize]: Session successfully  
initialized.  
GRPC Listening at: 0.0.0.0:50051  
Listening at: http://0.0.0.0:8001
```

Informações sobre aplicações

Outro aspecto da recuperação de registos e depuração que é importante explorar é a utilização de ferramentas de observabilidade. A observabilidade refere-se aos meios de capturar o estado do sistema (ou sistemas) em qualquer ponto no tempo. Isso parece um pouco complicado, mas, em resumo, significa que confias em ferramentas como painéis, agregação de registos, gráficos e mecanismos de alerta para visualizar os sistemas como um todo. Uma vez que todo o tema da observabilidade está repleto de ferramentas e processos, é muitas vezes complicado colocar tal coisa em produção.

A observabilidade é crucial porque não se trata de registos de aplicações; trata-se de contar uma história sobre aplicações quando surgem problemas. Claro, encontraste um traceback Python num registo, mas isso não significa

necessariamente que o código Python precisa de ser corrigido. E se a entrada esperada fosse um payload JSON de outro sistema, mas em vez disso, o sistema externo enviou um arquivo vazio? Por que isso aconteceria? A observabilidade traz clareza a sistemas aparentemente caóticos. Os problemas tornam-se exponencialmente mais complicados quando lidas com sistemas distribuídos.

Não é incomum definir um sistema de pipeline para ingerir dados em modelos que envolvem várias etapas diferentes. Obtém os dados, limpa-os, remove colunas de lixo, normaliza-os e versiona este novo conjunto de dados. Supõe que tudo isto está a ser feito com o Python SDK e aproveitando os accionadores do Azure. Por exemplo, novos dados entram no sistema de armazenamento, o que desencadeia uma **Função Azure** que executa algum Python. Nesta cadeia de eventos, é difícil contar uma história sem ferramentas.

O Azure oferece as ferramentas certas prontas a utilizar com a mais simples das chamadas no SDK. Chama-se **Application Insights** e está repleto de todos os gráficos e dashboards úteis logo após a sua ativação. No entanto, não são apenas os registo ou os gráficos agradáveis que tem para oferecer. Um conjunto completo de dados cruciais com uma interface altamente visual está prontamente disponível. Tempos de resposta, taxas de falha e exceções - todos eles são agregados, marcados com data e hora e representados graficamente.

É assim que habilita o Application Insights em um serviço implantado anteriormente:

```
from azureml.core.webservice import Webservice

# requires `ws` previously created with `config.json`
service = Webservice(ws, "mobelinetv-deploy")

service.update(enable_app_insights=True)
```

Quando o Application Insights está ativado, a secção do ponto de extremidade da API no ML Studio irá apresentá-lo como mostrado na **Figura 8-9**.

```
Application Insights enabled  
true
```

```
Application Insights url  
https://portal.azure.com#resource/subscriptions/f65bdeaf-b15f/resourcegroups/examples/providers/microsoft.insights/components/06308
```

Figura 8-9. Application Insights ativado

Segue a ligação fornecida para o painel de controlo do serviço. Uma infinidade de vários gráficos e fontes de informações está disponível. Esta imagem captura uma pequena parte dos gráficos disponíveis para solicitações feitas ao modelo implantado em um contêiner, como mostrado na Figura 8-10.

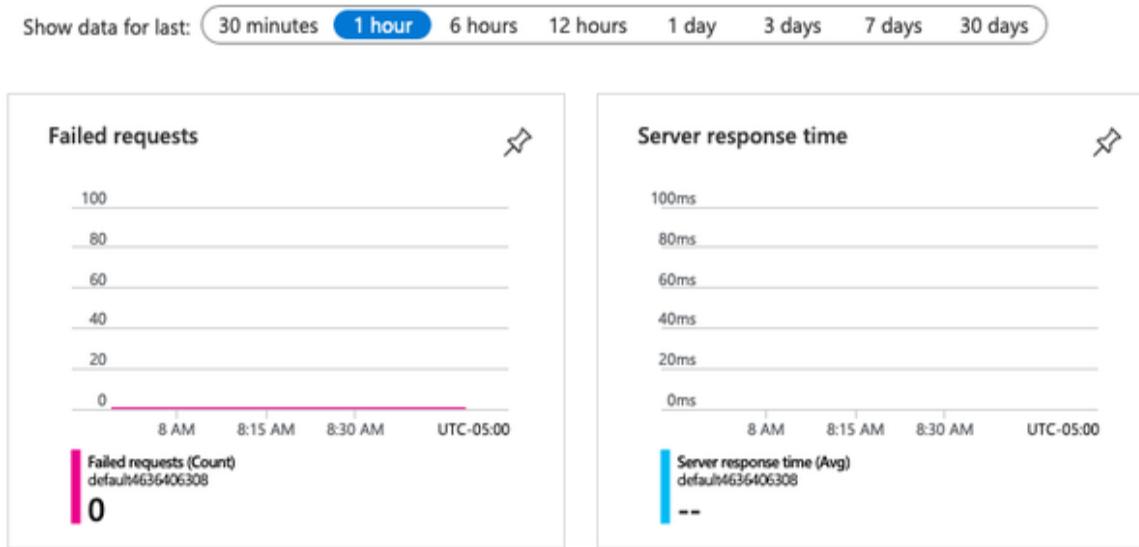


Figura 8-10. Informações sobre a aplicação

Depurando localmente

Seguindo os conselhos dos princípios DevOps, ao depurar, deves questionar tudo e nunca assumir nada. Uma técnica que é útil para depurar problemas é executar um serviço de produção ou, neste caso, um modelo treinado, em ambientes diferentes. As implantações em contêineres e os contêineres em geral oferecem essa flexibilidade, onde algo que é executado em produção

no Azure pode ser executado localmente. Ao executar um contentor localmente, que é o mesmo contentor que é executado em produção com problemas, podes fazer muito mais coisas para além de ver os registos. A depuração local (potencialmente na tua máquina) é um recurso tremendo para compreender e tirar partido, evitando interrupções de serviço ou interrupções de serviço catastróficas. Já estive em várias situações em que a única opção era "iniciar sessão no servidor Web de produção para ver o que se passa". Isto é perigoso e altamente problemático.

Ao ser cético em relação a questões e problemas, é crucial tentar replicar o problema. Reproduzir a questão é o bilhete de ouro para resolver problemas. Por vezes, para reproduzir produtos orientados para o cliente, reinstalei sistemas operativos de raiz e fiz implementações em ambientes separados. Os programadores usam, por brincadeira, a frase "*funciona na minha máquina*", e aposto contigo que quase sempre é verdade - mas, na realidade, não significa nada. O conselho de "*questionar tudo*" pode ser aplicado aqui: implementa várias vezes, em diferentes ambientes, incluindo localmente, e tenta replicar.

Embora não seja razoável esperar que tenhas de executar localmente a oferta Kubernetes do Azure, o Python SDK oferece facilidades para expor um serviço web (local) onde podes implementar o mesmo modelo de grau de produção, num contentor, localmente. Há algumas vantagens nesta abordagem que já mencionei. Ainda assim, há outra crucial: a maioria das APIs do SDK do Python vão funcionar com esta implementação local , *para além de* todos os comandos de ferramentas de contentores que podes executar para mexer no contentor em tempo de execução. Operações como obter os logs do container, ou entrar no container para inspecionar o ambiente, são todas possíveis e sem problemas.

NOTA

Uma vez que estas operações estão a lidar com implementações em contentores, é necessário ter **Docker** instalado e em execução no teu ambiente.

Existem algumas etapas necessárias para executar um serviço localmente. Primeiro, tens de registar o modelo no diretório atual:

```
from azureml.core.model import Model

model = Model.register(
    model_path="roberta-base-11.onnx",
    model_name="roberta-base",
    description="Transformer-based language model for text
generation.",
    workspace=ws)
```

Em seguida, cria um ambiente, que instala todas as dependências necessárias para que o modelo funcione dentro do contentor. Por exemplo, se o tempo de execução do ONNX for necessário, tens de o definir:

```
from azureml.core.environment import Environment

environment = Environment("LocalDeploy")
environment.python.conda_dependencies.add_pip_package("onnx")
```

Um arquivo de pontuação (geralmente chamado *score.py*) é necessário para implantar o modelo. Esse script é responsável por carregar o modelo, definir as entradas para esse modelo e pontuar os dados. Os scripts de pontuação são sempre específicos para o modelo, e não há uma maneira genérica de escrever um para qualquer modelo. O script requer duas funções: *init()* e *run()*. Agora, cria uma *Configuração de inferência*, que reúne o script de avaliação e o ambiente:

```
from azureml.core.model import InferenceConfig

inference_config = InferenceConfig(
    entry_script="score.py",
    environment=environment)
```

Agora, para juntar tudo isto, tens de usar a classe *LocalWebservice* do Python SDK para lançar o modelo num contentor local:

```

from azureml.core.model import InferenceConfig, Model
from azureml.core.webservice import LocalWebservice

# inference with previously created environment
inference_config = InferenceConfig(entry_script="score.py",
environment=myenv)

# Create the config, assigning port 9000 for the HTTP API
deployment_config =
LocalWebservice.deploy_configuration(port=9000)

# Deploy the service
service = Model.deploy(
    ws, "roberta-base",
    [model], inference_config,
    deployment_config)

service.wait_for_deployment(True)

```

O lançamento do modelo usará um contêiner nos bastidores que executará a API HTTP exposta na porta **9000**. Não só podes enviar pedidos HTTP diretamente para `localhost:9000`, como também é possível aceder ao contentor em tempo de execução. O meu tempo de execução do contentor não tinha o contentor pronto no meu sistema, mas a execução do código para implementar localmente retirou tudo do Azure:

```

Downloading model roberta-base:1 to
/var/folders/pz/T/azureml5b/roberta-base/1
Generating Docker build context.
[...]
Successfully built 0e8ee154c006
Successfully tagged mymodel:latest
Container (name:determined_ardinghelli,
id:d298d569f2e06d10c7a3df505e5f30afc21710a87b39bdd6f54761) cannot
be killed.
Container has been successfully cleaned up.
Image sha256:95682dcea5527a045bb283cf4de9d8b4e64deaf60120
successfully removed.
Starting Docker container...
Docker container running.
Checking container health...
Local webservice is running at http://localhost:9000
9000

```

Agora que a implantação foi concluída, posso verificá-la executando docker:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
2b2176d66877        mymodel             "runsvdir /var/runit"
PORTS
8888/tcp, 127.0.0.1:9000->5001/tcp, 127.0.0.1:32770->8883/tcp
```

Entro no container e posso verificar que o meu script *score.py* está lá, juntamente com o modelo:

```
root@2b2176d66877:/var/azureml-app# find /var/azureml-app
/var/azureml-app/
/var/azureml-app/score.py
/var/azureml-app/azureml-models
/var/azureml-app/azureml-models/roberta-base
/var/azureml-app/azureml-models/roberta-base/1
/var/azureml-app/azureml-models/roberta-base/1/roberta-base-
11.onnx
/var/azureml-app/main.py
/var/azureml-app/model_config_map.json
```

Ao tentar implementar, tive alguns problemas com o script *score.py*. O processo de implantação imediatamente apresentou erros e teve algumas sugestões:

```
Encountered Exception Traceback (most recent call last):
  File "/var/azureml-server/aml_blueprint.py", line 163, in
register
    main.init()
AttributeError: module 'main' has no attribute 'init'

Worker exiting (pid: 41)
Shutting down: Master
Reason: Worker failed to boot.
2020-11-19T23:58:11,811467402+00:00 - gunicorn/finish 3 0
2020-11-19T23:58:11,812968539+00:00 - Exit code 3 is not normal.
Killing image.

ERROR - Error: Container has crashed. Did your init method fail?
```

A função `init()`, neste caso, tem de aceitar um argumento, e o meu exemplo não o exigia. Depurar localmente e mexer num contentor implantado localmente com um modelo é muito útil e uma excelente forma de iterar rapidamente diferentes definições e alterações ao modelo antes de o experimentar no próprio Azure.

Pipelines de ML do Azure

Os pipelines não são mais do que vários passos para atingir o objetivo desejado. Se algumavez trabalhou com uma plataforma de integração contínua (CI) ou de entrega contínua (CD) como o **Jenkins**, então qualquer fluxo de trabalho de "pipeline" parecerá familiar. O Azure descreve os seus pipelines de ML como um bom ajuste para três cenários distintos: aprendizagem automática, preparação de dados e orquestração de aplicações. Estes cenários têm definições e configurações semelhantes, embora trabalhem com diferentes fontes de informação e objectivos para a sua conclusão.

Tal como acontece com a maioria das ofertas do Azure, podes utilizar o Python SDK ou o Azure ML Studio para criar pipelines. Como já referi, os pipelines são *passos* para atingir um objetivo, e cabe-te a ti ordenar esses passos para obteres o resultado final. Por exemplo, um pipeline pode ter que lidar com dados; já abordamos conjuntos de dados neste capítulo, portanto, recupera um conjunto de dados existente para que uma etapa do pipeline possa ser criada. Neste exemplo, o conjunto de dados torna-se a entrada de um script Python, formando um *passo* distinto *do pipeline*:

```
from azureml.pipeline.steps import PythonScriptStep
from azureml.pipeline.core import PipelineData
from azureml.core import Datastore

# bankmarketing_dataset already retrieved with `get_by_name()`
# make it an input to the script step
dataset_input = bankmarketing_dataset.as_named_input("input")

# set the output for the pipeline
output = PipelineData(
```

```

    "output",
    datastore=Datastore(ws, "workspaceblobstore"),
    output_name="output")

prep_step = PythonScriptStep(
    script_name="prep.py",
    source_directory=".src",
    arguments=["--input", dataset_input.as_download(), "--"
output", output],
    inputs=[dataset_input],
    outputs=[output],
    allow_reuse=True
)

```

NOTA

O SDK do Azure pode mudar frequentemente, por isso não te esqueças de consultar a [documentação oficial do Microsoft AzureML](#).

O exemplo usa o `PythonScriptStep`, uma das muitas etapas diferentes disponíveis como uma etapa de pipeline. Lembra-te: os pipelines têm tudo a ver com etapas que trabalham para um objetivo, e o Azure fornece diferentes etapas para diferentes tipos de trabalho no SDK e no Azure ML Studio. No entanto, esta etapa está faltando uma parte importante: o destino de computação. Mas já tem quase tudo o que é necessário para fazer a preparação de dados. Primeiro, usa o objeto de conjunto de dados e chama `as_named_input`, que o `PythonScriptStep` usa como um argumento. O passo do script é uma classe Python, mas tenta representar uma ferramenta de linha de comandos, por isso os argumentos usam traços e os valores para esses argumentos são passados como itens numa lista. É assim que recuperas um destino de cálculo criado anteriormente com o SDK:

```

from azureml.core.compute import ComputeTarget, AmlCompute
from azureml.core import Workspace

ws = Workspace.from_config()

# retrieve the compute target by its name, here a previously

```

```
created target
# is called "mlops-target"
compute_target = ws.compute_targets["mlops-target"]
```

Para além do destino de computação, que já abordámos neste capítulo, podes opcionalmente fornecer uma *configuração de tempo de execução*, que permite definir variáveis de ambiente para dizer ao Azure como gerir o ambiente. Por exemplo, se quiseres gerir as tuas dependências em vez de teres o Azure a tratar disso por ti, então a configuração de tempo de execução seria a forma de o fazer. Aqui está uma maneira simplificada de definir essa opção específica:

```
from azureml.core.runconfig import RunConfiguration

run_config = RunConfiguration()

# a compute target should be defined already, set it in the config:
run_config.target = compute_target

# Disable managed dependencies
run_config.environment.python.user_managed_dependencies = False
```

Publicar pipelines

Já fiz anteriormente a comparação de sistemas de integração contínua como o Jenkins com pipelines: muitos passos a trabalhar de forma coordenada para atingir um objetivo. Mas embora outros sistemas de CI/CD como o Jenkins tenham esta capacidade, uma coisa que é muito difícil de conseguir é expor estes trabalhos fora do ambiente. O Azure tem uma forma simples de o conseguir, tanto através do Azure ML Studio como com o SDK. Essencialmente, o que acontece com o pipeline é que ele fica disponível por HTTP para que qualquer sistema em qualquer lugar do mundo possa acessar o pipeline e acioná-lo.

As possibilidades são infinitas. Já não está vinculado à utilização de serviços, pipelines e accionadores a partir do Azure. Ainda assim, as etapas do pipeline podem começar em outro lugar, talvez em um ambiente fechado no local da sua empresa ou em um serviço de código-fonte público como o

GitHub. Esta é uma flexibilidade interessante porque oferece mais opções, e as restrições do fornecedor de Cloud desaparecem. Não precisas de criar um novo pipeline sempre que o quiseres publicar. Podes encontrar isto na documentação quando tentas descobrir como publicar um pipeline. Neste exemplo, um experimento e uma execução anteriores são recuperados para publicar o pipeline:

```
from azureml.core.experiment import Experiment
from azureml.pipeline.core import PipelineRun

experiment = Experiment(ws, "practical-ml-experiment-1")

# run IDs are unique, this one already exists
run_id = "78e729c3-4746-417f-ad9a-abe970f4966f"
pipeline_run = PipelineRun(experiment, run_id)

published_pipeline = pipeline_run.publish_pipeline(
    name="ONNX example Pipeline",
    description="ONNX Public pipeline", version="1.0")
```

Agora que sabes como publicá-la, podes interagir com ela através de HTTP. Estes pontos de extremidade da API requerem autenticação, mas o SDK tem tudo o que precisas para obter os cabeçalhos de autenticação necessários para fazer pedidos:

```
from azureml.core.authentication import
InteractiveLoginAuthentication
import requests

interactive_auth = InteractiveLoginAuthentication()
auth_header = interactive_auth.get_authentication_header()

rest_endpoint = published_pipeline.endpoint
response = requests.post(
    rest_endpoint,
    headers=auth_header,
    json={"ExperimentName": "practical-ml-experiment-1"}
)

run_id = response.json().get('Id')
print(f"Pipeline run submitted with ID: {run_id}")
```

Designer de aprendizagem automática do Azure

Para os que gostam de gráficos, o designer de Aprendizagem automática do Azure é uma boa opção para abstrair a complexidade da criação de projectos de aprendizagem automática no Azure. O processo para treinar um modelo é o seguinte:

1. Inicia sessão no Azure ML Studio.
2. Selecciona a interface Designer, como mostra a Figura 8-11.

Welcome to the Azure Machine Learning Studio

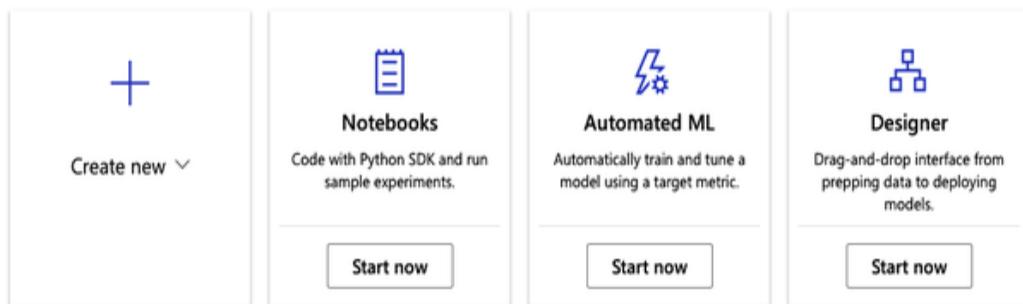


Figura 8-11. Designer do Azure ML

3. Selecciona um projeto de amostra para explorar, como o projeto Automobile Regression na Figura 8-12. Nota que existem muitos projectos de amostra para explorar, ou podes criar o teu projeto de ML a partir do zero. Um excelente recurso para investigar projectos de amostra do ML Designer é a [documentação oficial do designer de Aprendizagem automática do Microsoft Azure](#).

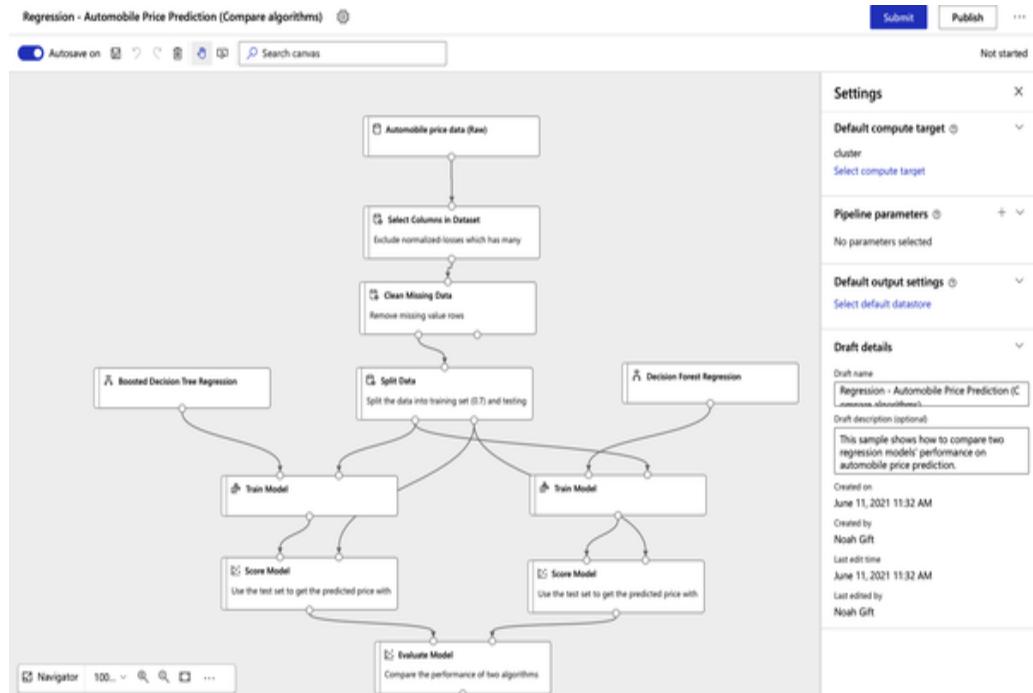


Figura 8-12. Projeto de regressão automóvel do Azure ML Designer

4. Para executar o projeto, submete uma tarefa de pipeline, conforme apresentado na **Figura 8-13**.

Set up pipeline run

Experiment

Select existing Create new

Existing experiment *

Run description *

Compute target

Default cluster

Submit **Cancel**

Figura 8-13. Submissão do designer do Azure ML

O designer do Azure ML pode parecer um pouco extravagante, mas pode desempenhar um papel essencial na compreensão do funcionamento do ecossistema do Azure ML Studio. Ao "experimentar" um projeto de amostra, fica exposto a todos os aspectos essenciais do Azure ML Studio, incluindo AutoML, armazenamento, clusters de computação e relatórios. Em seguida, vamos falar sobre como tudo isto se relaciona com o ciclo de vida do ML no Azure.

Ciclo de vida do ML

No final, todas as ferramentas e serviços no Azure estão lá para ajudar no ciclo de vida do modelo. Esta metodologia não é totalmente específica do Azure, mas é útil compreender como estes serviços ajudam a colocar os modelos em produção. Como mostra a [Figura 8-14](#), começa com o treinamento, que pode ocorrer em um Notebook, AutoML ou com o SDK. Em seguida, pode passar para a validação com o Azure Designer ou o próprio Azure ML Studio. As implantações de produção podem então aproveitar o dimensionamento com Kubernetes enquanto mantém a atenção aos problemas com o Application Insights.

A [Figura 8-14](#) tenta deixar claro que esse não é um processo linear, e o loop de feedback constante durante todo o processo de envio para a produção pode exigir o retorno às etapas anteriores para resolver problemas de dados e outros problemas comuns observados nos modelos. No entanto, o ciclo de feedback e o ajuste constante são essenciais para um ambiente saudável; não basta clicar em uma caixa de seleção que permite o monitoramento ou que o Kubernetes está lidando com o dimensionamento. Sem uma avaliação consistente, o sucesso é impossível, independentemente do fornecedor de Cloud.

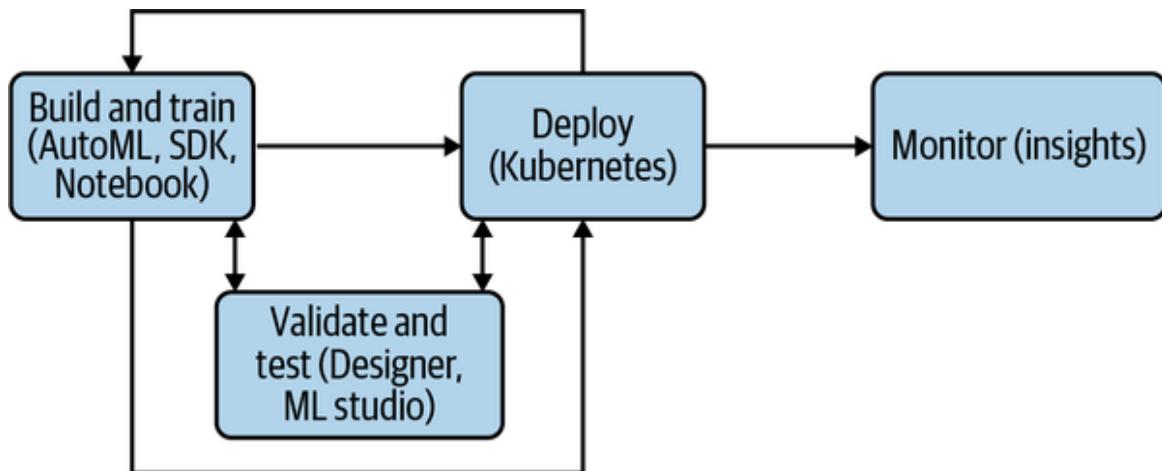


Figura 8-14. Ciclo de vida do ML

Conclusão

Não há dúvida de que o Azure já está a resolver problemas desafiantes relacionados com a operacionalização da aprendizagem automática, desde o registo e o controlo de versões de conjuntos de dados até à facilitação da monitorização e implementação de modelos de inferência em tempo real em clusters escaláveis. Tudo isto parece relativamente novo e o Azure como um todo ainda está a recuperar o atraso em termos de funcionalidade em relação a outros fornecedores de Cloud - mas isso não deve importar. A plataforma escolhida (mesmo que não seja o Azure) tem de permitir fluxos de trabalho com facilidade; tens de aproveitar o que está disponível. Verás a repetição e a ênfase de algumas ideias ao longo do livro sobre como aproveitar a tecnologia e evitar resolver desafios com soluções incompletas. A reutilização da tecnologia irá impulsionar qualquer empreendimento. Lembra-te que a coisa mais importante a conseguir como engenheiro MLOps é enviar modelos para produção, não reinventar funcionalidades Cloud.

No próximo capítulo, vais analisar a Google Cloud Platform.

Exercícios

- Recupera um modelo ONNX de uma fonte pública e regista-o no Azure com o Python SDK.
- Implementa um modelo na ACI e cria um script Python que devolve a resposta do modelo, à medida que esta regressa da API HTTP.
- Implementa um contentor localmente, utilizando o Python SDK do Azure, e produz alguns pedidos HTTP para inferências em tempo real.
- Publica um novo pipeline e, em seguida, aciona-o. O acionador deve mostrar a saída do `run_id` após uma solicitação bem-sucedida.
- Treina um modelo utilizando o Azure AutoML a partir do Python SDK, obtendo um conjunto de dados do Kaggle.

Questões para discussão sobre pensamento crítico

- Há muitas maneiras de treinar modelos na plataforma Azure: Azure ML Studio Designer, Azure Python SDK, Azure Notebooks e Azure AutoML. Quais são as vantagens e desvantagens de cada um?
- Porque é que é uma boa ideia ativar a autenticação?
- Como é que os ambientes reprodutíveis podem ajudar a fornecer modelos?
- Descreve dois aspectos de boas técnicas de depuração e porque são úteis.
- Quais são as vantagens dos modelos de controlo de versões?

- Porque é que o controlo de versões dos conjuntos de dados é importante?

Capítulo 9. MLOps para GCP

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

Os melhores professores de bonsai têm um domínio da realidade e uma capacidade não só de explicar mas também de inspirar. O John disse uma vez: "Jina esta parte e prende-a com arame para que seque com uma boa forma". "Que forma?", perguntei-te. Perguntei-te. "Tu é que decides", respondeu ele, "não me cabe a mim cantar a tua canção por ti!"

—Dr. Joseph Bogen

A Google Cloud Platform (GCP) é única em comparação com os seus concorrentes. Por um lado, tem sido marginalmente focada nas empresas; por outro, tem pesquisa e desenvolvimento de classe mundial que criou tecnologia líder na categoria, incluindo produtos como Kubernetes e TensorFlow. No entanto, mais um aspeto único do Google Cloud é a rica coleção de recursos educativos disponíveis para estudantes e profissionais através de <https://edu.google.com>.

Vamos mergulhar no Google Cloud com ênfase na sua utilização para realizar MLOps.

Visão geral do Google Cloud Platform

Todas as plataformas Cloud têm prós e contras, por isso vamos começar por abordar os três principais contras da Google Cloud Platform. Primeiro, com o Google atrás da AWS e da Microsoft Azure, uma desvantagem de usar o Google é que ele tem menos profissionais certificados. Na [Figura 9-1](#),

podes ver que, em 2020, a AWS e o Azure controlavam mais de 50% do mercado e o Google Cloud tinha menos de 9%. Como resultado, a contratação de talentos para o Google Cloud Platform é mais desafiadora.

Uma segunda desvantagem é o facto de a Google fazer parte daquilo a que a professora de Harvard **Shoshana Zuboff** chama capitalismo de vigilância, em que "Silicon Valley e outras empresas estão a extrair informações dos utilizadores para prever e moldar o seu comportamento". Assim, é teoricamente possível que a regulamentação tecnológica possa ter impacto na quota de mercado no futuro.

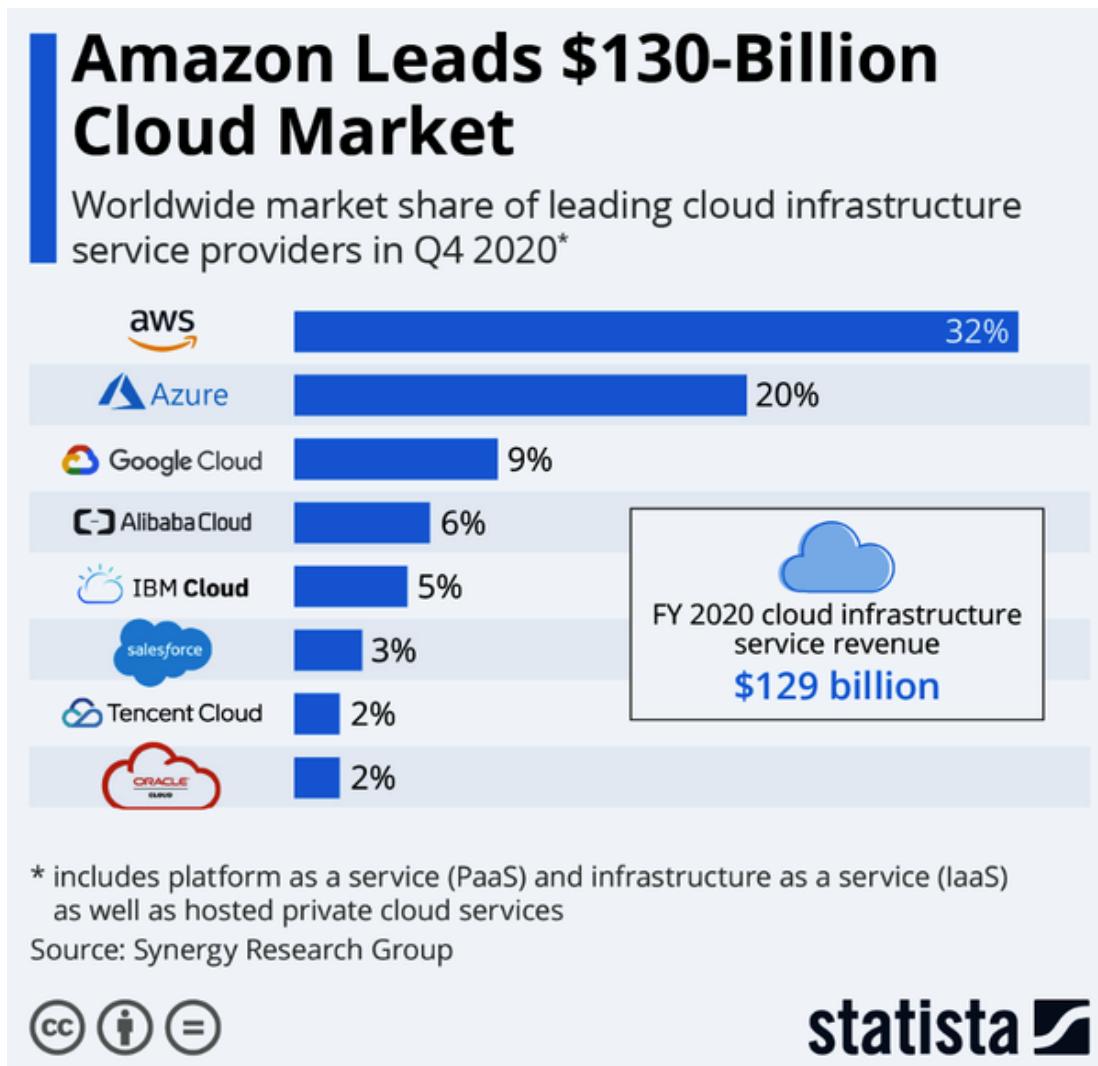


Figura 9-1. Quota de mercado do GCP Cloud

Por último, a Google tem a reputação de ter uma má experiência de utilizador e de cliente e abandona frequentemente produtos como o Google Hangouts e a rede social Google Plus. Poderá então descontinuar o Google Cloud se este continuar a ser a terceira melhor opção nos próximos cinco anos?

Embora se trate de desafios substanciais, e a Google seria sensata se resolvesse rapidamente as questões culturais que levaram a estes contras, existem muitas vantagens únicas na plataforma Google devido à sua cultura. Por exemplo, enquanto a AWS e a Microsoft são culturas orientadas para o serviço de apoio ao cliente com uma história rica de apoio ao cliente empresarial, a Google não tinha apoio telefónico para a maioria dos produtos. Em vez disso, a sua cultura centra-se em entrevistas intensas ao estilo "leet code" para "contratar apenas os melhores". Além disso, a pesquisa e o desenvolvimento de soluções complexas e entorpecedoras que funcionam à "escala do planeta" é algo que faz bem. Em particular, três dos projectos de código aberto mais bem sucedidos da Google demonstram esta força cultural: Kubernetes, a linguagem Go e a estrutura de aprendizagem profunda TensorFlow.

Em última análise, a principal vantagem de utilizar o Google Cloud pode ser o facto de a sua tecnologia ser ideal para uma estratégia multicloud. Tecnologias como Kubernetes e Tensor Flow funcionam bem em qualquer Cloud e são amplamente adoptadas. Como resultado, a utilização do Google Cloud pode ser uma cobertura para as grandes empresas que pretendem verificar o poder da sua relação de fornecedor com a AWS ou o Azure. Além disso, essas tecnologias são amplamente adotadas, por isso é relativamente simples contratar para cargos que exigem experiência em TensorFlow.

Vamos dar uma vista de olhos às principais ofertas do Google Cloud. Esses serviços se dividem em quatro categorias claras: Computação, Armazenamento, Big Data e Aprendizado de máquina, conforme mostrado na [Figura 9-2](#).

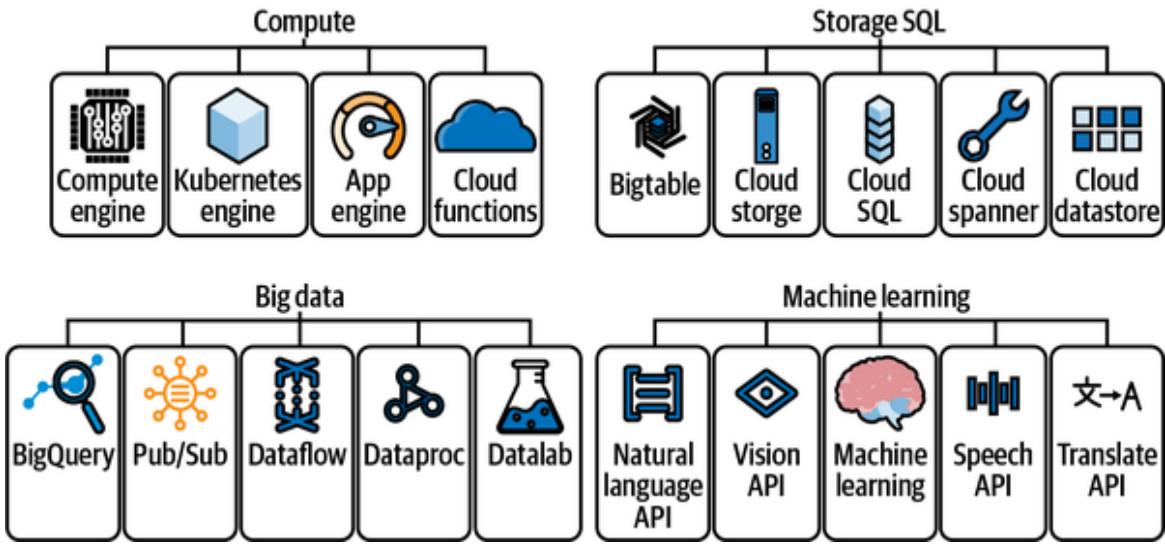


Figura 9-2. Serviços do GCP Cloud

De seguida, vamos definir os principais componentes do Google Cloud, começando pelo Compute:

Motor de computação

Tal como outros fornecedores de Cloud (nomeadamente a AWS e o Azure), o GCP oferece máquinas virtuais como um serviço. O Compute Engine é um serviço que te permite criar e executar máquinas virtuais na infraestrutura do Google. Talvez a conclusão mais importante seja que existem muitas máquinas virtuais diferentes, incluindo Compute Intensive, Memory Intensive, Accelerator Optimized e General Purpose. Além disso, existem VMs preemptivas, disponíveis por até 24 horas, adequadas para trabalhos em lote e que podem economizar até 80% nos custos de armazenamento.

Como profissional de MLOps, é fundamental utilizar os tipos de máquinas adequados para a tarefa em questão. Os custos são importantes no mundo real, e a capacidade de prever com precisão os custos pode fazer a diferença para uma empresa que faz aprendizado de máquina. Por exemplo, com o Deep Learning, pode ser ideal utilizar instâncias Accelerator Optimized, uma vez que podem aproveitar as capacidades adicionais massivamente paralelas das GPUs NVIDIA. Por outro lado, seria um desperdício incrível usar essas instâncias para treinamento de aprendizado de máquina que não pode tirar proveito das

GPUs. Da mesma forma, ao arquitetar em torno de VMs preemptivas para aprendizado de máquina em lote, uma organização poderia economizar até 80% do custo.

Motor Kubernetes e Cloud Run

Uma vez que a Google criou o Kubernetes e o mantém, o suporte para trabalhar no Kubernetes é excelente através do seu GKE (Google Kubernetes Engine). Em alternativa, o Cloud Run é um serviço de alto nível que abstrai muitas das complexidades da execução de contêineres. O Cloud Run é um bom ponto de partida para o Google Cloud Platform para organizações que desejam uma maneira simples de implantar aplicativos de aprendizado de máquina em contêineres.

Motor de aplicações

O Google App Engine é uma PaaS totalmente gerida. Podes escrever código em várias linguagens, incluindo Node.js, Java, Ruby, C#, Go, Python ou PHP. Os fluxos de trabalho do MLOps podem usar o App Engine como ponto de extremidade da API de um pipeline de entrega contínua totalmente automatizado usando o GCP Cloud Build para implantar alterações.

Funções Cloud

As Google Cloud Functions funcionam como FaaS (funções como serviço). O FaaS funciona bem com uma arquitetura orientada por eventos. Por exemplo, o Cloud Functions pode acionar um trabalho de treinamento de aprendizado de máquina em lote ou fornecer uma previsão de ML em resposta a um evento.

De seguida, vamos falar de armazenamento no Google Cloud. No que diz respeito aos MLOps, a principal opção a discutir é o seu produto Cloud Storage. Oferece armazenamento ilimitado, acessibilidade mundial, baixa latência, redundância geográfica e elevada durabilidade. Estes factos significam que, para os fluxos de trabalho MLOps, o lago de dados é o local

onde residem os dados não estruturados e estruturados para o processamento em lote dos trabalhos de formação de aprendizagem automática.

Estreitamente associadas a esta oferta estão as ferramentas de megadados disponíveis no GCP. Muitas ofertas ajudam a mover, consultar e computar grandes volumes de dados. Uma das ofertas mais populares é o Google BigQuery, porque oferece uma interface SQL, um paradigma sem servidor e a capacidade de fazer aprendizagem automática na plataforma. O Google BigQuery é um ótimo lugar para começar a fazer aprendizado de máquina no GCP porque podes resolver toda a cadeia de valor do MLOps a partir desta única ferramenta.

Por fim, coordena as capacidades de aprendizagem automática e de IA num produto chamado IA vértice. Uma vantagem da abordagem da Google é que pretende ser uma solução MLOps desde o início. O fluxo de trabalho do vértice AI permite uma abordagem estruturada ao ML, incluindo o seguinte:

- Criação e armazenamento de conjuntos de dados
- Treinar um modelo ML
- Armazena o modelo no vértice AI
- Implementar o modelo num ponto final para previsão
- Testar e criar pedidos de previsão
- Utilizar a divisão de tráfego para pontos finais
- Gerir o ciclo de vida dos modelos e pontos finais de ML

De acordo com a Google, estas capacidades são importantes para a forma como a IA vértice permite os MLOps, como mostra a [Figura 9-3](#). No centro desses sete componentes está o gerenciamento de dados e modelos, o elemento central do MLOps.

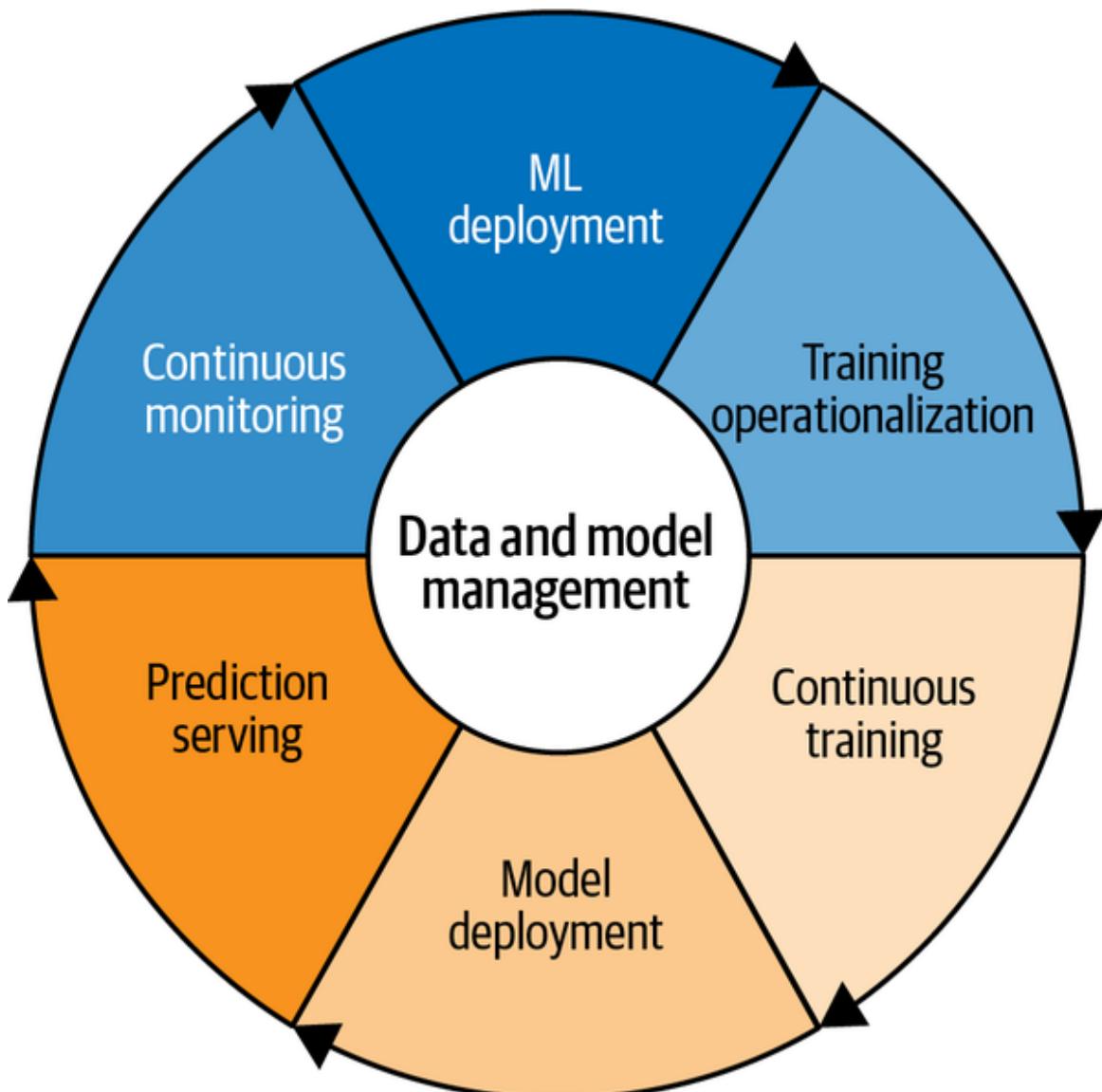


Figura 9-3. Os sete componentes dos MLOps da Google

Este processo de pensamento culmina na ideia da Google de MLOps de ponta a ponta, conforme descrito na [Figura 9-4](#). Uma plataforma abrangente como a Vértice AI permite uma forma abrangente de gerir os MLOps.

Em suma, o MLOps no Google Cloud Platform é simples devido ao Vértice AI e aos subcomponentes deste sistema, como o Google BigQuery. Em seguida, vamos entrar em mais detalhes sobre CI/CD no GCP, um componente fundamental não opcional do MLOps.

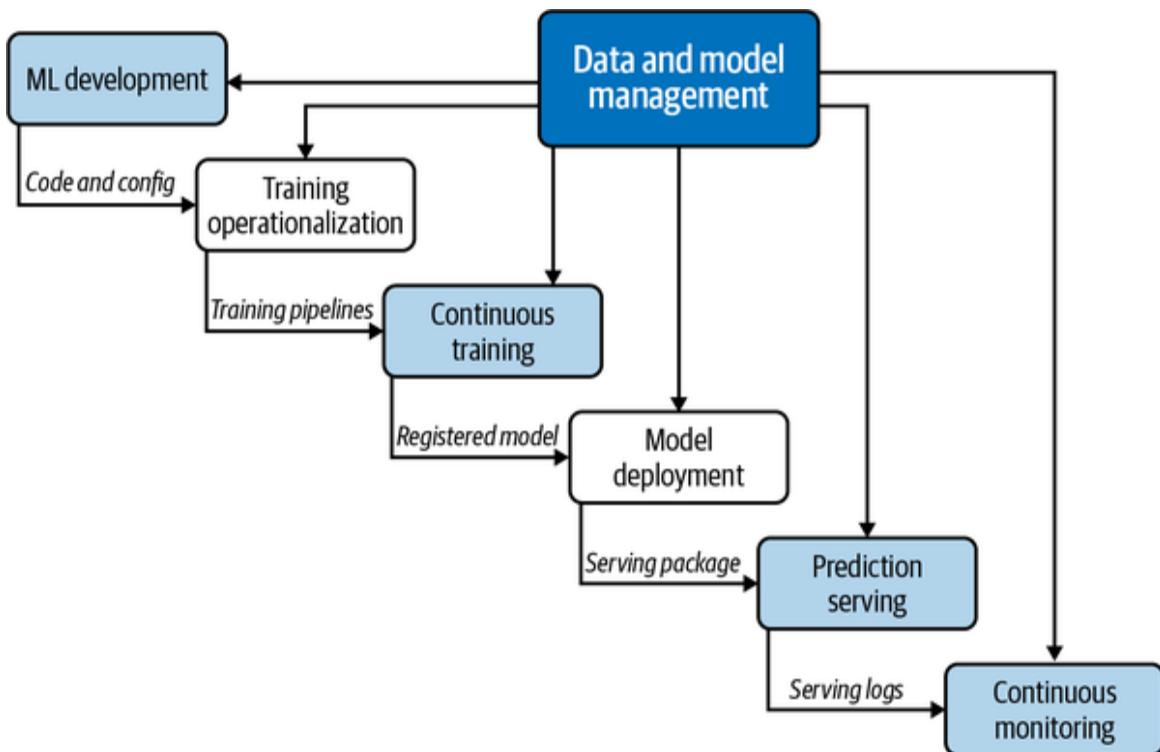


Figura 9-4. MLOps de ponta a ponta no GCP

Integração Contínua e Entrega Contínua

Uma das áreas mais importantes e ainda assim negligenciadas de um projeto envolve a integração contínua. O teste é um componente fundamental para fazer tanto DevOps quanto MLOps. Para o GCP, há duas opções principais de integração contínua: usa uma oferta SaaS como o GitHub Actions ou usa a solução nativa da Cloud, o [Cloud Build](#). Vamos dar uma olhada nas duas opções. Podes ver um scaffold de projeto inicial completo neste [repositório do GitHub gcp-from-zero](#).

Primeiro, vamos dar uma olhada no Google Cloud Build. Aqui está um exemplo do arquivo de configuração do Google Cloud Build, [`cloudbuild.yaml`](#):

```

steps:
- name: python:3.7
  id: INSTALL
  entrypoint: python3
  args:
  - '-m'

```

```

- 'pip'
- 'install'
- '-t'
- '.'
- '-r'
- 'requirements.txt'
- name: python:3.7
  entrypoint: ./pylint_runner
  id: LINT
  waitFor:
  - INSTALL
- name: "gcr.io/cloud-builders/gcloud"
  args: ["app", "deploy"]
timeout: "1600s"
images: ['gcr.io/$PROJECT_ID/pylint']

```

Uma forma recomendada de trabalhar com o Google Cloud é usar o editor incorporado juntamente com o terminal, conforme mostrado na [Figura 9-5](#). Observa que um ambiente virtual Python está ativado.



The screenshot shows the Google Cloud Platform Cloud Shell interface. At the top, there's a menu bar with 'Terminal' and 'Help'. Below it is a tab bar with 'Makefile' (which is currently active) and 'Cloud Build'. The main area displays a 'Makefile' with the following content:

```

1 install:
2     pip install --upgrade pip && \
3         pip install -r requirements.txt
4
5 test:
6     pytest -vv --cov-report term-missing --cov=app test_*.py
7
8 format:
9     black *.py
10
11 lint:
12     pylint --disable=R,C app.py
13
14 all: install lint test

```

Below the code editor, there's a 'Problems' section which is currently empty. At the bottom, the terminal window shows the command `(.gcp-from-zero) noah_gift@cloudshell:~ (cloudai-194723)$`.

Figura 9-5. Editor GCP

Uma conclusão é que o Google Cloud Build é um pouco desajeitado com os testes e o código linting em comparação com o GitHub Actions, mas facilita a implementação de serviços como o Google App Engine.

Agora vamos ver como funciona o GitHub Actions. Podes fazer referência ao [ficheiro de configuração](#) `python-publish.yml`:

```
name: Python application test with GitHub Actions

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Set up Python 3.8
        uses: actions/setup-python@v1
        with:
          python-version: 3.8
      - name: Install dependencies
        run: |
          make install
      - name: Lint with pylint
        run: |
          make lint
      - name: Test with pytest
        run: |
          make test
      - name: Format code
        run: |
          make format
```

Uma diferença crítica entre as duas abordagens é que o GitHub se concentra em fornecer uma experiência incrível ao desenvolvedor, enquanto o GCP se concentra na experiência Cloud. Uma estratégia consiste em utilizar o GitHub Actions para obter feedback do programador, ou seja, para obter linting e testar o código, e utilizar o Google Cloud Build para a implementação.

Com uma ideia dos sistemas CI/CD no GCP, vamos explorar uma das principais tecnologias de computação do Google, o Kubernetes.

Kubernetes Hello World

Uma forma de pensar sobre Kubernetes é como uma "mini-cloud" ou uma "cloud-in-a-box". O Kubernetes permite a criação de aplicações de

complexidade quase infinita. Alguns dos recursos do Kubernetes que o tornam ideal para MLOps incluem:

- Arquitetura de alta disponibilidade
- Escala automática
- Ecossistema rico
- Descoberta de serviços
- Gestão do estado dos contentores
- Gestão de segredos e configuração
- Kubeflow (plataforma ML de ponta a ponta para Kubernetes)

A Figura 9-6 observa que as estruturas de ML do TensorFlow para o scikit-learn se coordenam no topo da arquitetura central do Kubernetes.

Finalmente, o Kubernetes, como discutido anteriormente, pode ser executado em muitas Clouds ou no teu próprio centro de dados.

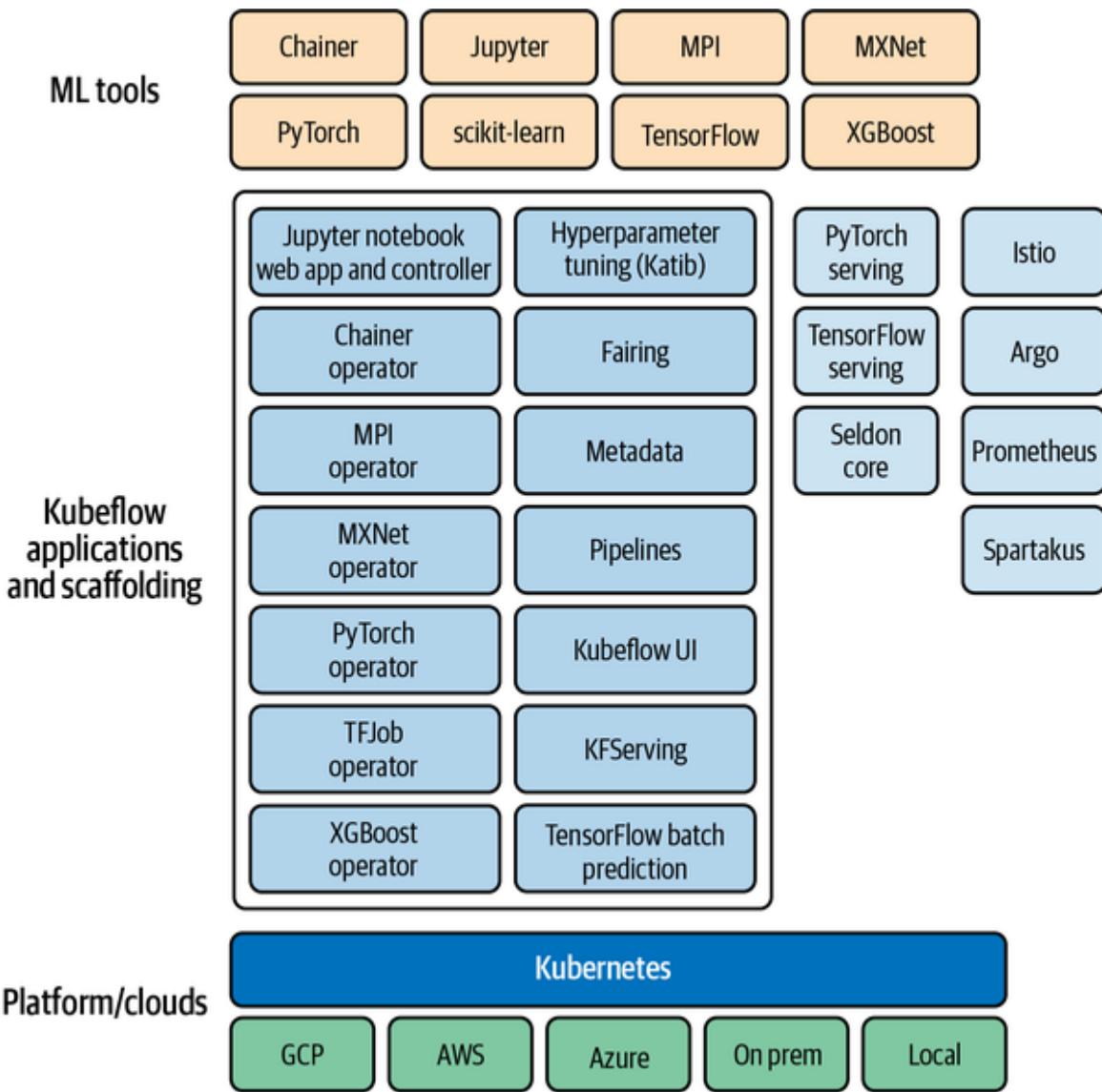


Figura 9-6. Arquitetura do Kubeflow

A base da arquitetura Kubernetes na [Figura 9-7](#) mostra que as principais operações envolvidas no Kubernetes incluem o seguinte:

- Cria um cluster Kubernetes.
- Implantação de uma aplicação no cluster.
- Expondo portas de aplicação.
- Dimensiona uma aplicação.
- Actualiza uma aplicação.

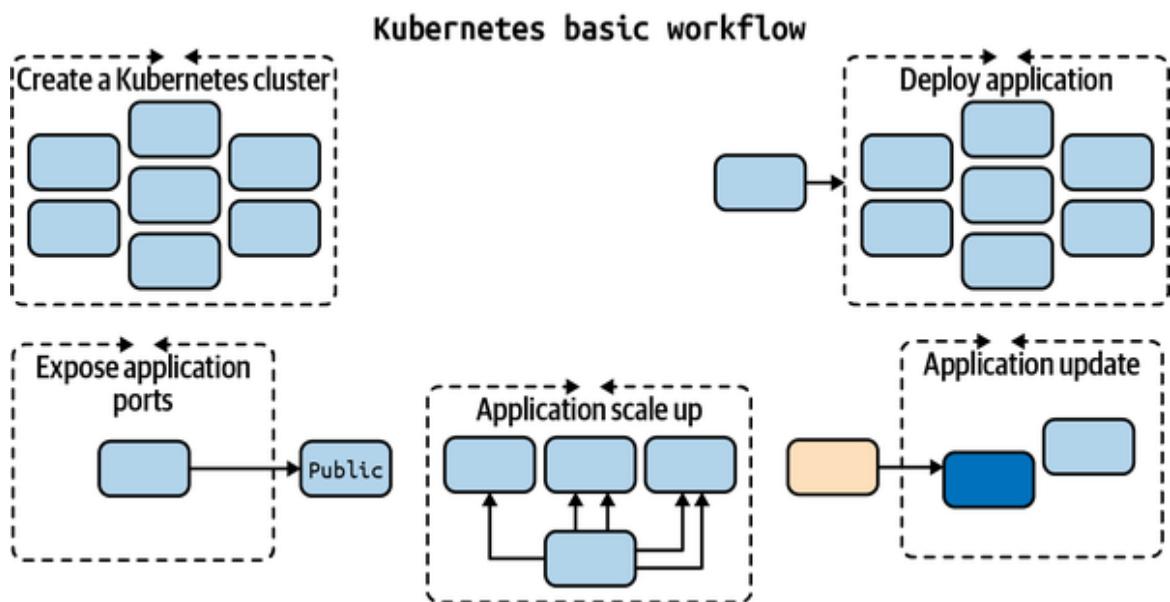


Figura 9-7. Noções básicas do Kubernetes

A hierarquia do Kubernetes na [Figura 9-8](#) mostra um nó de controlo do Kubernetes que gere os outros nós que contêm um ou mais contentores dentro de um pod.

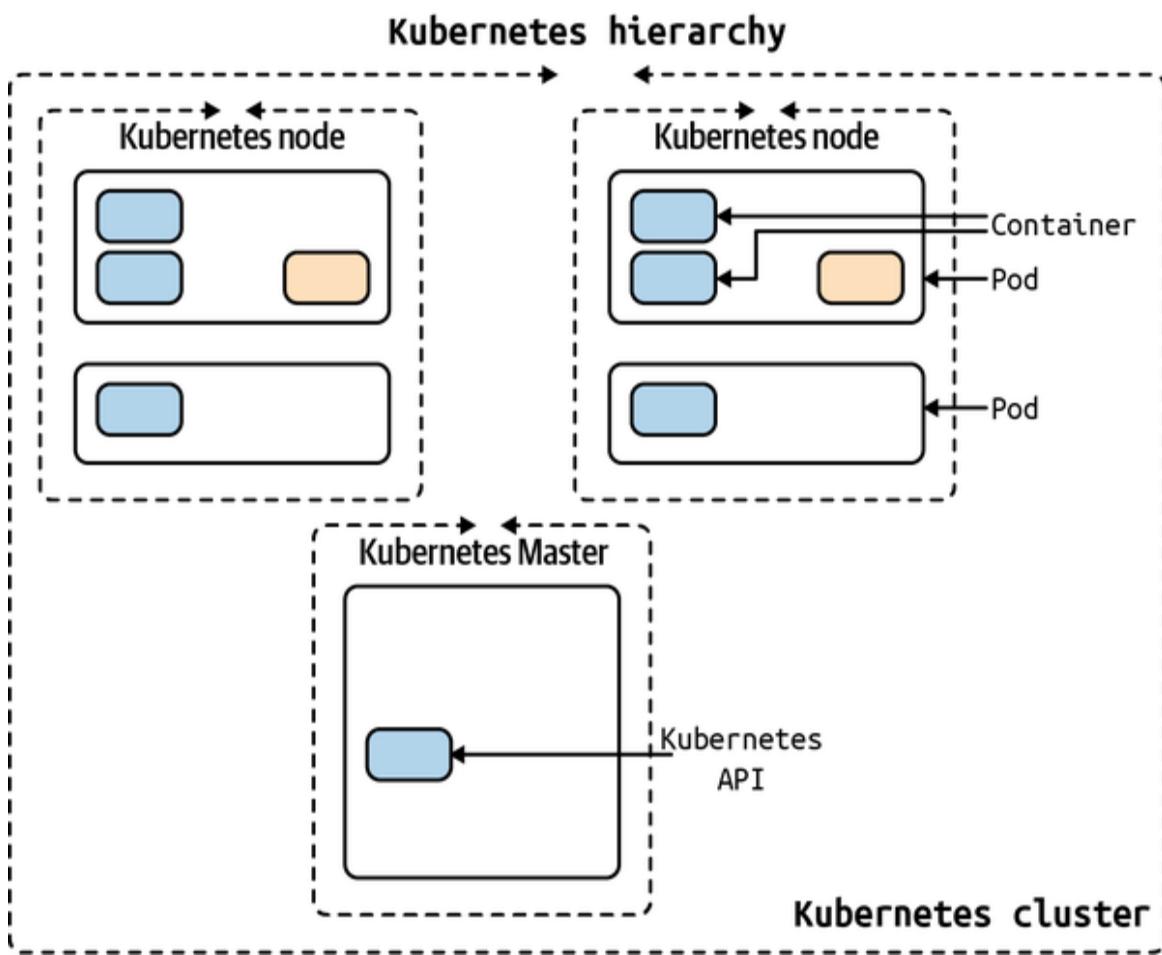


Figura 9-8. Hierarquia do Kubernetes

NOTA

Existem dois métodos principais: configurar um cluster local (de preferência com o Docker Desktop) ou provisionar um cluster Cloud: Amazon através do Amazon EKS, Google através do Google Kubernetes Engine GKE e Microsoft através do Azure Kubernetes Service (AKS).

Um dos recursos "matadores" do Kubernetes é a capacidade de configurar o escalonamento automático por meio do Horizontal Pod Autoscaler (HPA). O HPA do Kubernetes escalará automaticamente o número de pods (lembra-te que podem conter vários contentores) num controlador de replicação, implantação ou conjunto de réplicas. O dimensionamento usa a utilização da CPU, a memória ou métricas personalizadas definidas no Servidor de Métricas do Kubernetes.

Na [Figura 9-9](#), o Kubernetes está usando um loop de controle para monitorar as métricas do cluster e executar ações com base nas métricas recebidas.

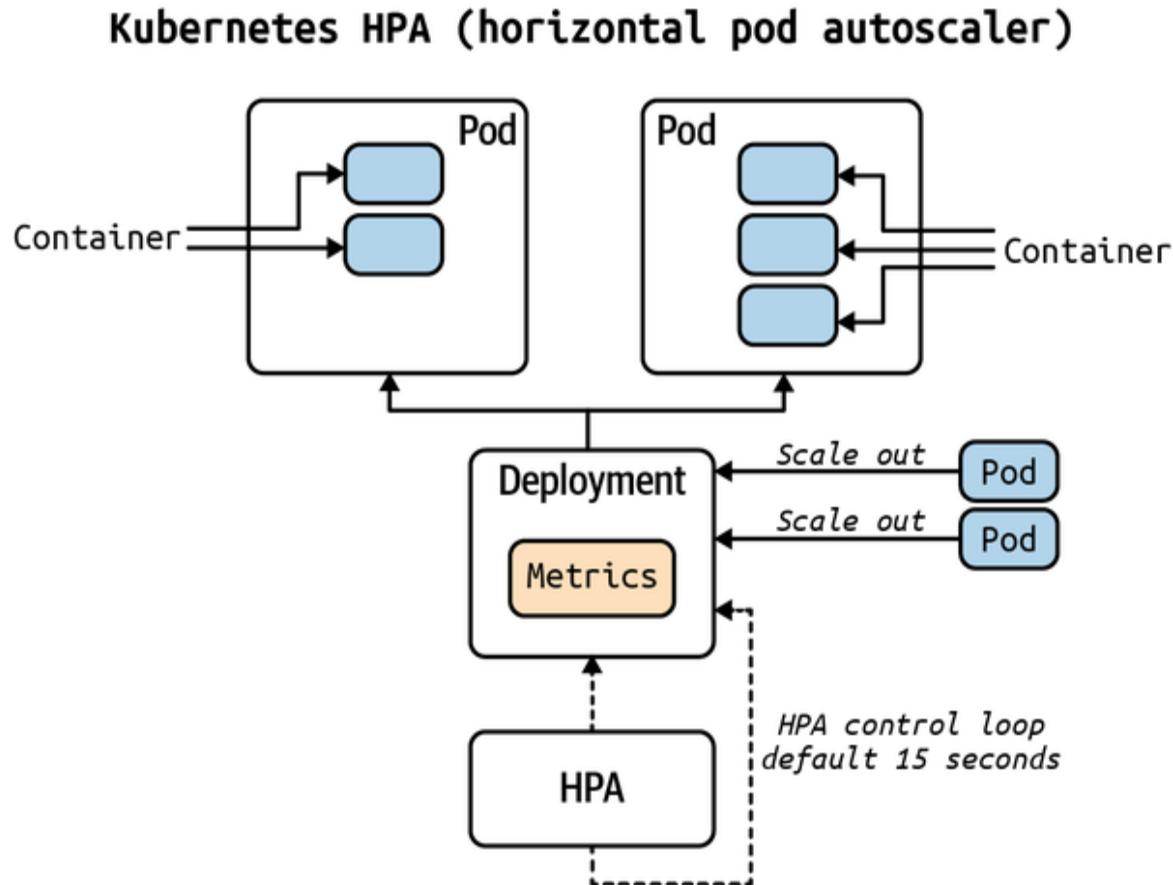


Figura 9-9. Autoscaler de Kubernetes

Como o Kubernetes é um ponto forte da Plataforma do Google e grande parte do MLOps funciona na plataforma, vamos mergulhar em um exemplo de Kubernetes "hello world". Este projeto usa [um aplicativo Flask simples que retorna a alteração correta](#) como o projeto base e o converte para o Kubernetes. Podes encontrar a [fonte completa no repositório no GitHub](#).

Na [Figura 9-10](#), os nós do Kubernetes são anexados ao balanceador de carga.

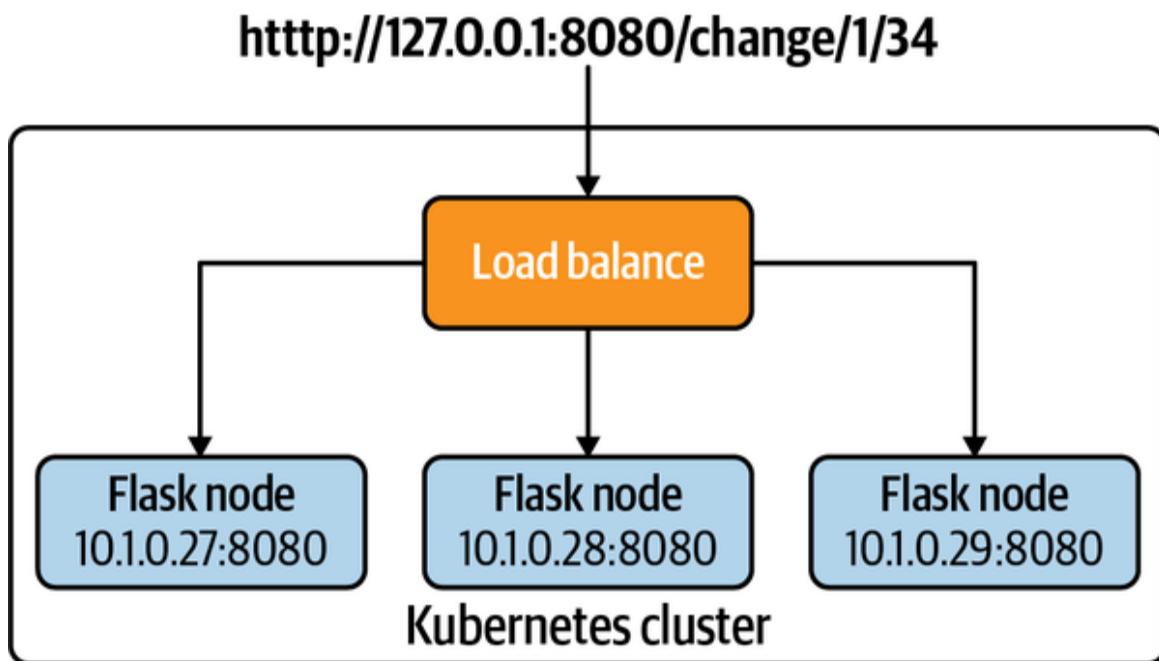


Figura 9-10. Kubernetes olá mundo

Vamos ver os activos no repositório.

- *Faz o Makefile*: Constrói o projeto
- *Dockerfile*: Configuração do contentor
- *app.py*: Aplica o Flask
- *kube-hello-change.yaml*: Configura o YAML do Kubernetes

Para começar, segue os seguintes passos:

1. Cria um ambiente virtual Python:

```
python3 -m venv ~/.kube-hello && source ~/.kube-
hello/bin/activate
```

2. Executa `make all` para executar várias etapas de compilação, incluindo a instalação das bibliotecas, a criação de linting para o projeto e a execução dos testes.

UTILIZAÇÃO DE CONTENTORES CERTIFICADOS

Uma das vantagens do fluxo de trabalho do Docker para os programadores é a utilização de contentores certificados das equipas de desenvolvimento "oficiais". Neste diagrama, um desenvolvedor usa a imagem base oficial do Python desenvolvida pelos principais desenvolvedores do Python. Este passo funciona utilizando a instrução **FROM**, que carrega uma imagem de contentor criada anteriormente.

À medida que o desenvolvedor faz alterações no Dockerfile, ele testa localmente e, em seguida, envia as alterações para um repositório privado do Docker Hub. Por exemplo, na [Figura 9-11](#), as alterações podem ficar disponíveis por um processo de implantação para uma Cloud ou outro desenvolvedor.

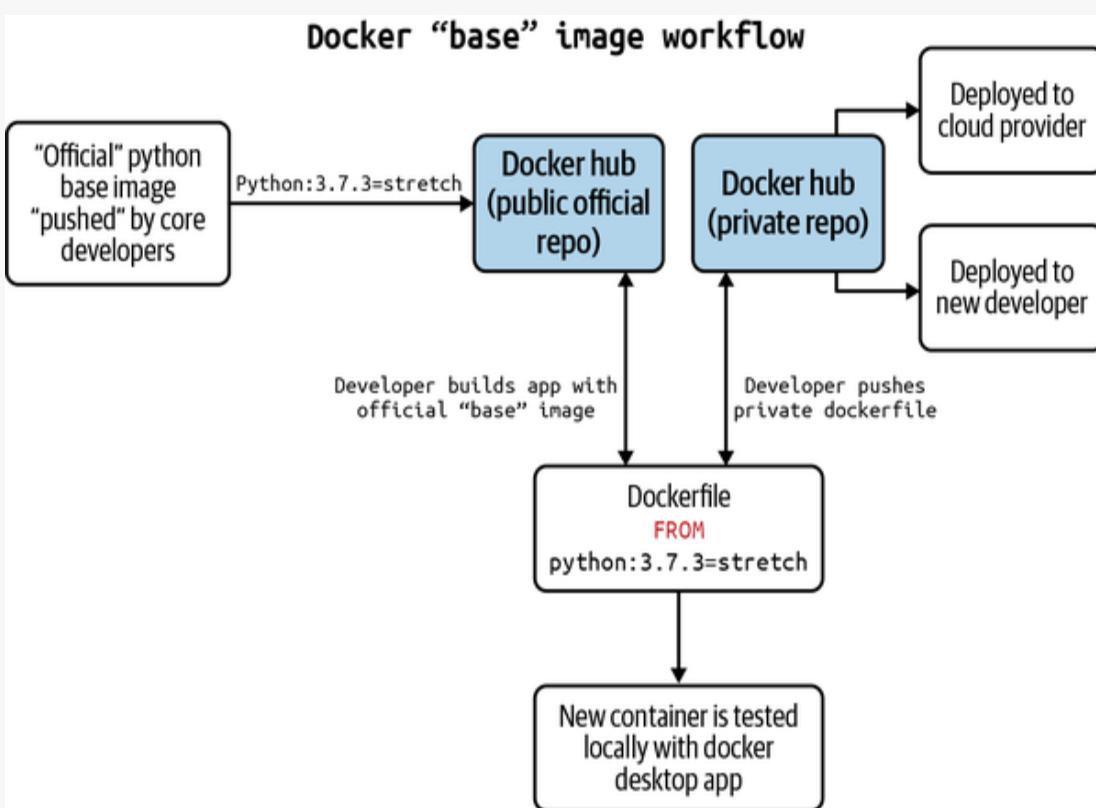


Figura 9-11. Fluxo de trabalho do desenvolvedor do Docker

Em seguida, constrói e executa um Docker Container:

1. Instala o Docker Desktop

2. Para criar a imagem localmente, faz o seguinte:

```
docker build -t flask-change:latest .
```

ou executa `make build`, que tem o mesmo comando.

3. Para verificar a execução do contentor `docker image ls`

4. Para executar o contentor, faz o seguinte:

```
docker run -p 8080:8080 flask-change
```

ou executa `make run`, que tem o mesmo comando.

5. Num terminal separado, invoca o serviço Web através de `curl`, ou executa `make invoke` que tem o mesmo comando:

```
curl http://127.0.0.1:8080/change/1/34
```

Aqui tens o resultado:

```
$ kubectl get nodes
[
  {
    "5": "quarters"
  },
  {
    "1": "nickels"
  },
  {
    "4": "pennies"
  }
]
```

6. Pára o contentor Docker em execução utilizando o comando Ctrl+C

Em seguida, executa o Kubernetes localmente:

1. Verifica se o Kubernetes está a funcionar através do contexto docker-desktop:

```
( .kube-hello) → kubernetes-hello-world-python-flask git:  
(main) kubectl \  
get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
docker-desktop	Ready	master	30d	v1.19.3

2. Executa a aplicação no Kubernetes usando o seguinte comando, que diz ao Kubernetes para configurar o serviço com balanceamento de carga e executá-lo:

```
kubectl apply -f kube-hello-change.yaml
```

ou executa `make run-kube`, que tem o mesmo comando.

Podes ver no ficheiro de configuração que um平衡ador de carga juntamente com três nós são a aplicação configurada:

```
apiVersion: v1  
kind: Service  
metadata:  
  name: hello-flask-change-service  
spec:  
  selector:  
    app: hello-python  
  ports:  
    - protocol: "TCP"
```

```

    port: 8080
    targetPort: 8080
    type: LoadBalancer

    ...
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-python
spec:
  selector:
    matchLabels:
      app: hello-python
  replicas: 3
  template:
    metadata:
      labels:
        app: hello-python
  spec:
    containers:
      - name: flask-change
        image: flask-change:latest
        imagePullPolicy: Never
    ports:
      - containerPort: 8080

```

3. Verifica se o contentor está a funcionar:

```
kubectl get pods
```

Aqui tens o resultado:

NAME	READY	STATUS
RESTARTS	AGE	

flask-change-7b7d7f467b-26htf	1/1	Running	0
8s			
flask-change-7b7d7f467b-fh6df	1/1	Running	0
7s			
flask-change-7b7d7f467b-fpsxr	1/1	Running	0
6s			

4. Descreve o serviço de equilíbrio de carga:

```
kubectl describe services hello-python-service
```

Deves ver um resultado semelhante a este:

```
Name:           hello-python-service
Namespace:      default
Labels:          <none>
Annotations:    <none>
Selector:        app=hello-python
Type:            LoadBalancer
IP Families:   <none>
IP:              10.101.140.123
IPs:             <none>
LoadBalancer Ingress: localhost
Port:            <unset>  8080/TCP
TargetPort:      8080/TCP
NodePort:        <unset>  30301/TCP
Endpoints:      10.1.0.27:8080,10.1.0.28:8080,10.1.0.29:8080
Session Affinity: None
External Traffic Policy: Cluster
Events:          <none>
```

5. Invoca o ponto final para curl:

```
make invoke
```

Na secção seguinte, executa o comando `make invoke` para consultar o microsserviço. A saída dessa ação é mostrada aqui:

```
curl http://127.0.0.1:8080/change/1/34
[
  {
    "5": "quarters"
  },
  {
    "1": "nickels"
  },
  {
    "4": "pennies"
  }
]
```

Para limpar a implantação, executa `kubectl delete deployment hello-python`.

O próximo passo além deste tutorial básico é usar o GKE (Google Kubernetes Engine), o Google Cloud Run (contêiner como serviço) ou o vértice AI para implantar um ponto de extremidade de aprendizado de máquina. Podes usar o [repositório Python MLOps Cookbook](#) como base para o fazer. A tecnologia Kubernetes é uma excelente base para a criação de APIs alimentadas por ML e, com o GCP, existem muitas opções disponíveis se utilizares contentores no formato Docker desde o início.

Com os conceitos básicos de computação no GCP fora do caminho, vamos discutir como as bases de dados nativas da Cloud, como o Google BigQuery, contribuem muito para a adoção de MLOps.

Design e escolha da base de dados nativa Cloud

Uma das jóias da coroa da Google Cloud Platform é o Google BigQuery, por algumas razões. Uma das razões é a facilidade de começar e outra razão são as bases de dados publicamente disponíveis. Uma boa lista de conjuntos de dados abertos do Google BigQuery está disponível [nesta página do Reddit](#). Por fim, do ponto de vista do MLOps, um dos principais recursos do Google BigQuery é a capacidade de treinar e hospedar modelos de ML dentro da plataforma do Google BigQuery.

Ao olhar para a [Figura 9-12](#), repara que o Google BigQuery é o centro de um pipeline MLOps que pode exportar produtos para a inteligência empresarial e para a engenharia de aprendizagem automática, incluindo a IA vértice. Esse fluxo de trabalho MLOps é possível graças às entradas DataOps (Operacionalização de dados), como conjuntos de dados públicos, API de streaming e o produto Google Dataflow. O facto de o Google BigQuery executar a aprendizagem automática em linha simplifica o processamento de grandes conjuntos de dados.

MLOps with Google BigQuery

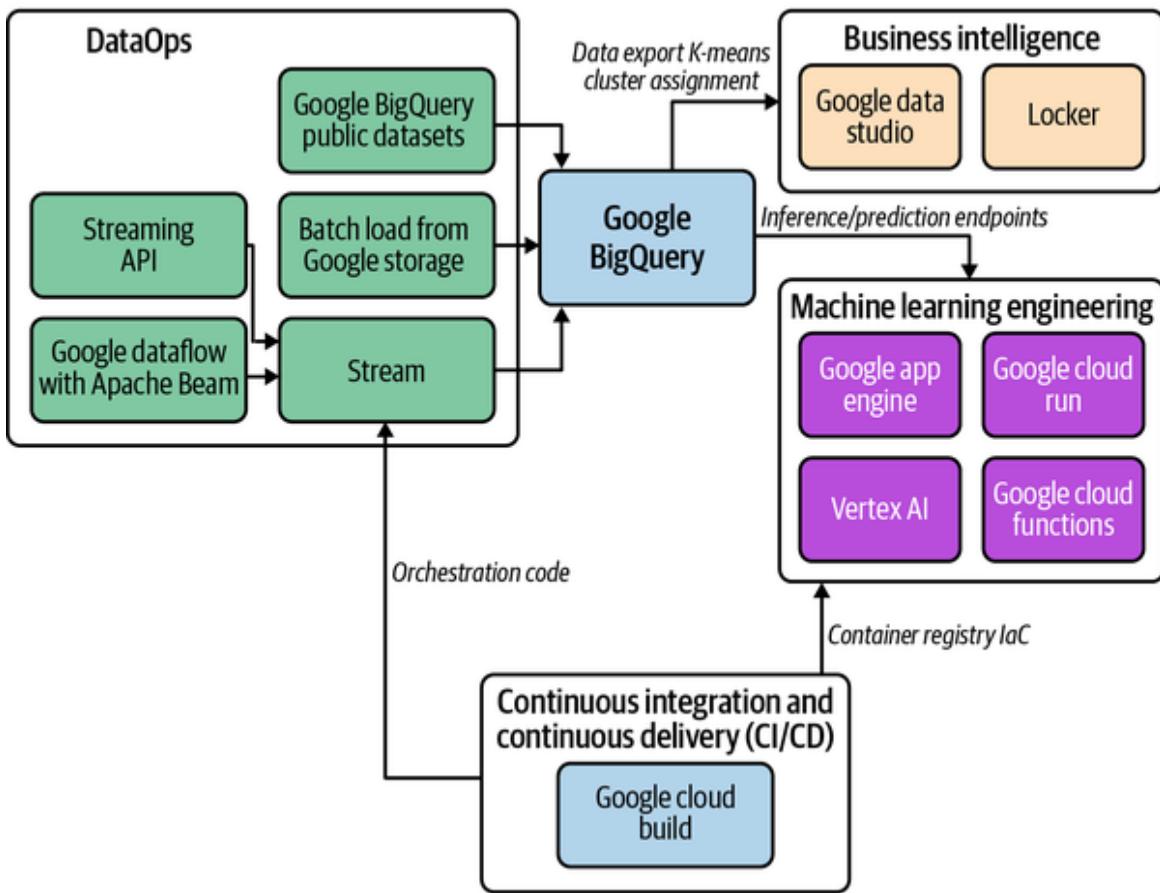


Figura 9-12. Fluxo de trabalho do Google BigQuery MLOps

Um exemplo deste fluxo de trabalho é apresentado na [Figura 9-13](#), onde, após a modelação ML ter ocorrido no Google BigQuery, os resultados são exportados para o Google Data Studio. O artefacto criado a partir do BigQuery é a análise de agrupamento K-means [apresentada neste relatório partilhável](#).



Figura 9-13. Agrupamento K-means do Google Data Studio

Como ponto de partida para a realização de MLOps na plataforma GCP, o BigQuery é uma escolha óptima devido à flexibilidade da plataforma. Em

seguida, vamos falar sobre DataOps e Engenharia de dados aplicada na plataforma GCP.

DataOps no GCP: Engenharia de dados aplicada

Os dados são necessários para a construção da aprendizagem automática em escala e, como tal, são um aspeto crítico dos MLOps. De certa forma, o GCP tem uma quantidade quase ilimitada de formas de automatizar o fluxo de dados. Este facto deve-se à variedade de opções de computação e armazenamento disponíveis, incluindo ferramentas de alto nível como o Dataflow.

Para simplificar, vamos usar uma abordagem sem servidor para a engenharia de dados usando o Cloud Functions. Para fazer isso, vamos ver como o Google Cloud Functions funciona e como ele pode servir como solução de ML por meio de uma chamada de API de IA ou como um pipeline de MLOps usando o Cloud Functions para falar com o [Google Pub/Sub](#).

Vamos começar com uma Função do Google Cloud intencionalmente simples que retorna a alteração correta. Podes encontrar [o exemplo completo aqui](#).

Para começar, abre a Consola do Google Cloud, cria uma nova Função Cloud e cola o seguinte código no interior, conforme apresentado na [Figura 9-14](#). Também podes "desalternar" a opção "Exigir autenticação" para "Permitir invocações não autenticadas".

```
import json

def hello_world(request):

    request_json = request.get_json()
    print(f"This is my payload: {request_json}")
    if request_json and "amount" in request_json:
        raw_amount = request_json["amount"]
        print(f"This is my amount: {raw_amount}")
```

```
    amount = float(raw_amount)
    print(f"This is my float amount: {amount}")
res = []
coins = [1, 5, 10, 25]
coin_lookup = {25: "quarters", 10: "dimes", 5: "nickels", 1:
"pennies"}
coin = coins.pop()
num, rem = divmod(int(amount * 100), coin)
res.append({num: coin_lookup[coin]})
while rem > 0:
    coin = coins.pop()
    num, rem = divmod(rem, coin)
    if num:
        if coin in coin_lookup:
            res.append({num: coin_lookup[coin]})

result = f"This is the res: {res}"
return result
```

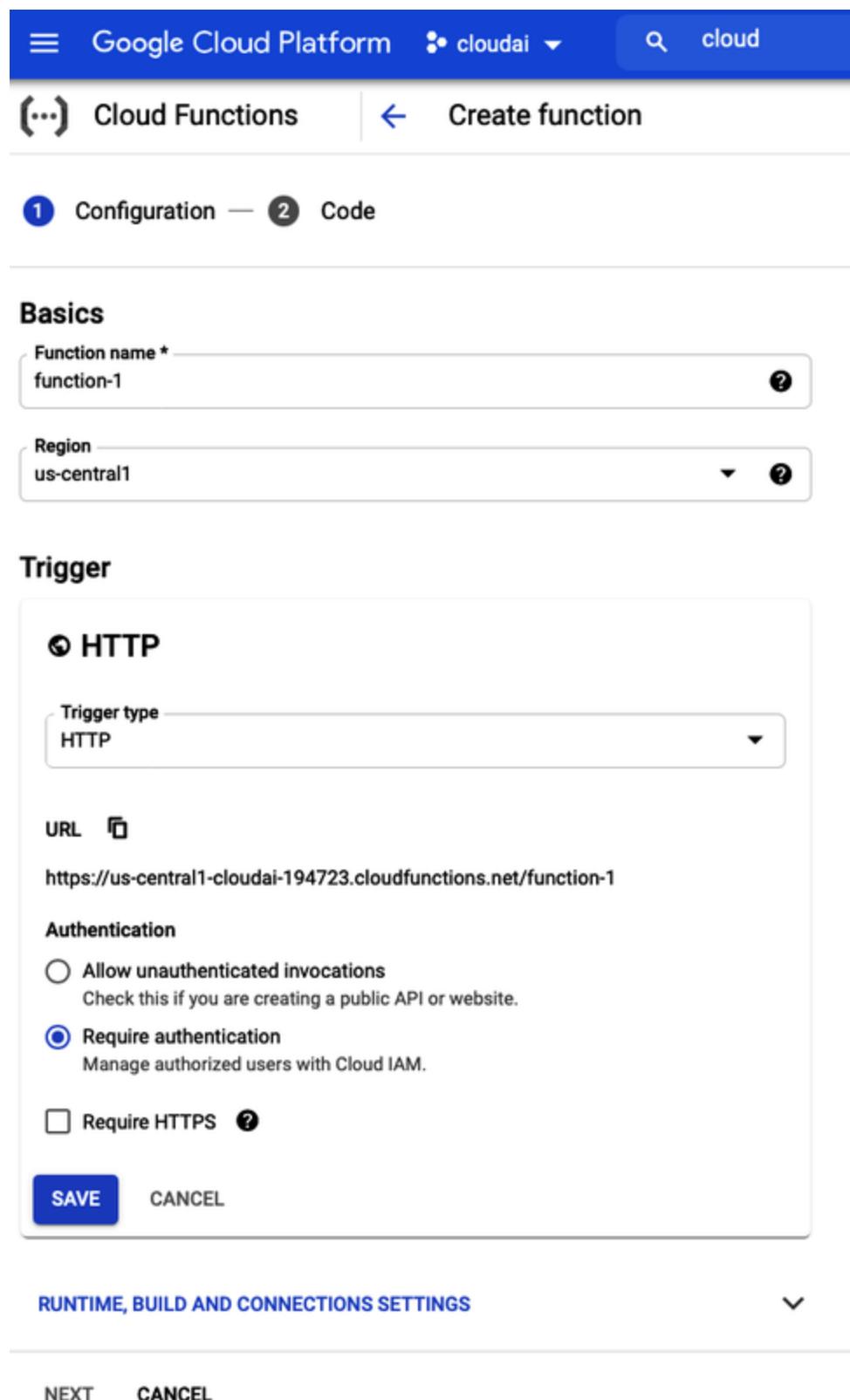


Figura 9-14. Função Google Cloud

Para invocar através da linha de comando gcloud, faz o seguinte:

```
gcloud functions call changemachine --data '{"amount":"1.34"}'
```

Para invocar através do comando `curl`, podes utilizar o seguinte.

```
curl -d '{
  "amount": "1.34"
}'      -H "Content-Type: application/json" -X POST
<trigger>/function-3
```

Outra abordagem é criar uma ferramenta de linha de comando para invocar o teu ponto final:

```
#!/usr/bin/env python
import click
import requests

@click.group()
@click.version_option("1.0")
def cli():
    """Invoker"""

    @cli.command("http")
    @click.option("--amount", default=1.34, help="Change to Make")
    @click.option(
        "--host",
        default="https://us-central1-cloudai-
194723.cloudfunctions.net/change722",
        help="Host to invoke",
    )
    def mkrequest(amount, host):
        """Asks a web service to make change"""

        click.echo(
            click.style(
                f"Querying host {host} with amount: {amount}",
                bg="green", fg="white"
            )
        )
        payload = {"amount": amount}
        result = requests.post(url=host, json=payload)
        click.echo(click.style(f"result: {result.text}", bg="red",
fg="white"))
```

```
if __name__ == "__main__":
    cli()
```

Finalmente, uma outra abordagem é carregar o teu modelo de ML para a IA v rtice ou chamar um ponto de extremidade da API existente que executa vis o computacional, PNL ou outra tarefa relacionada com ML. Podes encontrar o exemplo completo no GitHub. No exemplo seguinte, vamos utilizar uma API de PNL pr -existente. Tamb m ser  necess rio adicionar duas bibliotecas de terceiros editando o arquivo *requirements.txt* inclu do no scaffolding do Google Cloud (consulte a Figura 9-15).

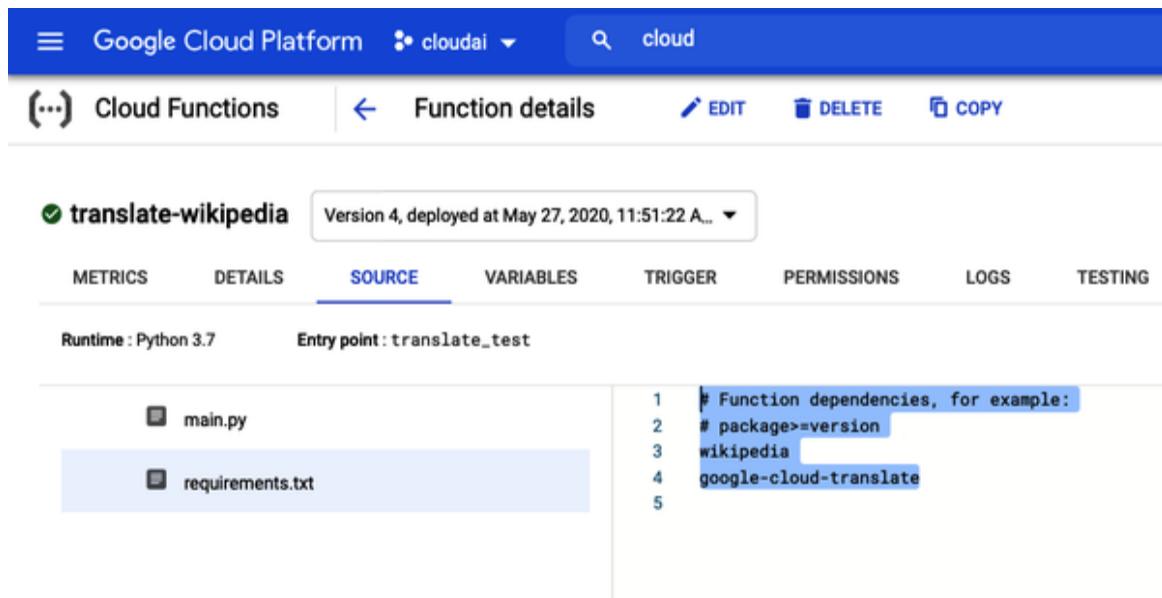


Figura 9-15. Adiciona requisitos

Cola este c digo na fun o *main.py* na Consola do Google Cloud Shell:

```
import wikipedia

from google.cloud import translate


def sample_translate_text(
    text="YOUR_TEXT_TO_TRANSLATE", project_id="YOUR_PROJECT_ID",
    language="fr"
):
    """Translating Text."""
    pass
```

```

client = translate.TranslationServiceClient()

parent = client.location_path(project_id, "global")

# Detail on supported types can be found here:
# https://cloud.google.com/translate/docs/supported-formats
response = client.translate_text(
    parent=parent,
    contents=[text],
    mime_type="text/plain", # mime types: text/plain,
text/html
    source_language_code="en-US",
    target_language_code=language,
)
print(f"You passed in this language {language}")
# Display the translation for each input text provided
for translation in response.translations:
    print("Translated text:
{}".format(translation.translated_text))
return "Translated text:
{}".format(translation.translated_text)

def translate_test(request):
    """Takes JSON Payload {"entity": "google"}"""
    request_json = request.get_json()
    print(f"This is my payload: {request_json}")
    if request_json and "entity" in request_json:
        entity = request_json["entity"]
        language = request_json["language"]
        sentences = request_json["sentences"]
        print(entity)
        res = wikipedia.summary(entity, sentences=sentences)
        trans = sample_translate_text(
            text=res, project_id="cloudai-194723",
            language=language
        )
        return trans
    else:
        return f"No Payload"

```

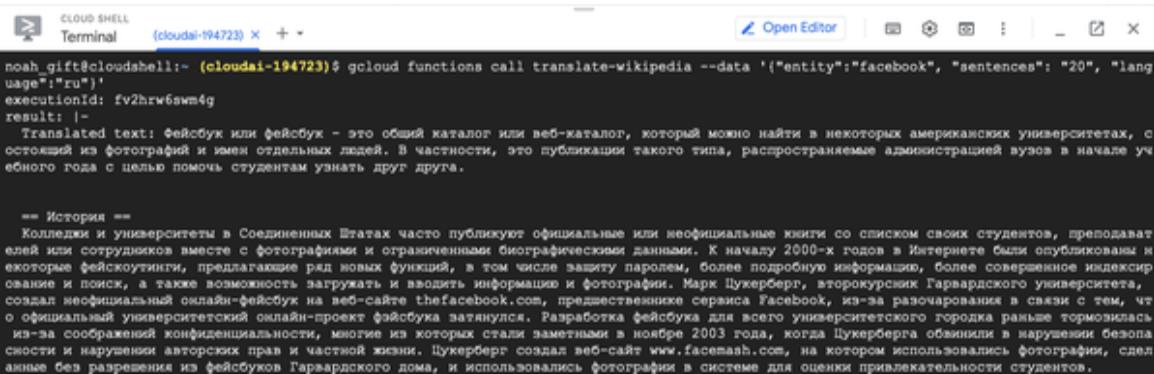
Para invocar a função, podes chamá-la a partir do Google Cloud Shell:

```

gcloud functions call translate-wikipedia --data\
'{"entity":"facebook", "sentences": "20", "language":"ru"}'

```

Podes ver o resultado da tradução para russo no terminal do Google Cloud Shell na Figura 9-16.



```
noah gift@cloudshell:~ (cloudai-194723)$ gcloud functions call translate-wikipedia --data '{"entity": "facebook", "sentences": "20", "language": "ru"}'
executionId: fv2hrw6swm4g
result: {
  Translated text: Фейсбук или фейсбук - это общий каталог или веб-каталог, который можно найти в некоторых американских университетах, состоящий из фотографий и имен отдельных людей. В частности, это публикации такого типа, распространяемые администрацией вузов в начале учебного года с целью помочь студентам узнать друг друга.

  == История ==
  Колледжи и университеты в Соединенных Штатах часто публикуют официальные или неофициальные книги со списком своих студентов, преподавателей или сотрудников вместе с фотографиями и ограниченными биографическими данными. К началу 2000-х годов в Интернете были опубликованы и некоторые фейсбукинги, предлагавшие ряд новых функций, в том числе защиту паролем, более подробную информацию, более совершенный индексир ование и поиск, а также возможность загружать и вводить информацию и фотографии. Марк Цукерберг, второкурсник Гарвардского университета, создал неофициальный онлайн-фейсбук на веб-сайте thefacebook.com, предшественнике сервиса Facebook, из-за разочарования в связи с тем, что официальный университетский онлайн-проект фейсбука затянулся. Разработка фейсбука для всего университетского города раннее тормозилась из-за сообщений конфиденциальности, многие из которых стали заметными в ноябре 2003 года, когда Цукерберга обвинили в нарушении авторских прав и частной жизни. Цукерберг создал веб-сайт www.facemash.com, на котором использовались фотографии, сделанные без разрешения из фейсбуков Гарвардского дома, и использовались фотографии в системе для оценки привлекательности студентов.
```

Figura 9-16. Traduzir

Para criar protótipos de fluxos de trabalho de engenharia de dados, não existe um método mais rápido do que a tecnologia sem servidor, como o Google Cloud Functions. A minha recomendação é resolver um fluxo de trabalho inicial de engenharia de dados utilizando a tecnologia sem servidor e passar para ferramentas mais complexas, se necessário.

Uma nota importante é que a plataforma de IA da Vértice acrescenta muitos componentes adicionais de engenharia de dados e de engenharia de ML que melhoram projectos maiores. Em particular, a capacidade de utilizar a IA explicável, controlar a qualidade de um modelo e utilizar um armazenamento de características são componentes valiosos de uma solução MLOps abrangente. Vamos analisar estas opções a seguir.

Operacionalização de modelos de ML

Atualmente, todas as grandes plataformas Cloud têm uma plataforma MLOps. No GCP, a plataforma é a IA vértice e integra muitos serviços individuais que desenvolveu ao longo dos anos, incluindo a tecnologia AutoML. Em particular, alguns dos componentes essenciais das plataformas MLOps incluem Armazenamento de Características, IA Explicável e controlo da qualidade do modelo. Se iniciares um projeto MLOps numa empresa de maiores dimensões, o primeiro ponto de partida no GCP será a

sua plataforma Vertex AI, tal como o SageMaker no AWS ou o Azure ML Studio no Azure.

Outra opção é utilizar componentes como soluções autónomas para operacionalizar modelos de ML na plataforma GCP. Um serviço a utilizar é o **serviço de previsão** para implementar modelos e depois aceitar pedidos.

Por exemplo, podes testar um modelo local do sklearn utilizando um comando semelhante ao seguinte:

```
gcloud ai-platform local predict --model-dir\  
    LOCAL_OR_CLOUD_STORAGE_PATH_TO_MODEL_DIRECTORY/ \  
--json-instances LOCAL_PATH_TO_PREDICTION_INPUT.JSON \  
--framework NAME_OF_FRAMEWORK
```

Mais tarde, poderás criar um ponto de extremidade e chamá-lo a partir de um exemplo mostrado anteriormente no capítulo, por exemplo, Google Cloud Functions, Google Cloud Run ou Google App Engine.

Vamos analisar um exemplo de como um projeto do Google App Engine fica na Cloud do GCP usando **este repositório como ponto de partida**. Para começar, observa a arquitetura principal da entrega contínua no GCP. Para começar, cria um novo projeto do Google App Engine, conforme mostrado no fluxo de trabalho "leve" do MLOps na **Figura 9-17**.

NOTA

Repara que este fluxo de trabalho leve permite uma forma transparente e direta de implementar um modelo de ML, mas um fluxo "pesado" pode acrescentar um enorme valor se as funcionalidades apresentadas, como a IA explicável, forem necessárias para um projeto.

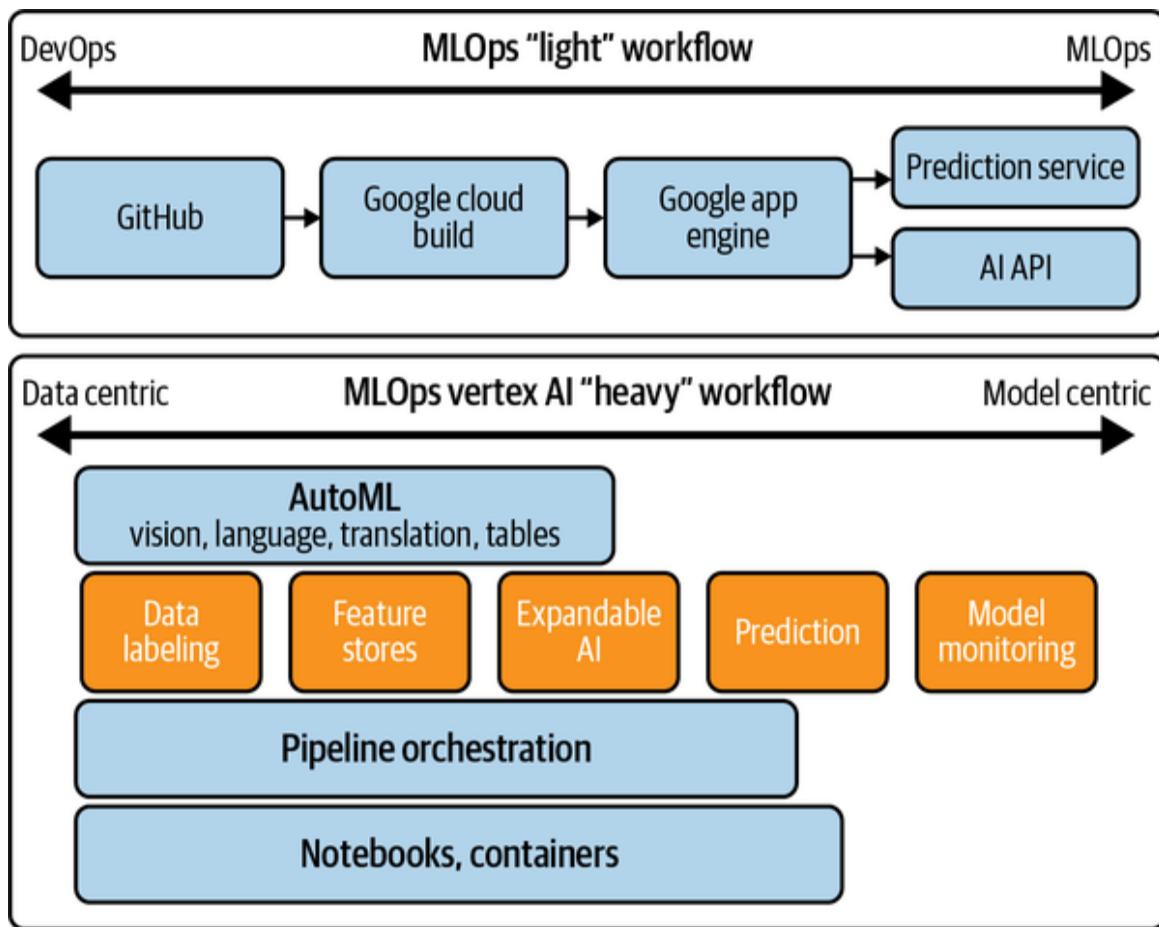


Figura 9-17. Fluxos de trabalho leves e pesados do MLOps

Em seguida, ativa a API Cloud Build, conforme mostrado na Figura 9-18.

The screenshot shows the Google Cloud Platform interface for Cloud Build settings:

- Left sidebar:** Shows Cloud Build navigation with options: Dashboard, History, Triggers, and Settings (which is selected).
- Right pane:**
 - Settings:** Subtitle under Cloud Build.
 - Service Account:** Displays the service account email: 318125260996@cloudbuild.gserviceaccount.com.
 - DATA SHARING:** A table showing roles assigned to various GCP services:

GCP Service	Role	Status
Cloud Functions	Cloud Functions Developer	ENABLED
Cloud Run	Cloud Run Admin	DISABLED
App Engine	App Engine Admin	ENABLED
Kubernetes Engine	Kubernetes Engine Developer	DISABLED
Compute Engine	Compute Instance Admin (v1)	DISABLED
Firebase	Firebase Admin	DISABLED
Cloud KMS	Cloud KMS CryptoKey Decrypter	DISABLED
Secret Manager	Secret Manager Secret Accessor	DISABLED
Service Accounts	Service Account User	ENABLED
 - Bottom note:** Roles not listed here can be managed in the [IAM section](#).

Figura 9-18. Constrói a Cloud

O arquivo *cloudbuild.yml* precisa apenas de um comando de implantação:

```
steps:  
- name: "gcr.io/cloud-builders/gcloud"  
  args: ["app", "deploy"]  
timeout: "1600s"
```

Os únicos outros requisitos são *app.yaml*, *requirements.txt* e *main.py*, todos encontrados neste [repositório de exemplo](#). Um passo final para fazer com que esta aplicação faça qualquer forma de aprendizagem automática é chamar uma API ML/AI ou utilizar o alojamento do ponto final da Plataforma AI.

A vantagem da abordagem simples é que é fácil configurar todo um pipeline MLOps em uma ou duas horas, no máximo. Também podes escolher entre APIs de IA, serviços de previsão e um ponto de extremidade AutoML.

Existem abordagens "leves" e "pesadas" para realizar MLOps no GCP. Este exemplo explorou a abordagem "leve", mas há mérito em utilizar a tecnologia da plataforma vértice AI, uma vez que inclui funcionalidades avançadas que muitas empresas desejam.

Vamos terminar o capítulo e discutir os próximos passos para utilizar o GCP para MLOps.

Conclusão

Uma conclusão final sobre as tecnologias MLOps, como a IA vértice, é que elas resolvem um problema complicado melhor do que a maioria das organizações pode fazer por conta própria. Lembro-me de falar com alguém num laboratório de pesquisa, e eles gabaram-se de como a Cloud era sobrevalorizada, uma vez que tinham uma tonelada de GPUs. Uma tonelada de GPUs não te dá serviços de plataforma como estas plataformas. Esta afirmação é um mal-entendido fundamental sobre como o software empresarial e as startups funcionam. A vantagem comparativa é crítica

tanto para startups em fase inicial como para empresas da Fortune 500. Não construas algo pior do que aquilo que podes comprar por um custo trivial.

Recomendo que pegues em tudo o que foi dito neste capítulo e o apliques num projeto final de aprendizagem automática que consiste em criar uma aplicação de ML nativa da Cloud no GCP. Este projeto deve dar-te a capacidade de criar soluções realistas e funcionais concebidas com técnicas modernas.

Antes de começares, certifica-te de que lês o artigo de Sculley et al. (2015) para considerar **a dívida técnica em sistemas de aprendizagem automática (ML)**. O teu projeto pode beneficiar da utilização de dados públicos dos conjuntos de dados do Google BigQuery. Como alternativa, se estiveres a usar o AutoML, os dados podem ser dados de tutoriais ou dados personalizados.

A ideia principal é criar um projeto de portfólio que demonstre a tua capacidade de fazer engenharia de ML no Google Cloud. Segue-se uma sugestão de requisitos para o projeto:

- Código fonte armazenado no GitHub
- Implementação contínua a partir do CircleCI
- Dados armazenados no GCP (BigQuery, Google Cloud Storage, etc.)
- Previsões de ML criadas e fornecidas (AutoML, BigQuery, Plataforma de IA, etc.)
- Monitorização nativa da Cloud
- O Google App Engine envia pedidos HTTP através da API REST com um payload JSON
- Implementado no ambiente GCP utilizando o Google Cloud Build

Eis alguns itens a acrescentar a uma lista de verificação dos requisitos do projeto final:

- A aplicação faz inferência de ML?
- Existem ambientes separados?
- Existe uma monitorização e alertas abrangentes?
- É utilizado o datastore correto?
- Aplica o princípio da menor segurança?
- Os dados são encriptados em trânsito?

Podes ver alguns projectos recentes de estudantes e de topo na área da ciência de dados no [repositório de código oficial](#). Estes projectos fornecem um quadro de referência para o que podes construir, e podes submeter um pull request no futuro para que o teu projeto seja adicionado.

- Jason Adams: Análise de sentimento FastAPI com Kubernetes
- James Salafatinos: Classificação de imagens em tempo real com TensorFlow.js
- Nikhil Bhargava: Previsão do preço dos ténis
- Predictor de Covid
- Absentismo no trabalho

O próximo capítulo aborda a interoperabilidade da aprendizagem automática e a forma como resolve de forma única os problemas de MLOps que surgem de diferentes plataformas, tecnologias e formatos de modelos.

Exercícios

- Usando o Google Cloud Shell Editor, cria um novo repositório GitHub com o scaffolding Python necessário usando um Makefile, linting e testes. Adiciona etapas como a formatação de código no teu Makefile.

- Cria um pipeline "hello world" para o Google Cloud que chama um projeto do Google App Engine (GAE) baseado em Python e retorna "hello world" como uma resposta JavaScript Object Notation (JSON).
- Cria um pipeline de ingestão para ETL utilizando ficheiros CSV e o Google BigQuery. Agende um cron job recorrente para atualizar os dados em lote.
- Treina um modelo de classificação multiclasse na visão Google AutoML e implementa-o num dispositivo de ponta.
- Cria um ambiente de produção e de desenvolvimento e implementa um projeto em ambos os ambientes utilizando o Google Cloud Build.

Questões para discussão sobre pensamento crítico

- Que problemas é que um sistema de IC resolve e porque é que um sistema de IC é uma parte essencial do software SaaS?
- Porque é que as plataformas Cloud são o alvo ideal para aplicações analíticas e como é que o Deep Learning beneficia da Cloud?
- Quais são as vantagens dos serviços geridos como o Google BigQuery e em que é que o Google BigQuery difere de um SQL tradicional?
- Como é que a previsão de ML diretamente do BigQuery acrescenta valor à plataforma Google e que vantagens poderá ter para a engenharia de aplicações analíticas?
- Como é que o AutoML tem um custo total de propriedade (TCO) mais baixo e como é que poderia ter um TCO mais elevado?

Capítulo 10. Interoperabilidade da aprendizagem automática

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

Os cérebros dos mamíferos têm um poder considerável para a computação generalizada, mas funções especiais (por exemplo, a subjetividade) requerem normalmente estruturas especializadas. Uma tal estrutura hipotética foi chamada, de forma jocosa, de "bomba da subjetividade" por Marcel Kinsbourne. Bem, é exatamente isso que alguns de nós procuram. E o mecanismo para a subjetividade é duplo, como mostra a dualidade da anatomia, o sucesso da hemisferectomy e os resultados da divisão do cérebro (em gatos e macacos, bem como em humanos).

—Dr. Joseph Bogen

O Peru tem vários *milhares* de variedades de batatas. Como alguém que cresceu no Peru, acho isto surpreendente. É fácil pensar que a maioria das batatas tem um sabor semelhante, mas não é bem assim. Pratos diferentes requerem variedades diferentes de batatas. Não vais querer discutir com um cozinheiro peruano se a receita pedir batatas *Huayro* e tu quiseres usar a batata comum *para assar* encontrada na maior parte dos EUA. Se alguma vez tiveres a oportunidade de estar na América do Sul (e certamente no Peru), experimenta a experiência de passear por um mercado de rua. A quantidade de legumes frescos, incluindo várias dezenas de variedades de batatas, pode deixar-te tonto.

Já não vivo no Peru e sinto falta dessa variedade de batatas. Não consigo cozinar alguns pratos e dar-lhes o mesmo sabor com uma batata comum

comprada no supermercado local. Não é a mesma coisa. Considero que a variedade e os diferentes sabores proporcionados por um vegetal tão humilde são fundamentais para a identidade da cozinha peruana. Podes continuar a achar que não há problema em cozinhar com a batata comum, mas, no fim de contas, trata-se de escolhas e selecções.

A variedade e a capacidade de escolher o que melhor se adapta às tuas necessidades são uma vantagem. Esta capacidade também se aplica à aprendizagem automática quando se trata do produto final: o modelo treinado.

Os modelos de aprendizagem automática treinados também têm restrições distintas e a maioria deles não vai funcionar num ambiente que não esteja especificamente adaptado para o suportar. O principal conceito de interoperabilidade de modelos é poder *transformar* um modelo de uma plataforma para outra, o que cria escolhas. Este capítulo defende que, embora existam argumentos sólidos para usar modelos prontos para uso de um provedor de Cloud e não se preocupar muito com a dependência de fornecedores, é essencial entender como exportar modelos para outros formatos que podem funcionar em outras plataformas com restrições diferentes. Assim como os contêineres podem funcionar perfeitamente em quase todos os sistemas, desde que haja um tempo de execução de contêiner subjacente (como explico em "[Contêineres](#)"), a interoperabilidade de modelos ou ter um modelo exportado para diferentes formatos é fundamental para a flexibilidade e a capacitação. [A Figura 10-1](#) demonstra uma visão geral aproximada do que esta interoperabilidade significa ao treinar em qualquer estrutura de ML, transformando o modelo resultante *uma vez*, para o implementar em praticamente qualquer lugar: desde dispositivos de ponta a telemóveis e outros sistemas operativos.

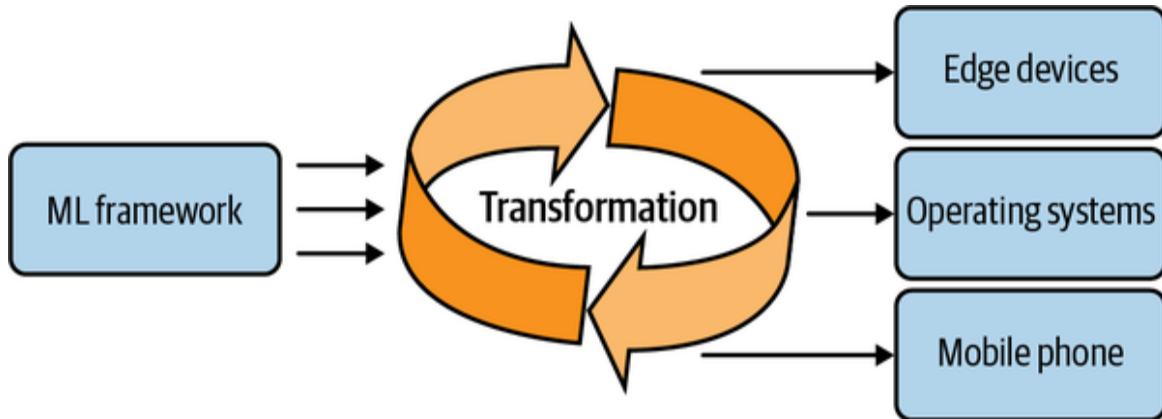


Figura 10-1. Visão geral da interoperabilidade

Esta situação é como produzir parafusos que podem funcionar com qualquer chave de fendas, em vez de produzir parafusos que só podem funcionar com chaves de fendas de uma única loja de artigos para a casa. Em ["Dispositivos Edge"](#), já tivemos problemas quando um modelo não funcionava com o Edge TPU, e a conversão acabou por ser necessária. Atualmente, existem algumas opções interessantes, e estou particularmente surpreendido com o ONNX, um projeto orientado para a comunidade com normas abertas que pretende facilitar a interação com modelos, reduzindo a complexidade das cadeias de ferramentas. Este capítulo abordará alguns dos detalhes que tornam o ONNX uma opção atraente de aprendizado de máquina e como a maioria das Clouds já está suportando esse formato.

Porque é que a interoperabilidade é fundamental

A abstração de processos e interações complexas é um padrão típico da engenharia de software. Por vezes, as abstrações podem ficar completamente fora de controlo, fazendo com que a abstração seja tão complexa como o software subjacente que estava a tentar abstrair. Um excelente exemplo disso é o Openstack (uma plataforma de infraestrutura como serviço de código aberto) e seu instalador. Instalar e configurar uma plataforma de infraestrutura pode ser muito complicado. Diferentes tipos de máquinas e topologias de rede criam uma combinação difícil de resolver com um instalador de tamanho único.

Um novo instalador foi criado para facilitar a instalação do TripleO (Openstack On Openstack). O TripleO produzia uma instância temporária que, por sua vez, instalava o Openstack. O projeto resolveu muitos problemas associados à instalação e configuração, mas alguns pensaram que ainda era complicado e que era necessário abstrair mais. Foi assim que surgiu o QuintupleO (Openstack On Openstack On Openstack). Sem estar ativamente envolvido no Openstack, posso dizer-te que é difícil de implementar e que as equipas de engenharia estão a tentar resolver esses problemas de forma genérica. Mas tenho dúvidas de que adicionar outra camada seja a solução.

É fácil convences-te de que mais uma camada tornará as coisas mais fáceis, mas, na realidade, é muito difícil fazer isto bem e agradar a todos. Tenho uma pergunta aberta que utilizo frequentemente para conceber sistemas: *o sistema pode ser muito simples e opinativo, ou flexível e complexo. Qual é que escolhes?* Ninguém gosta destas opções, e toda a gente prefere o *simples e flexível*. É possível criar um sistema assim, mas é difícil de o conseguir.

Na aprendizagem automática, várias plataformas e fornecedores de Cloud treinam modelos de formas diferentes e particulares. Isto não tem grande importância se permanecer na plataforma e na forma como esta interage com o modelo, mas é uma receita para a frustração se alguma vez precisares de pôr esse modelo treinado a funcionar noutro local.

Recentemente, ao treinar um conjunto de dados com o AutoML no Azure, deparei-me com vários problemas ao tentar fazer funcionar a inferência local. O Azure tem uma implantação "sem código" com o AutoML, e vários tipos de modelos treinados suportam esse tipo de implantação. Isto significa que não há necessidade de escrever uma única linha de código para criar a inferência e servir as respostas. O Azure trata da documentação da API que explica quais são as entradas e as saídas esperadas. Não consegui encontrar o *script de pontuação* para o modelo treinado e não consegui encontrar nenhuma dica sobre o script de pontuação que me ajudasse a executá-lo localmente. Não há uma maneira óbvia de entender como carregar e interagir com o modelo.

O sufixo do modelo implicava que ele estava usando o módulo `pickle` do Python, então, depois de tentar algumas coisas diferentes, consegui carregá-lo, mas não consegui fazer nenhuma inferência nele. A seguir, tive que lidar com as dependências. O AutoML no Azure não anuncia as versões exactas e as bibliotecas utilizadas para treinar um modelo. Atualmente estou usando Python 3.8, mas não consegui instalar o SDK do Azure no meu sistema porque o SDK só suporta 3.7. Tive de instalar o Python 3.7, depois criei um ambiente virtual e instalei o SDK nesse ambiente.

Uma das bibliotecas(*xgboost*) tinha uma compatibilidade não retroactiva para as suas versões mais recentes (os módulos foram movidos ou renomeados), pelo que tive de adivinhar uma que permitisse uma importação específica. Descobriu-se que era a 0.90 de 2019. Finalmente, ao treinar um modelo com o AutoML no Azure, parece usar a versão mais recente do SDK do Azure *no momento em que o modelo é treinado*. Mas isso também não é anunciado. Ou seja, se um modelo for treinado em janeiro e estiveres a tentar utilizá-lo um mês mais tarde, com algumas versões do SDK pelo meio, não podes utilizar a versão mais recente do SDK. Tens de voltar atrás e encontrar a versão mais recente do SDK quando o Azure treinou o modelo.

Esta situação não pretende de forma alguma ser uma descrição demasiado crítica do AutoML do Azure. A plataforma é excelente de utilizar e poderia melhorar se publicitasse melhor quais as versões utilizadas e como interagir com um modelo localmente. O problema predominante é que perdes a granularidade do controlo no processo: pouco código ou nenhum código é excelente para a velocidade, mas pode tornar-se complicado para a portabilidade. Enfrentei todos estes problemas ao tentar fazer inferências locais, mas o mesmo pode acontecer se estiveres a utilizar o Azure para a aprendizagem automática em geral, mas a tua empresa estiver a utilizar a AWS para alojamento.

Outra situação problemática que acontece frequentemente é que os cientistas que produzem um modelo numa plataforma têm de fazer suposições, como o ambiente subjacente, que inclui potência de computação, armazenamento e memória. O que acontece quando um

modelo que tem um bom desempenho na AWS precisa de ser implementado num dispositivo TPU de ponta que é totalmente incompatível? Vamos supor, por um segundo, que a tua empresa já tem esta situação resolvida e produz o mesmo modelo para diferentes plataformas. Ainda assim, o modelo resultante para o dispositivo de ponta é de cinco gigabytes, o que está além da capacidade máxima de armazenamento do acelerador.

A interoperabilidade de modelos resolve estes problemas descrevendo abertamente as restrições, facilitando a *transformação* de *modelos* de um formato para o outro, ao mesmo tempo que beneficia do apoio de todos os fornecedores de Cloud proeminentes. Na próxima secção, abordarei em pormenor o que faz do ONNX um projeto sólido para uma maior interoperabilidade e como podes criar automação para transformar modelos sem esforço.

ONNX: Troca aberta de redes neurais

Como já referi, o ONNX não só é uma excelente escolha para a interoperabilidade de modelos, como também é a primeira iniciativa para um sistema que permita mudar facilmente de frameworks. O projeto teve início em 2017, quando o Facebook e a Microsoft apresentaram o ONNX como um ecossistema aberto para a interoperabilidade de modelos de IA e desenvolveram conjuntamente o projeto e as ferramentas para impulsionar a adoção. Desde então, o projeto cresceu e amadureceu como um grande projeto de código aberto com uma estrutura estabelecida que inclui SIGs (Grupos de Interesse Especial) e grupos de trabalho para diferentes áreas, como lançamentos e formação.

Para além da interoperabilidade, a uniformidade da estrutura permite que os fornecedores de hardware visem o ONNX e tenham impacto em várias outras estruturas ao mesmo tempo. Ao aproveitar a representação do ONNX, as optimizações deixam de ter de ser integradas individualmente em cada estrutura (um processo moroso). Embora o ONNX seja relativamente recente, é revigorante vê-lo bem suportado pelos provedores

de Cloud. Não é de admirar que o Azure ofereça suporte nativo para modelos ONNX no seu SDK de aprendizagem automática.

A ideia principal é treinar uma vez na tua estrutura preferida e executar em qualquer lugar: desde a Cloud até à periferia. Quando o modelo estiver no formato ONNX, podes implementá-lo em vários dispositivos e plataformas. Isto inclui também diferentes sistemas operativos. O esforço para que isto aconteça é substancial. Não te lembra de muitos exemplos de software que possam ser executados em vários sistemas operativos diferentes, dispositivos de ponta e na Cloud, todos utilizando o mesmo formato.

Embora existam várias estruturas de aprendizagem automática suportadas (sendo adicionadas mais constantemente), a Figura 10-2 mostra o padrão de transformação mais comum.

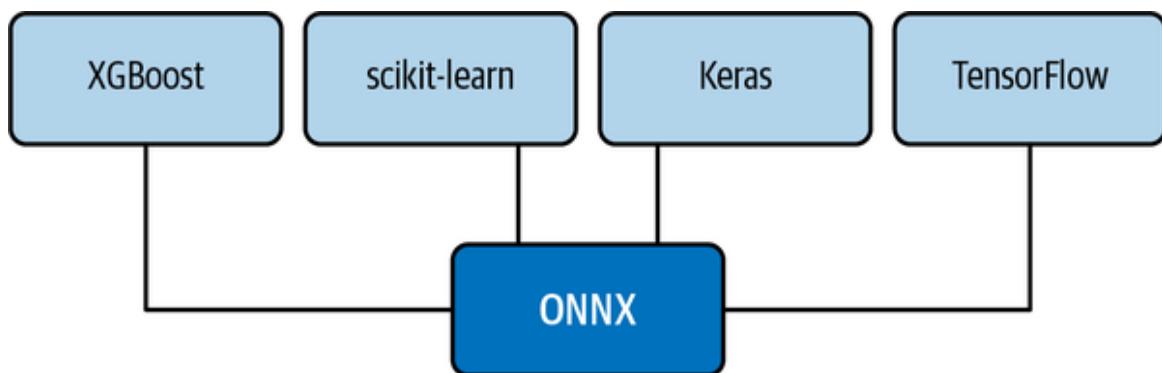


Figura 10-2. Converte em ONNX

Aproveita o conhecimento e as funcionalidades da estrutura que mais gosta e depois transforma-a em ONNX. No entanto, como demonstro em "[Apple Core ML](#)", também é possível (embora não seja muito comum) converter um modelo ONNX num tempo de execução diferente. Estas transformações também não são "gratuitas": podes ter problemas quando as novas funcionalidades ainda não são suportadas (os conversores ONNX estão sempre a atualizar-se), ou quando os modelos mais antigos não são suportados pelas versões mais recentes. Continuo a acreditar que a ideia de uniformidade e de "correr em qualquer lugar" é sólida e que é útil tirar partido dela sempre que possível.

A seguir, vamos ver onde podes encontrar modelos ONNX pré-treinados, para experimentares alguns deles.

ONNX Modelo Zoo

O *Model Zoo* é frequentemente referido quando se fala de modelos ONNX. Embora seja normalmente descrito como um registo de modelos ONNX prontos a usar, é principalmente um **repositório informativo no GitHub** que tem ligações a vários modelos pré-treinados contribuídos pela comunidade e com curadoria no repositório. Os modelos estão separados em três categorias: visão, linguagem e tudo o resto. Se queres começar a usar o ONNX e fazer algumas inferências, o Model Zoo é o sítio certo.

Em "**Packaging for ML Models**", utilizei o modelo *RoBERTa-SequenceClassification* do Model Zoo. Como eu queria me registrar no Azure, precisei adicionar algumas informações, como a versão de tempo de execução do ONNX. **A Figura 10-3** mostra como tudo isso está disponível no Model Zoo para esse modelo específico.

Model					
Model	Download	Download (with sample test data)	ONNX version	Opset version	Accuracy
RoBERTa-BASE	499 MB	295 MB	1.6	11	88.5
RoBERTa-SequenceClassification	499 MB	432 MB	1.6	9	MCC of 0.85

Figura 10-3. Zoo modelo

Para além da informação sobre a versão e o tamanho, a página terá normalmente alguns exemplos sobre como interagir com o modelo, o que é crucial se quiseres criar uma prova de conceito para o experimentar rapidamente. Há uma outra coisa que eu acho que vale a pena notar nestas páginas de documentação: obter uma *proveniência* (uma fonte de verdade para o modelo). No caso do modelo *RoBERTa-SequenceClassification*, ele vem do *PyTorch RoBERTa*, depois para o ONNX e, finalmente, é disponibilizado no Model Zoo.

Não é imediatamente claro por que razão é essencial conhecer a origem dos modelos e das fontes de trabalho. Sempre que são necessárias alterações ou

detectados problemas que precisam de ser resolvidos, é melhor estares pronto para identificar a fonte da verdade, para que tudo o que precisa de ser modificado possa ser feito com confiança. Quando eu era gerente de lançamento de um grande projeto de código aberto, eu era responsável pela construção de RPM e outros tipos de pacotes para diferentes distribuições Linux. Um dia, o repositório de produção foi corrompido e pediram-me para reconstruir estes pacotes. Enquanto os reconstruía, não conseguia encontrar que script, pipeline ou plataforma de CI estava a produzir um destes pacotes incluídos em várias dezenas destes repositórios.

Depois de rastrear os vários passos envolvidos para encontrar a origem daquele pacote, descobri que um script estava baixando-o do diretório pessoal de um desenvolvedor (há muito tempo fora da empresa) em um servidor que não tinha nada a ver com a construção de pacotes. Um único arquivo em um diretório pessoal em um servidor que não tinha responsabilidade pelo sistema de compilação é uma bomba-relógio. Eu não poderia dizer qual era a origem do pacote, como fazer qualquer atualização nele, ou a razão pela qual ele precisava ser incluído desta forma. Estas situações não são incomuns. Deves estar preparado para ter tudo em ordem e ter uma resposta sólida ao determinar a fonte da verdade de todos os elementos no teu pipeline de produção.

Quando estiveres a obter modelos de locais como o Model Zoo, certifica-te de que captas o máximo de informações possível e inclui-as onde quer que o destino seja para estes modelos. O Azure tem vários campos que podes utilizar para este fim ao registrar modelos. Como verás nas secções seguintes, alguns dos transformadores de modelos permitem adicionar metadados. Tirar partido desta tarefa aparentemente sem importância pode ser fundamental para a depuração de problemas de produção. Duas práticas benéficas que reduziram o tempo de depuração e aceleraram a integração e a facilidade de manutenção são a utilização de nomes significativos e o máximo de metadados possível. A utilização de um nome com significado é crucial para a identificação e para proporcionar clareza. Um modelo registado como "production-model-1" não me diz o que é ou do que se trata.

Se o juntassem sem metadados ou informações adicionais, isso causará frustração e atrasos na resolução de um problema de produção.

Converte PyTorch em ONNX

Começar a utilizar uma estrutura diferente é sempre assustador, mesmo quando a tarefa subjacente é treinar um modelo a partir de um conjunto de dados. O PyTorch faz um excelente trabalho ao incluir modelos pré-treinados que podem rapidamente ajudar-te a começar, uma vez que podes experimentar diferentes aspectos da estrutura sem teres de lidar com a curadoria de um conjunto de dados e depois descobrir como treiná-lo. Muitas outras estruturas (como o TensorFlow e o scikit-learn) estão a fazer o mesmo, e é uma excelente forma de começar a aprender. Nesta secção, utilizo um modelo de visão pré-treinado do PyTorch e depois exporto-o para o ONNX.

Cria um novo ambiente virtual e um ficheiro *requirements.txt* com o seguinte aspeto:

```
numpy==1.20.1
onnx==1.8.1
Pillow==8.1.2
protobuf==3.15.6
six==1.15.0
torch==1.8.0
torchvision==0.9.0
typing-extensions==3.7.4.3
```

Instala as dependências e, em seguida, cria um ficheiro *convert.py* para produzir primeiro o modelo PyTorch:

```
import torch
import torchvision

dummy_tensor = torch.randn(8, 3, 200, 200)

model = torchvision.models.resnet18(pretrained=True)

input_names = [ "input_%d" % i for i in range(12) ]
output_names = [ "output_1" ]
```

```
torch.onnx.export(  
    model,  
    dummy_tensor,  
    "resnet18.onnx",  
    input_names=input_names,  
    output_names=output_names,  
    opset_version=7,  
    verbose=True,  
)
```

Vamos passar por algumas das etapas que o script Python percorre para produzir um modelo ONNX. Cria um tensor preenchido com números aleatórios usando três canais (crucial para o modelo pré-treinado). Em seguida, recupera o modelo pré-treinado *resnet18* disponível na biblioteca *torchvision*. Define algumas entradas e saídas e, finalmente, exporta o modelo com toda essa informação.

O exemplo é demasiado simplista para provar um ponto. O modelo exportado não é de todo robusto e está cheio de valores fictícios que não são significativos. A ideia é demonstrar como o PyTorch te permite exportar o modelo para ONNX de uma forma simples. O facto de o conversor fazer parte da estrutura é tranquilizador porque é responsável por garantir que isto funciona sem falhas. Apesar de existirem bibliotecas e projectos de conversores separados, prefiro as frameworks que oferecem a conversão, como o PyTorch.

NOTA

O argumento `opset_version` na função `export()` é crítico. A indexação tensorial do PyTorch pode colocar-te em problemas com uma versão de opset ONNX não suportada. Alguns tipos de indexadores não suportam nada além da versão 12 (a última versão). Verifica sempre duas vezes se as versões correspondem às funcionalidades suportadas de que necessitas.

Executa o script *convert.py*, que cria um ficheiro *resnet18.onnx*. Deves ver um resultado semelhante a este:

```
$ python convert.py
graph(%learned_0 : Float(8, 3, 200, 200, strides=[120000, 40000,
200, 1],
      requires_grad=0, device/cpu),
      %fc.weight : Float(1000, 512, strides=[512, 1],
      requires_grad=1, device/cpu),
      %fc.bias : Float(1000, strides=[1], requires_grad=1,
      device/cpu),
      %193 : Float(64, 3, 7, 7, strides=[147, 49, 7, 1],
      requires_grad=0,
```

Agora que o modelo ONNX está disponível e é produzido pelo script usando o PyTorch, vamos usar o framework ONNX para verificar se o modelo produzido é compatível. Cria um novo script chamado *check.py*:

```
import onnx

# Load the previously created ONNX model
model = onnx.load("resnet18.onnx")

onnx.checker.check_model(model)

print(onnx.helper.printable_graph(model.graph))
```

Executa o script *check.py* a partir do mesmo diretório onde se encontra o *resnet18.onnx* e verifica se o resultado é semelhante a este:

```
$ python check.py
graph torch-jit-export (
  %learned_0[FLOAT, 8x3x200x200]
) optional inputs with matching initializers (
  %fc.weight[FLOAT, 1000x512]
[...]
  %189 = GlobalAveragePool(%188)
  %190 = Flatten[axis = 1](%189)
  %output_1 = Gemm[alpha = 1, beta = 1, transB = 1](%190,
  %fc.weight, %fc.bias)
  return %output_1
}
```

A verificação da chamada à função `check_model()` não deve produzir quaisquer erros, provando que a conversão tem algum nível de correção.

Para garantir totalmente que o modelo convertido está correto, a inferência precisa ser avaliada, capturando qualquer possível desvio. Se não tiveres a certeza sobre quais as métricas a utilizar ou como criar uma estratégia de comparação sólida, consulta a secção "["Noções básicas de monitorização de modelos"](#)". A seguir, vamos ver como podemos usar o mesmo padrão de verificação em uma ferramenta de linha de comando.

Cria um verificador genérico ONNX

Agora que já te expliquei os pormenores da exportação de um modelo do PyTorch para o ONNX e de ter um passo para verificar, vamos criar uma ferramenta simples e genérica que possa verificar qualquer modelo ONNX, não apenas um em particular. Embora dediquemos uma grande secção sobre a construção de ferramentas de linha de comandos poderosas no próximo capítulo (ver "["Ferramentas de linha de comandos"](#)" em particular), podemos tentar construir algo que funcione bem para este caso de utilização. Outro conceito que vem do DevOps e da minha experiência como administrador de sistemas é tentar a automação sempre que possível e começar com o problema mais simples primeiro. Para este exemplo, não usarei nenhuma estrutura de ferramenta de linha de comando ou qualquer análise avançada.

Primeiro, cria um novo ficheiro chamado *onnx-checker.py* com uma única função `main()`:

```
def main():
    help_menu = """
    A command line tool to quickly verify ONNX models using
    check_model()
    """
    print(help_menu)

if __name__ == '__main__':
    main()
```

Executa o script, e o resultado deve mostrar o menu de ajuda:

```
$ python onnx-checker.py
```

```
  A command line tool to quickly verify ONNX models using
  check_model()
```

O script ainda não está a fazer nada de especial. Usa a função `main()` para produzir o menu de ajuda e uma muleta muito usada em Python para chamar uma função específica quando o Python executa o script no terminal. De seguida, precisamos de lidar com entradas arbitrárias. As estruturas de ferramentas de linha de comando ajudam nisto, sem dúvida, mas ainda podemos ter algo valioso com o mínimo de esforço. Para verificar os argumentos do script (vamos precisar deles para saber que modelo verificar), precisamos de utilizar o módulo `sys.argv`. Atualiza o script, para que ele importe o módulo e o passe para a função:

```
import sys

def main(arguments):
    help_menu = """
        A command line tool to quickly verify ONNX models using
        check_model()
    """

    if "--help" in arguments:
        print(help_menu)

if __name__ == '__main__':
    main(sys.argv)
```

A alteração fará com que o script produza o menu de ajuda apenas quando utiliza a bandeira `--help`. O script ainda não está a fazer nada de útil, por isso vamos atualizar a função `main()` mais uma vez para incluir a funcionalidade de verificação ONNX:

```
import sys
import onnx

def main(arguments):
    help_menu = """
        A command line tool to quickly verify ONNX models using
        check_model()
```

```

"""
if "--help" in arguments:
    print(help_menu)
    sys.exit(0)

model = onnx.load(arguments[-1])
onnx.checker.check_model(model)
print(onnx.helper.printable_graph(model.graph))

```

Há duas alterações cruciais na função. Primeiro, chama agora `sys.exit(0)` após a verificação do menu de ajuda para impedir a execução do bloco de código seguinte. Em seguida, se a condição de ajuda não for atendida, usa o último argumento (seja ele qual for) como o caminho do modelo a ser verificado. Finalmente, usa as mesmas funções da estrutura ONNX para a verificação do modelo. Nota que não há qualquer saneamento ou verificação de inputs. Este é um script muito frágil, mas ainda se mostra útil se o executares:

```

$ python onnx-checker.py ~/Downloads/roberta-base-11.onnx
graph torch-jit-export (
    %input_ids[INT64, batch_sizexseq_len]
) initializers (
    %1621[FLOAT, 768x768]
    %1622[FLOAT, 768x768]
    %1623[FLOAT, 768x768]
    [...]
    %output_2 = Tanh(%1619)
    return %output_1, %output_2
}

```

O caminho que usei é para o modelo base RoBERTa que está em um caminho separado no meu diretório *Downloads*. Este tipo de automação é um tijolo de construção: a abordagem mais simples possível para fazer uma verificação rápida que pode ser aproveitada mais tarde noutra automação, como um sistema CI/CD ou um pipeline num fluxo de trabalho de um fornecedor Cloud. Agora que já experimentámos alguns modelos, vamos ver como converter modelos criados noutras estruturas populares em ONNX.

Converte TensorFlow em ONNX

Existe um projeto dedicado a fazer conversões de modelos do TensorFlow no repositório ONNX GitHub. Oferece uma vasta gama de versões suportadas tanto do ONNX como do TensorFlow. Mais uma vez, é fundamental garantir que qualquer ferramenta que escolhas tem as versões que o teu modelo requer para conseguir uma conversão bem sucedida para o ONNX.

Encontrar o projeto, biblioteca ou ferramenta certa para fazer conversões pode ser complicado. Para o TensorFlow especificamente, podes usar o [onnxmltools](#), que tem uma função `onnxmltools.convert_tensorflow()`, ou o projeto [tensorflow-onnx](#), que tem duas formas de fazer uma conversão: com uma ferramenta de linha de comandos ou usando a biblioteca.

Esta secção usa o projeto *tensorflow-onnx* com um módulo Python que podes usar como uma ferramenta de linha de comandos. O projeto permite converter modelos das duas versões principais do TensorFlow (1 e 2), bem como do *tflite* e do *tf.keras*. O amplo suporte do opset ONNX é excelente (da versão 7 à 13) porque permite uma maior flexibilidade ao planear uma estratégia de conversão.

Antes de entrar na conversão propriamente dita, vale a pena explorar a forma de invocar o conversor. O projeto *tf2onnx* usa um atalho Python para expor uma ferramenta de linha de comandos a partir de um ficheiro em vez de empacotar uma ferramenta de linha de comandos com o projeto. Isto significa que a invocação requer que uses o executável Python com uma bandeira especial. Começa por instalar a biblioteca num novo ambiente virtual. Cria um ficheiro *requirements.txt* para garantir que todas as versões adequadas para este exemplo funcionarão:

```
certifi==2020.12.5
chardet==4.0.0
flatbuffers==1.12
idna==2.10
numpy==1.20.1
onnx==1.8.1
```

```
protobuf==3.15.6
requests==2.25.1
six==1.15.0
tf2onnx==1.8.4
typing-extensions==3.7.4.3
urllib3==1.26.4
tensorflow==2.4.1
```

Agora usa `pip` para instalar todas as dependências nas suas versões fixadas:

```
$ pip install -r requirements.txt
Collecting tf2onnx
[...]
Installing collected packages: six, protobuf, numpy, typing-
extensions,
onnx, certifi, chardet, idna, urllib3, requests, flatbuffers,
tf2onnx
Successfully installed numpy-1.20.1 onnx-1.8.1 tf2onnx-1.8.4 ...
```

NOTA

Se instalares o projeto `tf2onnx` sem o ficheiro `requirements.txt`, a ferramenta não funcionará porque não lista `tensorflow` como um requisito. Para os exemplos nesta secção, estou a utilizar `tensorflow` na versão 2.4.1. Certifica-te de que a instalas para evitar problemas de dependência.

Corre o menu de ajuda para verificar o que está disponível. Lembra-te, a invocação parece um pouco não convencional porque precisa do executável Python para a usar:

```
$ python -m tf2onnx.convert --help
usage: convert.py [...]
Convert tensorflow graphs to ONNX.

[...]
Usage Examples:
python -m tf2onnx.convert --saved-model saved_model_dir --output
```

```

model.onnx
python -m tf2onnx.convert --input frozen_graph.pb --inputs X:0 \
    --outputs output:0 --output model.onnx
python -m tf2onnx.convert --checkpoint checkpoint.meta --inputs
X:0 \
    --outputs output:0 --output model.onnx

```

Estou a omitir algumas secções do menu de ajuda por brevidade. Chamar o menu de ajuda é uma maneira confiável de garantir que a biblioteca possa ser carregada após a instalação. Isto não seria possível se tensorflow não estivesse instalado, por exemplo. Deixei os três exemplos do menu de ajuda porque são os que vais precisar, dependendo do tipo de conversão que estás a fazer. Nenhuma destas conversões é simples, a menos que tenhas um bom conhecimento dos componentes internos do modelo que estás a tentar converter. Vamos começar com uma conversão que não requer nenhum conhecimento do modelo, permitindo que a conversão funcione imediatamente.

Primeiro, transfere o modelo [ssd_mobilenet_v2](#) (comprimido num ficheiro *tar.gz*) a partir do *tfhub*. Depois, cria um diretório e descomprime-o aí:

```

$ mkdir ssd
$ cd ssd
$ mv ~/Downloads/ssd_mobilenet_v2_2.tar.gz .
$ tar xzvf ssd_mobilenet_v2_2.tar.gz
x ./
x ./saved_model.pb
x ./variables/
x ./variables/variables.data-00000-of-00001
x ./variables/variables.index

```

Agora que o modelo descomprimido está num diretório, usa a ferramenta de conversão *tf2onnx* para transferir o *ssd_mobilenet* para o ONNX. Certifica-te de que estás a usar um opset de 13 para evitar características incompatíveis do modelo. Este é um rastreio de exceção abreviado que podes experimentar quando especificas um opset não suportado:

```

File "/.../.../tf2onnx/tfonnx.py", line 294, in
tensorflow_onnx_mapping
    func(g, node, **kwargs, initialized_tables=initialized_tables,

```

```

    ...)
File "/.../.../tf2onnx/onnx_opset/tensor.py", line 1130, in
version_1
    k = node.inputs[1].get_tensor_value()
File "/.../.../tf2onnx/graph.py", line 317, in get_tensor_value
    raise ValueError("get tensor value: '{}' must be
Const".format(self.name))
ValueError: get tensor value:
'StatefulPartitionedCall/.../SortByField/strided_slice_1738'
must be Const

```

Utiliza o sinalizador `--saved-model` com o caminho para onde o modelo foi extraído para que a conversão funcione finalmente. Neste caso, estou a utilizar o opset 13:

```

$ python -m tf2onnx.convert --opset 13 \
    --saved-model /Users/alfredo/models/ssd --output ssd.onnx
2021-03-24 - WARNING - '--tag' not specified for saved_model.
Using --tag serve
2021-03-24 - INFO - Signatures found in model: [serving_default].
2021-03-24 - INFO - Using tensorflow=2.4.1, onnx=1.8.1,
tf2onnx=1.8.4/cd55bf
2021-03-24 - INFO - Using opset <onnx, 13>
2021-03-24 - INFO - Computed 2 values for constant folding
2021-03-24 - INFO - folding node using tf type=Select,
    name=StatefulPartitionedCall/Postprocessor/.../Select_1
2021-03-24 - INFO - folding node using tf type=Select,
    name=StatefulPartitionedCall/Postprocessor/.../Select_8
2021-03-24 - INFO - Optimizing ONNX model
2021-03-24 - INFO - After optimization: BatchNormalization -53
(60->7), ...
    Successfully converted TensorFlow model
/Users/alfredo/models/ssd to ONNX
2021-03-24 - INFO - Model inputs: ['input_tensor:0']
2021-03-24 - INFO - Model outputs: ['detection_anchor_indices',
...]
2021-03-24 - INFO - ONNX model is saved at ssd.onnx

```

Estes exemplos podem parecer demasiado simplistas, mas a ideia aqui é que são blocos de construção para que possas explorar mais automação, sabendo o que é possível nas conversões. Agora que já demonstrei o que é necessário para converter um modelo TensorFlow, vamos ver o que é

necessário para converter um modelo *tflite*, outro dos tipos suportados pelo *tf2onnx*.

Descarrega uma versão quantizada *tflite* do modelo *mobilenet* a partir do **tflite**. O suporte do *tflite* no *tf2onnx* torna a invocação ligeiramente diferente. Este é um daqueles casos em que uma ferramenta é criada seguindo um critério (converter modelos TensorFlow para ONNX) e depois tem de dar suporte a outra coisa que não se encaixa no mesmo padrão. Neste caso, tens de usar o sinalizador `--tflite`, que deve apontar para o ficheiro descarregado:

```
$ python -m tf2onnx.convert \
--tflite ~/Downloads/mobilenet_v2_1.0_224_quant.tflite \
--output mobilenet_v2_1.0_224_quant.onnx
```

Rapidamente volto a ter problemas ao executar o comando, porque o conjunto de opções suportado não corresponde ao padrão. Além disso, este modelo é quantizado, o que constitui outra camada que o conversor tem de resolver. Aqui tens outro pequeno excerto de um traceback de uma tentativa:

```
File "/.../.../tf2onnx/tfonnx.py", line 294, in
tensorflow_onnx_mapping
    func(g, node, **kwargs,
initialized_tables=initialized_tables, dequantize)
  File "/.../.../tf2onnx/tflite_handlers/tfl_math.py", line 96,
in version_1
    raise ValueError
ValueError: \
Opset 10 is required for quantization.
Consider using the --dequantize flag or --opset 10.
```

Pelo menos desta vez o erro indica que o modelo é quantizado e que devo considerar a utilização de um opset diferente (em vez do opset predefinido, que claramente não funciona).

DICA

Diferentes operações do TensorFlow têm suporte variável para o ONNX e, às vezes, podem criar problemas se a versão incorreta for usada. A [página Status de suporte do tf2onnx](#) pode ser útil ao tentar determinar qual é a versão correta a ser usada.

Normalmente, fico muito desconfiado quando um livro ou demo mostra perfeição o tempo todo. Há um valor tremendo em rastreamentos, erros e em ter problemas - o que está acontecendo comigo ao tentar fazer o *tf2onnx* funcionar corretamente. Se os exemplos neste capítulo te mostrarem como tudo "simplesmente funciona", irás sem dúvida pensar que existe uma lacuna significativa de conhecimento ou que as ferramentas estão a falhar, não dando qualquer oportunidade de perceber porque é que as coisas não estão a funcionar corretamente. Eu adiciono esses rastreamentos e erros porque o *tf2onnx* tem um grau mais alto de complexidade que me permite chegar a um estado quebrado sem muito esforço.

Vamos corrigir a invocação e dar-lhe um opset de 13 (o maior offset suportado neste momento), e tenta novamente:

```
$ python -m tf2onnx.convert --opset 13 \
--tflite ~/Downloads/mobilenet_v2_1.0_224_quant.tflite \
--output mobilenet_v2_1.0_224_quant.onnx

2021-03-23 INFO - Using tensorflow=2.4.1, onnx=1.8.1,
tf2onnx=1.8.4/cd55bf
2021-03-23 INFO - Using opset <onnx, 13>
2021-03-23 INFO - Optimizing ONNX model
2021-03-23 INFO - After optimization: Cast -1 (1->0), Const -307
(596->289)...
2021-03-23 INFO - Successfully converted TensorFlow model \
~/Downloads/mobilenet_v2_1.0_224_quant.tflite
to ONNX
2021-03-23 INFO - Model inputs: ['input']
2021-03-23 INFO - Model outputs: ['output']
2021-03-23 INFO - ONNX model is saved at
mobilenet_v2_1.0_224_quant.onnx
```

Finalmente, o modelo *tflite* quantizado é convertido para ONNX. Há margem para melhorias e, tal como vimos nos passos anteriores desta secção, é fundamental conheceres bem as entradas e saídas do modelo e a forma como o modelo foi criado. Este conhecimento é crucial no momento da conversão, onde podes fornecer à ferramenta o máximo de informação possível para garantir um resultado bem sucedido. Agora que já converti alguns modelos para ONNX, vamos ver como os implementar com o Azure.

Implementa o ONNX no Azure

O Azure tem uma excelente integração do ONNX na sua plataforma, com suporte direto no seu Python SDK. Podes criar uma experiência para treinar um modelo utilizando o PyTorch que pode depois ser exportado para o ONNX e, como te mostrarei nesta secção, podes implementar esse modelo ONNX num cluster para inferências em tempo real. Esta secção não abordará a forma de realizar o treino real do modelo; no entanto, explicarei como é simples utilizar um modelo ONNX treinado que está registado no Azure e implementá-lo num cluster.

Em "[Empacotamento de modelos de ML](#)", abordei todos os detalhes necessários para colocar um modelo no Azure e, em seguida, empacotá-lo em um contêiner. Vamos reutilizar parte do código do contentor para criar o *ficheiro de pontuação*, que o Azure precisa para o implementar como um serviço Web. Na essência, é a mesma coisa: o script recebe um pedido, que sabe como traduzir as entradas para fazer uma previsão com o modelo carregado e, em seguida, devolve osvalores.

NOTA

Estes exemplos utilizam o *espaço de trabalho* do Azure, definido como um objeto `ws`. É necessário configurá-lo antes de começar. Isto é abordado em pormenor em "[Azure CLI e Python SDK](#)".

Cria o ficheiro de pontuação, chama-lhe *score.py* e adiciona uma função *init()* para carregar o modelo:

```
import torch
import os
import numpy as np
from transformers import RobertaTokenizer
import onnxruntime

def init():
    global session
    model = os.path.join(
        os.getenv("AZUREML_MODEL_DIR"), "roberta-sequence-
classification-9.onnx"
    )
    session = onnxruntime.InferenceSession(model)
```

Agora que os princípios básicos do script de pontuação estão concluídos, é necessária uma função *run()* quando executada no Azure. Atualiza o script *score.py* criando a função *run()* que entende como interagir com o modelo de classificação RoBERTa:

```
def run(input_data_json):
    try:
        tokenizer = RobertaTokenizer.from_pretrained("roberta-
base")
        input_ids = torch.tensor(
            tokenizer.encode(input_data_json[0],
            add_special_tokens=True)
        ).unsqueeze(0)

        if input_ids.requires_grad:
            numpy_func = input_ids.detach().cpu().numpy()
        else:
            numpy_func = input_ids.cpu().numpy()

        inputs = {session.get_inputs()[0].name:
        numpy_func(input_ids)}
        out = session.run(None, inputs)

        return {"result": np.argmax(out)}
    except Exception as err:
```

```
    result = str(err)
    return {"error": result}
```

Em seguida, cria a configuração para realizar a inferência. Uma vez que estes exemplos estão a usar o Python SDK, podes usá-los num Jupyter Notebook ou usando diretamente a shell Python. Começa por criar um ficheiro YAML que descreve o teu ambiente:

```
from azureml.core.conda_dependencies import CondaDependencies

environ = CondaDependencies.create(
    pip_packages=[
        "numpy", "onnxruntime", "azureml-core", "azureml-defaults",
        "torch", "transformers"
    )

    with open("environ.yml", "w") as f:
        f.write(environ.serialize_to_string())
```

Agora que o arquivo YAML está pronto, define a configuração:

```
from azureml.core.model import InferenceConfig
from azureml.core.environment import Environment

environ = Environment.from_conda_specification(
    name="environ", file_path="environ.yml"
)
inference_config = InferenceConfig(
    entry_script="score.py", environment=environ
)
```

Por fim, implanta o modelo usando o SDK. Cria primeiro a configuração da Instância de Contentor do Azure (ACI):

```
from azureml.core.webservice import AciWebservice

aci_config = AciWebservice.deploy_configuration(
    cpu_cores=1,
    memory_gb=1,
    tags={"model": "onnx", "type": "language"},
```

```
        description="Container service for the RoBERTa ONNX model",
    )
```

Com o aci_config, implementa o serviço:

```
from azureml.core.model import Model

# retrieve the model
model = Model(ws, 'roberta-sequence', version=1)

aci_service_name = "onnx-roberta-demo"
aci_service = Model.deploy(
    ws, aci_service_name,
    [model],
    inference_config,
    aci_config
)

aci_service.wait_for_deployment(True)
```

Várias coisas aconteceram para que essa implantação funcionasse.

Primeiro, definiste o ambiente com as dependências necessárias para que a inferência funcione. Depois, configuraste uma Instância de Contentor Azure e, finalmente, recuperaste a versão 1 do modelo ONNX *roberta_sequence* e utilizaste o método `Model.deploy()` do SDK para implementar o modelo. Mais uma vez, as especificidades de treinamento do modelo não são abordadas aqui. Podes muito bem treinar qualquer modelo no Azure, exportá-lo para o ONNX, registá-lo e retomá-lo nesta secção para continuar o processo de implementação. Poucas modificações são necessárias nestes exemplos para progredires. Talvez sejam necessárias bibliotecas diferentes e certamente uma forma diferente de interagir com o modelo. Ainda assim, este fluxo de trabalho permite-te adicionar outra camada de automatização para implementar modelos de forma programática do PyTorch para o ONNX (ou diretamente de modelos ONNX previamente registados) numa instância de contentor no Azure.

Vais querer utilizar o ONNX para implementar outros ambientes que não a nuvem, como dispositivos móveis, em algumas outras situações. Abordarei

alguns dos detalhes envolvidos na próxima secção com a estrutura de aprendizagem automática da Apple.

Apple Core ML

A estrutura de aprendizado de máquina da Apple é um pouco única, pois há suporte para converter modelos do Core ML em ONNX , *bem como* convertê-los de ONNX para Core ML. Como já referi neste capítulo, tens de ter muito cuidado e garantir que existe suporte para a conversão de modelos e para obter as versões corretas. Atualmente, o *coremltools* suporta as versões 10 e mais recentes do ONNX opset. Não é muito difícil chegar a uma situação em que um modelo não tenha suporte e ocorra uma quebra. Para além do suporte ONNX, tens de ter em atenção o ambiente de conversão de destino e se esse ambiente suporta versões iOS e macOS.

DICA

Vê o [alvo mínimo](#) suportado para diferentes ambientes na documentação do *Core ML*.

Para além do suporte num ambiente-alvo como o iOS, existe também uma boa lista de modelos testados da ONNX que funcionam bem. É dessa lista que vou escolher o modelo MNIST para experimentar uma conversão. Go to the Model Zoo e encontra a secção MNIST. [Descarrega](#) a versão mais recente (1.3, no meu caso). Agora cria um novo ambiente virtual e um *requirements.txt* com as seguintes bibliotecas, que incluem o *coremltools*:

```
attr==0.3.1
attrs==20.3.0
coremltools==4.1
mpmath==1.2.1
numpy==1.19.5
onnx==1.8.1
packaging==20.9
protobuf==3.15.6
pyparsing==2.4.7
scipy==1.6.1
```

```
six==1.15.0
sympy==1.7.1
tqdm==4.59.0
typing-extensions==3.7.4.3
```

Instala as dependências para que possamos criar ferramentas para fazer a conversão. Vamos criar a ferramenta mais simples possível, tal como em "[Criar um verificador ONNX genérico](#)", sem analisadores de argumentos ou quaisquer menus de ajuda extravagantes. Começa por criar a função `main()` e a magia especial Python necessária no final do ficheiro para que a possas chamar no terminal como um script:

```
import sys
from coremltools.converters.onnx import convert

def main(arguments):
    pass

if __name__ == '__main__':
    main(sys.argv)
```

Neste caso, o script ainda não está a fazer nada de útil e também não estou a implementar um menu de ajuda. Deves sempre incluir um menu de ajuda nos teus scripts para que os outros possam ter alguma ideia das entradas e saídas do programa quando precisarem de interagir com ele. Actualiza a função `main()` para experimentares a conversão. Assumirei que o último argumento recebido representará um caminho para o modelo ONNX que precisa de ser convertido:

```
def main(arguments):
    model_path = arguments[-1]
    basename = model_path.split('.onnx')[0]
    model = convert(model_path,
                    minimum_ios_deployment_target='13')
    model.short_description = "ONNX Model converted with
coremltools"
    model.save(f"{basename}.mlmodel")
```

Primeiro, a função capta o último argumento como o caminho para o modelo ONNX e, em seguida, calcula o nome de base retirando o sufixo

.onnx. Finalmente, faz a conversão (usando um alvo mínimo de versão iOS de 13), inclui uma descrição e guarda a saída. Experimenta o script atualizado com o modelo MNIST transferido anteriormente:

```
$ python converter.py mnist-8.onnx
1/11: Converting Node Type Conv
2/11: Converting Node Type Add
3/11: Converting Node Type Relu
4/11: Converting Node Type MaxPool
5/11: Converting Node Type Conv
6/11: Converting Node Type Add
7/11: Converting Node Type Relu
8/11: Converting Node Type MaxPool
9/11: Converting Node Type Reshape
10/11: Converting Node Type MatMul
11/11: Converting Node Type Add
Translation to CoreML spec completed. Now compiling the CoreML
model.
Model Compilation done.
```

A operação resultante deverá ter produzido um ficheiro *mnist-8.mlmodel*, que é um modelo Core ML que podes agora carregar num computador macOS que tenha o XCode instalado. Utiliza o Finder no teu computador Apple, faz duplo clique no modelo *coreml* recentemente gerado e faz duplo clique. O modelo MNIST deve ser carregado imediatamente, com a descrição incluída no script do conversor, conforme mostrado na Figura 10-4.

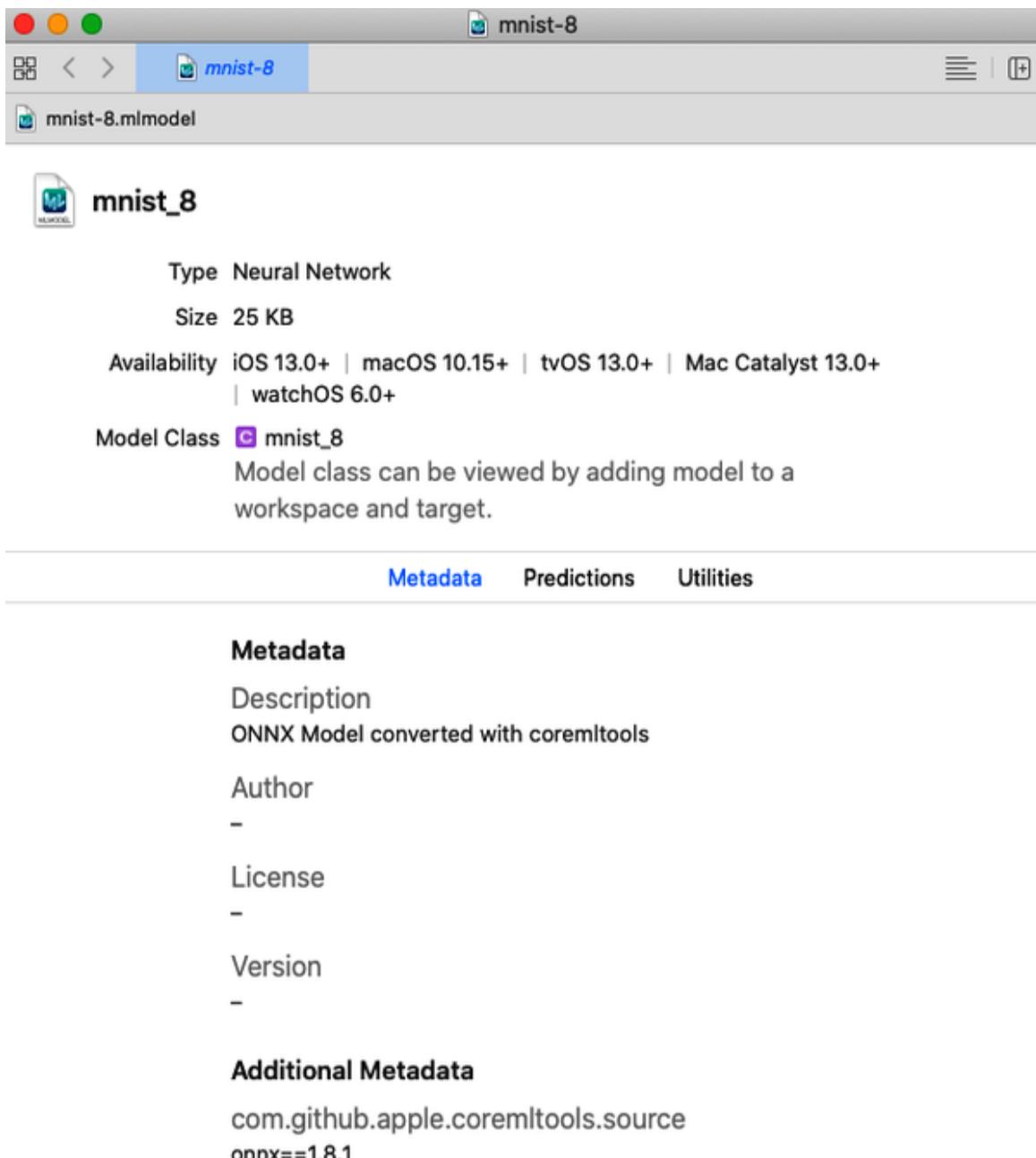


Figura 10-4. ONNX para o núcleo ML

Verifica se a secção Disponibilidade mostra os objectivos mínimos de iOS de 13, tal como o script do conversor os definiu. A secção Previsões contém informações úteis sobre as entradas e saídas que o modelo aceita, conforme apresentado na [Figura 10-5](#).

[Metadata](#) [Predictions](#) [Utilities](#)

Input

Input3
MultiArray (Float32 1 × 1 × 28 × 28)

Description

-

Output

Plus214_Output_0
MultiArray (Float32 1 × 10)

Description

-

Figura 10-5. Previsões ONNX para CoreML

Por fim, a secção Utilitários fornece ajudantes para implementar esse modelo utilizando um arquivo de modelo que se integra no CloudKit (ambiente da Apple para recursos de aplicações iOS), conforme apresentado na [Figura 10-6](#).

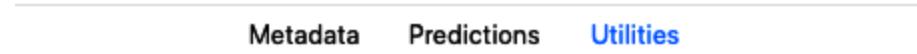


Figura 10-6. Arquivo do modelo ONNX para Core ML

É empolgante ver um amplo apoio ao ONNX e um apoio crescente noutras estruturas e sistemas operativos como, neste caso, o OSX. Se estiveres interessado no desenvolvimento iOS e na implantação de modelos, podes continuar a utilizar as estruturas a que estás habituado. Depois, podes fazer a conversão para o ambiente iOS pretendido para a implantação. Este processo é uma razão convincente para utilizar o ONNX, pois permite-te utilizar estruturas bem estabelecidas que podes transformar em ambientes específicos. De seguida, vamos ver algumas outras integrações de ponta que realçam ainda mais a utilidade da estrutura e das ferramentas ONNX.

Integração de bordas

Recentemente, a ONNX anunciou um novo formato de modelo interno conhecido como ORT, que minimiza o tamanho de construção do modelo para uma implementação óptima em dispositivos incorporados ou de ponta. O tamanho mais pequeno de um modelo tem vários aspectos que ajudam nos dispositivos de ponta. Os dispositivos de ponta têm uma capacidade de armazenamento limitada e, na maioria das vezes, o armazenamento não é

nada rápido. Ler e escrever em pequenos dispositivos de armazenamento pode rapidamente tornar-se problemático. Além disso, o ONNX, em geral, continua a tentar suportar mais e diferentes configurações de hardware com diferentes CPUs e GPUs; não é um problema fácil de resolver, mas é um esforço bem-vindo. Quanto melhor e mais amplo for o suporte, mais fácil será a implementação de modelos de aprendizagem automática para ajudar ambientes onde isso não era possível antes. Uma vez que abordei a maioria dos benefícios e aspectos cruciais do Edge em "[Dispositivos Edge](#)", esta secção irá concentrar-se na conversão de um modelo ONNX para o formato ORT.

Começa por criar um novo ambiente virtual, ativa-o e, em seguida, instala as dependências, conforme mostrado neste ficheiro *requirements.txt*:

```
flatbuffers==1.12
numpy==1.20.1
onnxruntime==1.7.0
protobuf==3.15.6
six==1.15.0
```

Atualmente, não existem ferramentas separadas disponíveis para instalação, pelo que é necessário clonar todo o repositório *onnxruntime* para tentar uma conversão para ORT. Após a clonagem, usarás o arquivo *convert_onnx_models_to_ort.py* no diretório *tools/python*:

```
$ git clone https://github.com/microsoft/onnxruntime.git
$ python onnxruntime/tools/python/convert_onnx_models_to_ort.py -
-h
usage: convert_onnx_models_to_ort.py [-h]
[...]
Convert the ONNX format model/s in the provided directory to ORT
format models.
All files with a `.onnx` extension will be processed. For each
one, an ORT
format model will be created in the same directory. A
configuration file will
also be created called `required_operators.config`, and will
contain the list
of required operators for all converted models. This
configuration file should
be used as input to the minimal build via the `--
```

```
include_ops_by_config`  
parameter.  
[...]
```

O conversor ORT produz um arquivo de configuração, bem como um arquivo de modelo ORT a partir do modelo ONNX. É possível implantar o modelo otimizado junto com uma compilação de tempo de execução ONNX exclusiva. Primeiro, tenta uma conversão com um modelo ONNX. Neste exemplo, descarreguei o modelo ONNX **mobilenet_v2-1.0** para o diretório *models/* e utilizei-o como argumento para o script do conversor:

```
$ python onnxruntime/tools/python/convert_onnx_models_to_ort.py \  
      models/mobilenetv2-7.onnx  
Converting optimized ONNX model to ORT format model  
models/mobilenetv2-7.ort  
Processed models/mobilenetv2-7.ort  
Created config in models/mobilenetv2-7.required_operators.config
```

A configuração é crucial aqui porque lista os operadores necessários ao modelo. Isto permite-te criar um tempo de execução ONNX apenas com esses operadores. Para o modelo convertido, esse ficheiro tem o seguinte aspeto:

```
ai.onnx;1;Conv,GlobalAveragePool  
ai.onnx;5;Reshape  
ai.onnx;7;Add  
com.microsoft;1;FusedConv
```

Podes conseguir uma redução de tamanho binário para o tempo de execução especificando apenas as necessidades do modelo. Não vou abordar todas as especificidades de como construir o tempo de execução do ONNX a partir do código fonte, mas podes usar o **guias de construção** como referência para os próximos passos à medida que exploramos quais as flags e opções que são úteis na redução binária.

Primeiro, tens de utilizar a bandeira `--include_ops_by_config`. Nesse caso, o valor para esse sinalizador é o caminho para a configuração gerada na etapa anterior. No meu caso, esse caminho é *models/mobilenetv2-*

`7.required_operators.config`. Também sugiro que experimentes o `--minimal_build`, que suporta apenas o carregamento e a execução de modelos ORT (eliminando o suporte para formatos ONNX normais). Por fim, se o teu objetivo for um dispositivo Android, a utilização do sinalizador `--android_cpp_shared` produzirá um binário mais pequeno ao utilizar a biblioteca partilhada `libc++` em vez da biblioteca estática que vem por defeito no tempo de execução.

Conclusão

A uniformidade que uma estrutura (e um conjunto de ferramentas) como o ONNX proporciona ajuda todo o ecossistema de aprendizagem automática. Um dos ideais deste livro é fornecer exemplos *práticos* para colocar os modelos em produção de uma forma reproduzível e fiável. Quanto mais abrangente for o suporte para modelos de aprendizagem automática, mais fácil será para os engenheiros experimentarem e tirarem partido de tudo o que a aprendizagem automática oferece. Estou particularmente entusiasmado com as aplicações de ponta, especialmente para ambientes remotos onde é impossível ligar à Internet ou ter qualquer conectividade de rede. A ONNX está a reduzir a fricção que existe para a implementação nesses ambientes. Espero que o esforço para continuar a facilitar isto, com mais ferramentas e melhor suporte, continue a beneficiar do conhecimento e das contribuições colectivas. Embora tenhamos experimentado brevemente ferramentas de linha de comandos, no próximo capítulo, entro em muito mais detalhes sobre como criar ferramentas de linha de comandos robustas com tratamento de erros e sinalizadores bem definidos. Além disso, abordo o empacotamento Python e a utilização de microsserviços, o que te dará a flexibilidade de experimentar diferentes abordagens ao resolver desafios de aprendizagem automática.

Exercícios

- Actualiza todos os scripts para verificar o modelo ONNX produzido com o script de "Create a Generic ONNX Checker".
- Modifica o script do conversor Core ML para utilizar a estrutura Click para uma melhor análise das opções e um menu de ajuda.
- Agrupa três conversores numa única ferramenta de linha de comandos para que seja fácil fazer conversões com diferentes entradas.
- Melhora o conversor *tf2onnx* de modo a que esteja envolvido num novo script, que pode detetar erros comuns e reportá-los com uma mensagem mais fácil de utilizar.
- Usa um modelo ONNX diferente para uma implantação do Azure.

Questões para discussão sobre pensamento crítico

- Porque é que a ONNX é importante? Apresenta pelo menos três razões.
- O que há de útil na criação de um script sem uma estrutura de ferramentas de linha de comando? Quais são as vantagens de utilizar uma estrutura?
- Qual é a utilidade do formato ORT? Em que situações o podes utilizar?
- Quais são alguns dos problemas que podes encontrar se a portabilidade não existir? Apresenta três razões pelas quais a resolução desses problemas melhorará a aprendizagem automática em geral.

Capítulo 11. Criando ferramentas de linha de comando e microsserviços MLOps

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Alfredo Deza

Os japoneses bombardearam Pearl Harbor a 7 de dezembro de 1941 e tornou-se imediatamente impossível comprar pneus, claramente necessários para a longa viagem até à Califórnia. O meu pai vasculhou o país à procura de pneus velhos para o Ford e para a pequena caravana que tínhamos usado em várias viagens familiares mais curtas. Partimos de Cincinnati em fevereiro de 1942 com 22 pneus amarrados aos tectos do atrelado e do carro e, antes de chegarmos à Califórnia, usámo-los todos.

—Dr. Joseph Bogen

Construir ferramentas de linha de comandos foi a forma como me iniciei no Python há alguns anos e acredito que é a intersecção perfeita entre o desenvolvimento de software e a aprendizagem automática. Lembro-me de todos aqueles anos atrás, quando me esforcei para aprender novos conceitos Python que me pareciam muito estranhos como Administrador de Sistemas: funções, classes, registo e testes. Como Administrador de Sistemas, estive maioritariamente exposto a shell scripting com Bash, escrevendo instruções de cima para baixo para conseguir fazer alguma coisa.

Há várias dificuldades em tentar resolver problemas com scripts de shell. Lidar com erros com relatórios simples, registo e depuração são características que não requerem muito esforço noutras linguagens como o Python. Várias outras linguagens oferecem características semelhantes, como Go e Rust, e devem dar-te uma ideia do porquê de usares algo diferente de uma linguagem de comandos como o Bash. Eu tendo a recomendar o uso de uma linguagem de script shell quando algumas linhas podem realizar a tarefa. Por exemplo, esta função de shell copia a minha chave SSH pública para uma máquina remota para as chaves autorizadas, permitindo que a minha conta acceda a esse servidor remoto sem precisar de uma palavra-passe:

```
ssh-copy-key() {  
    if [ $2 ]; then  
        key=$2  
    else  
        key="$HOME/.ssh/id_rsa.pub"  
    fi  
    cat "$key" | ssh ${1} \  
        'umask 0077; mkdir -p .ssh; cat >> .ssh/authorized_keys'  
}
```

NOTA

A função Bash de exemplo provavelmente não funcionará bem no teu ambiente. Se quiseres experimentá-la, os caminhos e os ficheiros de configuração terão de coincidir.

Fazer o mesmo com uma linguagem como o Python não faria muito sentido porque seriam necessárias muitas mais linhas de código e provavelmente algumas dependências extra instaladas. Esta função é simples, só faz uma coisa, e é muito portátil. A minha recomendação é que olhes para além de uma linguagem de scripting da shell quando a solução tem mais de uma dúzia de linhas.

DICA

Se dás por ti a criar pequenos fragmentos de comandos shell com frequência, é bom criar um repositório para os recolher. O [repositório](#) para os meus aliases, funções e configurações deve mostrar-te uma boa maneira de começar se precisares de uma referência.

A forma como aprendi Python foi automatizando tarefas aborrecidas e repetitivas com ferramentas de linha de comandos, desde criar sites de clientes utilizando modelos até adicionar e remover utilizadores dos servidores da empresa. A minha recomendação para te empenhares na aprendizagem é encontraras um problema interessante com um benefício direto para ti. Esta forma de aprender é totalmente diferente da que nos foi ensinada na escola e não se aplica necessariamente a qualquer situação, mas é uma boa opção para este capítulo.

Dependendo da forma como crias as ferramentas de linha de comandos, a sua instalação pode ser simples. Como o exemplo de um comando shell que deve funcionar em qualquer ambiente Unix com facilidade, há uma semelhança com os contêineres e os microsserviços. Como o contêiner agrupa as dependências, ele funcionará em qualquer sistema com um tempo de execução adequado ativado. Já passamos por alguns dos componentes dos microsserviços em "[Containers](#)", descrevendo algumas diferenças críticas em relação a aplicações monólitas.

Mas os microsserviços são mais do que contentores, e quase todos os fornecedores de Cloud oferecem uma solução *sem servidor*. A solução sem servidor permite que um programador se concentre em escrever pequenas aplicações sem se preocupar com o sistema operativo subjacente, as suas dependências ou o tempo de execução. Embora a oferta possa parecer demasiado simplista, podes aproveitar a solução para criar APIs HTTP completas ou um fluxo de trabalho semelhante a um pipeline. Podes ligar todos estes componentes e tecnologias à linha de comando e a algumas funcionalidades de ML dos fornecedores Cloud. O aspetto "misturar e combinar" destas tecnologias significa que um engenheiro pode criar soluções criativas para problemas complicados com muito pouco código.

Sempre que puderes automatizar tarefas e aumentar a produtividade, estarás a aplicar competências operacionais sólidas para te ajudar a colocar modelos em produção de forma robusta.

Empacotamento Python

Muitas ferramentas úteis da linha de comandos Python começam como um único ficheiro de script, e depois tendem a crescer para cenários mais complexos com outros ficheiros e talvez dependências. Não é até que o script precise de bibliotecas extras que não é mais viável manter o script sem empacotamento. O empacotamento do Python não é muito bom. Dói-me dizer isto depois de mais de uma década de experiência em Python, mas o empacotamento ainda é um aspeto do ecossistema da linguagem cheio de problemas complicados (não resolvidos).

Se estás a mexer e a experimentar alguma automação *sem dependências externas*, então um único script Python está bem sem empacotamento. Se, por outro lado, o teu script requer algumas outras dependências e talvez consista em mais do que um ficheiro, então deves sem dúvida considerar o empacotamento. Outra característica útil de uma aplicação Python devidamente empacotada é que ela pode ser publicada no [Índice de Pacotes Python](#) para que outros possam instalá-la com ferramentas como *pip* (o Instalador de Pacotes para Python).

Há alguns anos atrás, era problemático instalar pacotes Python num sistema: era impossível removê-los. Isso parece inacreditável hoje em dia, mas foi uma das muitas razões pelas quais os "ambientes virtuais" decolaram. Com ambientes virtuais, corrigir dependências era tão fácil quanto remover um diretório - enquanto mantinha os pacotes do sistema intactos. Hoje em dia, desinstalar pacotes Python é mais fácil (e possível!), mas a resolução de dependências ainda carece de robustez. Ambientes virtuais são então a forma sugerida para trabalhar em projectos Python, assim os ambientes são completamente isolados, e podes resolver problemas de dependências criando um novo ambiente.

Não deve ser surpreendente ver recomendações ao longo deste livro (e em outros lugares no ecossistema Python) para usar o módulo *virtualenv*.

Desde o Python 3, a maneira comum de criar e ativar um ambiente virtual é diretamente com o executável *Python*:

```
$ python -m venv venv  
$ source venv/bin/activate
```

Para verificar que o ambiente virtual está ativado, o executável Python deve agora ser diferente do Python do sistema:

```
$ which python  
/tmp/venv/bin/python
```

Recomendo a utilização de técnicas de empacotamento Python adequadas para que estejas bem preparado quando for necessário. Assim que a tua ferramenta de linha de comando precisar de uma dependência, ela estará pronta para ser declarada como um requisito. Os consumidores da tua ferramenta irão resolver estas dependências também, tornando mais fácil para os outros trabalharem com a tua criação.

O ficheiro de requisitos

Como verás nas próximas secções deste capítulo, existem duas formas populares de definir dependências. Uma delas é utilizar um ficheiro *requirements.txt*. As ferramentas de instalação podem usar este ficheiro como o *pip* para obter dependências instaladas a partir de um índice de pacotes. Neste arquivo, as dependências são declaradas em uma linha separada, e opcionalmente, com alguma restrição de versões. Neste exemplo, o framework Click não tem uma restrição e, portanto, o instalador (*pip*) usará a versão mais recente. O framework Pytest é *fixado* em uma versão específica, então *pip* sempre tentará encontrar essa versão específica no momento da instalação:

```
# requirements.txt
click
pytest==5.1.0
```

Para instalar dependências a partir de um ficheiro *requirements.txt*, tens de utilizar **pip**:

```
$ pip install -r requirements.txt
```

Embora não exista uma regra rígida de nomeação, podes normalmente encontrar dependências num ficheiro de texto simples chamado *requirements.txt*. Os responsáveis pelo projeto podem também definir vários ficheiros de texto com requisitos. Isso é mais comum quando as dependências de desenvolvimento são diferentes das dependências de envio para produção, por exemplo. Como verás na próxima secção, existe também um ficheiro *setup.py* que pode instalar dependências. Este é um efeito colateral infeliz do estado do empacotamento e gerenciamento de dependências em Python. Ambos os ficheiros podem atingir o objetivo de instalar dependências para um projeto Python, mas apenas o *setup.py* pode empacotar um projeto Python para distribuição. Como um arquivo *setup.py* é executado no momento da instalação pelo Python, ele permite fazer qualquer coisa além das tarefas de instalação. Eu não recomendo estender o *setup.py* para fazer qualquer coisa além de tarefas de empacotamento para evitar problemas ao distribuir uma aplicação.

Alguns projectos preferem definir as suas dependências num ficheiro *requirements.txt* e depois reutilizar o conteúdo desse ficheiro no ficheiro *setup.py*. Podes conseguir isto lendo o *requirements.txt* e usando a variável *dependencies*:

```
with open("requirements.txt", "r") as _f:
    dependencies = _f.readlines()
```

Distinguir entre esses arquivos de empacotamento e conhecer seu histórico é útil para evitar confusão e mau uso. Agora deves sentir-te mais à vontade

para discernir se um projeto se destina a distribuição(*setup.py*) ou um serviço ou projeto que não requer instalação.

Ferramentas de linha de comando

Uma das características da linguagem Python é a capacidade de criar rapidamente aplicações com quase tudo o que possas imaginar já incluído, desde o envio de pedidos HTTP ao processamento de ficheiros e texto, até à ordenação de fluxos de dados. O ecossistema de bibliotecas disponíveis é vasto. Não parece surpreendente que a comunidade científica tenha adotado o Python como uma das principais linguagens para lidar com cargas de trabalho que incluem a aprendizagem automática.

Uma excelente forma de abordar o desenvolvimento de ferramentas de linha de comandos é identificar uma situação específica que precisa de ser resolvida. Da próxima vez que te deparares com uma tarefa algo repetitiva, tenta criar uma ferramenta de linha de comandos para automatizar os passos para produzir o resultado. A automatização é outro princípio fundamental do DevOps que deves aplicar sempre que possível (e tanto quanto fizer sentido) a tarefas em todo o ML. Embora possas criar um único ficheiro Python e usá-lo como uma ferramenta de linha de comandos, os exemplos nesta secção vão usar técnicas de empacotamento adequadas que te vão permitir definir as dependências necessárias e instalar a ferramenta com instaladores Python como `pip`. No primeiro exemplo de ferramenta, vou mostrar estes padrões Python em detalhe para compreenderes as ideias por detrás das ferramentas de linha de comandos que podes aplicar ao resto deste capítulo.

Criar uma ponteira de conjunto de dados

Ao criar este livro, decidi reunir um conjunto de dados de classificações e descrições de vinhos. Não consegui encontrar nada semelhante, pelo que comecei a recolher informações para o conjunto de dados. Quando o conjunto de dados já tinha um bom número de entradas, o passo seguinte foi visualizar a informação e determinar a robustez dos dados. Como é

habitual no estado inicial dos dados, este conjunto de dados apresentava várias anomalias que exigiram algum esforço para serem corretamente identificadas.

Um dos problemas foi que, depois de carregar os dados como uma estrutura de dados do Pandas, ficou claro que uma das colunas era inutilizável: quase todas eram NaN (também chamadas de entradas nulas). Outro problema, que pode ser o pior de todos, foi que carreguei o conjunto de dados no Azure ML Studio para executar algumas tarefas AutoML que começaram a produzir alguns resultados surpreendentes. Embora o conjunto de dados tivesse seis colunas, o Azure estava a reportar cerca de quarenta.

Por fim, o *pandas* adicionou colunas sem nome ao salvar os dados processados localmente, e eu não estava ciente disso. O conjunto de dados está disponível para demonstrar os problemas. Carrega primeiro o ficheiro CSV (valor separado por vírgulas) como um quadro de dados *do pandas*:

```
import pandas as pd
csv_url = (
    "https://raw.githubusercontent.com/paiml/wine-
    ratings/main/wine-ratings.csv"
)
# set index_col to 0 to tell pandas that the first column is the
# index
df = pd.read_csv(csv_url, index_col=0)
df.head(-10)
```

A saída da tabela do pandas parece óptima, mas dá a entender que uma coluna pode estar vazia:

notes	name	grape	region	variety	rating
...
...	32765 Lewis Cella...	NaN	Napa Valley...	White Wine	92.0
Neil Young'..					
32766 Lewis Cella...	NaN	Napa Valley...	White Wine	93.0	
From the lo..					
32767 Lewis Cella...	NaN	Napa Valley...	White Wine	93.0	
Think of ou..					
32768 Lewis Cella...	NaN	Napa Valley...	Red Wine	92.0	

```
When asked ...
32769  Lewis Cella...      NaN  Napa Valley...  White Wine      90.0
The warm, v...
[32770 rows x 6 columns]
```

Ao descrever o conjunto de dados, um dos problemas é agora claro: a coluna da *uva* não tem quaisquer itens:

```
In [13]: df.describe()
Out[13]:
    grape      rating
count    0.0  32780.000000
mean     NaN   91.186608
std      NaN   2.190391
min     NaN   85.000000
25%     NaN   90.000000
50%     NaN   91.000000
75%     NaN   92.000000
max     NaN   99.000000
```

Deixa cair a coluna problemática e guarda o conjunto de dados num novo ficheiro CSV, para que possas manipular os dados sem teres de descarregar sempre o conteúdo:

```
df.drop(['grape'], axis=1, inplace=True)
df.to_csv("wine.csv")
```

Rele o ficheiro demonstra a coluna extra adicionada pelo *pandas*. Para reproduzir o problema, relê o arquivo CSV local, salva-o como um novo arquivo e olha para a primeira linha do arquivo recém-criado:

```
df = pd.read_csv('wine.csv')
df.to_csv('wine2.csv')
```

Olha para a primeira linha do ficheiro *wine2.csv* para identificar a nova coluna:

```
$ head -1 wine2.csv
,Unnamed: 0,name,region,variety,rating,notes
```

O problema do Azure estava mais envolvido e era difícil de detetar: O Azure ML estava a interpretar novas linhas e retornos de carro numa das colunas como novas colunas. Para encontrar estes caracteres especiais, tive de configurar o meu editor para os mostrar (normalmente, não são visíveis). Neste exemplo, o retorno de carro aparece como ^M:

```
"Concentrated aromas of dark stone fruits and toast burst^M
from the glass. Classic Cabernet Sauvignon flavors of^M
black cherries with subtle hints of baking spice dance^M
across the palate, bolstered by fine, round tannins. This^M
medium bodied wine is soft in mouth feel, yet long on^M
fruit character and finish."^M
```

Depois de eliminar a coluna sem itens, remover as colunas sem nome e eliminar os retornos de carruagem, os dados estavam agora num estado muito mais saudável. Agora que já fiz a limpeza devida, quero automatizar a deteção destes problemas. Provavelmente, daqui a um ano, vou esquecer-me do problema das colunas extra no Azure ou de uma coluna com valores inúteis. Vamos criar uma ferramenta de linha de comando para ingerir um arquivo CSV e produzir alguns avisos.

Cria uma nova diretoria chamada *csv-linter* e adiciona um ficheiro *setup.py* com o seguinte aspeto:

```
from setuptools import setup, find_packages

setup(
    name = 'csv-linter',
    description = 'lint csv files',
    packages = find_packages(),
    author = 'Alfredo Deza',
    entry_points="""
    [console_scripts]
    csv-linter=csv_linter:main
    """,
    install_requires = ['click==7.1.2', 'pandas==1.2.0'],
    version = '0.0.1',
    url = 'https://github.com/paiml/practical-mlops-book',
)
```

Este ficheiro permite aos instaladores Python capturar todos os detalhes do pacote Python como dependências e, neste caso, a disponibilidade de uma nova ferramenta de linha de comandos chamada *csv-linter*. A maioria dos campos na chamada `setup` são diretos, mas vale a pena notar os detalhes do valor `entry_points`. Esta é uma característica da biblioteca `setuptools`, que permite definir uma função dentro de um ficheiro Python para mapear de volta para um nome de ferramenta de linha de comandos. Neste caso, estou a dar o nome de *csv-linter* à ferramenta de linha de comandos, e estou a mapeá-la para uma função (`main`) que vou criar a seguir dentro de um ficheiro chamado `csv_linter.py`. Embora eu tenha escolhido *csv-linter* para ser o nome da ferramenta, ela pode ser chamada de qualquer coisa. Nos bastidores, a biblioteca `setuptools` criará o executável com o que quer que seja declarado aqui. Não há qualquer restrição para lhe dares o mesmo nome que o ficheiro Python.

Abre um novo ficheiro chamado `csv_linter.py` e adiciona uma única função que utiliza a estrutura Click:

```
import click

@click.command()
def main():
    return
```

NOTA

Mesmo quando os exemplos não mencionam explicitamente o uso dos ambientes virtuais do Python, é sempre uma boa ideia criares um. Ter ambientes virtuais é uma forma robusta de isolar dependências e potenciais problemas com outras bibliotecas instaladas num sistema.

Estes dois ficheiros são quase tudo o que precisas para criar uma ferramenta de linha de comandos que (por agora) não faz nada para além de fornecer um executável disponível no teu caminho da shell. A seguir, cria um ambiente virtual e ativa-o para instalar a ferramenta recém-criada:

```
$ python3 -m venv venv
$ source venv/bin/activate
$ python setup.py develop
running develop
running egg_info
...
csv-linter 0.0.1 is already the active version in easy-
install.pth
...
Using /Users/alfredo/.virtualenvs/practical-
mlops/lib/python3.8/site-packages
Finished processing dependencies for csv-linter==0.0.1
```

O script *setup.py* tem muitas formas diferentes de ser invocado, mas irás usar principalmente o argumento **install** ou o **develop** que usei no exemplo. Usar **develop** permite-te fazer alterações ao código fonte do script e tê-las automaticamente disponíveis no script, enquanto que **install** criaria um script separado (ou autónomo) sem ligações ao código fonte. Ao desenvolver ferramentas de linha de comandos, recomendo que uses **develop** para testar as alterações à medida que progrides rapidamente. Depois de chamar o script *setup.py*, testa a nova ferramenta disponível passando **--help** sinalizador:

```
$ csv-linter --help
Usage: csv-linter [OPTIONS]

Options:
  --help  Show this message and exit.
```

Obter um menu de ajuda sem ter que escrever um é ótimo, e é um recurso que alguns outros frameworks de ferramentas de linha de comando oferecem. Agora que a ferramenta está disponível como um script no terminal, é hora de adicionar recursos úteis. Para manter as coisas simples, este script aceitará um arquivo CSV como um único argumento. O framework Click tem um auxiliar embutido para aceitar arquivos como argumentos que garantem que o arquivo existe e produz um erro útil caso contrário. Actualiza o ficheiro *csv_linter.py* para usar o auxiliar:

```
import click

@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main():
    return
```

Embora o script ainda não esteja a fazer nada com o ficheiro, o menu de ajuda foi atualizado para refletir a opção:

```
$ csv-linter --help
Usage: csv-linter [OPTIONS] FILENAME
```

Mesmo assim, não está a fazer nada de útil. Vê o que acontece se passares um ficheiro CSV que não existe:

```
$ csv-linter bogus-dataset.csv
Usage: csv-linter [OPTIONS] FILENAME
Try 'csv-linter --help' for help.

Error: Invalid value for 'FILENAME': Path 'bogus-dataset.csv'
does not exist.
```

Leva a ferramenta um passo à frente, utilizando o argumento `filename` que é passado para a função `main()` com o Pandas para descrever o conjunto de dados:

```
import click
import pandas as pd

@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main(filename):
    df = pd.read_csv(filename)
    click.echo(df.describe())
```

O script usa o Pandas e outro ajudante do Click chamado `echo`, que nos permite imprimir facilmente a saída para o terminal. Usa como entrada o

ficheiro `wine.csv` previamente guardado quando processaste o conjunto de dados:

```
$ csv-linter wine.csv
      Unnamed: 0  grape      rating
count  32780.000000    0.0  32780.000000
mean   16389.500000    NaN  91.186608
std    9462.915248    NaN  2.190391
min    0.000000    NaN  85.000000
25%   8194.750000    NaN  90.000000
50%   16389.500000    NaN  91.000000
75%   24584.250000    NaN  92.000000
max   32779.000000    NaN  99.000000
```

Ainda assim, isto não é *muito útil*, apesar de agora ser fácil descrever qualquer ficheiro CSV usando o Pandas. O problema que precisamos de resolver aqui é alertar-nos para três potenciais problemas:

- Detecta colunas de contagem zero
- Avisa quando existem colunas **Unnamed**
- Verifica se há retornos de carroagem num campo

Vamos começar por detetar colunas com contagem zero. O Pandas permite-nos iterar sobre as suas colunas e tem um método `count()` que podemos utilizar para este fim:

```
In [10]: for key in df.keys():
...:     print(df[key].count())
...:
32780
0
32777
32422
32780
32780
```

Adapta o ciclo a uma função separada de `main()` no ficheiro `csv_linter.py` para que fique isolado e mantenha as coisas legíveis:

```

def zero_count_columns(df):
    bad_columns = []
    for key in df.keys():
        if df[key].count() == 0:
            bad_columns.append(key)
    return bad_columns

```

A função `zero_count_columns()` recebe como entrada o quadro de dados do Pandas, captura todas as colunas com uma contagem zero e devolve-as no final. Está isolada e ainda não coordena a saída com a função `main()`. Como ela retorna uma lista de nomes de colunas, faz um loop sobre o conteúdo do resultado na função `main()`:

```

@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main(filename):
    df = pd.read_csv(filename)
    # check for zero count columns
    for column in zero_count_columns(df):
        click.echo(f"Warning: Column '{column}' has no items in it")

```

Executa o script no mesmo ficheiro CSV (nota que removi a chamada `.describe()`):

```
$ csv-linter wine-ratings.csv
Warning: Column 'grape' has no items in it
```

Com 19 linhas, este script já me teria poupado muito tempo se o tivesse utilizado antes de enviar os dados para uma plataforma de ML. Em seguida, criei outra função que percorre as colunas para verificar se existem Unnamed:

```

def unnamed_columns(df):
    bad_columns = []
    for key in df.keys():
        if "Unnamed" in key:
            bad_columns.append(key)
    return len(bad_columns)

```

Neste caso, a função verifica se a cadeia "Unnamed" está presente no nome, mas não devolve os nomes (uma vez que assumimos que são todos semelhantes ou mesmo iguais), mas devolve a contagem total. Com essa informação, expande a função `main()` para incluir a contagem:

```
@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main(filename):
    df = pd.read_csv(filename)
    # check for zero count columns
    for column in zero_count_columns(df):
        click.echo(f"Warning: Column '{column}' has no items in it")
    unnamed = unnamed_columns(df)
    if unnamed:
        click.echo(f"Warning: found {unnamed} columns that are Unnamed")
```

Executa a ferramenta mais uma vez com o mesmo ficheiro CSV para verificar os resultados:

```
$ csv-linter wine.csv
Warning: Column 'grape' has no items in it
Warning: found 1 column that is Unnamed
```

Finalmente, e talvez o mais difícil de detetar, é encontrar retornos de carruagem num campo de texto grande. Esta operação pode ser dispendiosa, dependendo do tamanho do conjunto de dados. Embora existam maneiras mais eficientes de realizar a iteração, o próximo exemplo tentará usar a abordagem mais simples. Cria outra função que faça o trabalho sobre o quadro de dados do Pandas:

```
def carriage_returns(df):
    for index, row in df.iterrows():
        for column, field in row.iteritems():
            try:
                if "\r\n" in field:
                    return index, column, field
            except TypeError:
                continue
```

O ciclo impede que seja criado um `TypeError`. Se a função fizer uma verificação da cadeia de caracteres em relação a um tipo diferente, como um número inteiro, então será produzido um `TypeError`. Uma vez que a operação pode ser dispendiosa, a função sai do ciclo ao primeiro sinal de um carriage return. Por fim, o loop devolve o índice, a coluna e o campo inteiro para serem reportados pela função `main()`. Agora atualiza o script para incluir o relatório de retornos de carro:

```
@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main(filename):
    df = pd.read_csv(filename)
    for column in zero_count_columns(df):
        click.echo(f"Warning: Column '{column}' has no items in it")
    unnamed = unnamed_columns(df)
    if unnamed:
        click.echo(f"Warning: found {unnamed} columns that are Unnamed")

    carriage_field = carriage_returns(df)
    if carriage_field:
        index, column, field = carriage_field
        click.echo((
            f"Warning: found carriage returns at index {index}"
            f" of column '{column}':")
        )
        click.echo(f"          '{field[:50]}'"")
```

Testar esta última verificação é complicado porque o conjunto de dados já não tem retornos de carruagem. [O repositório para este capítulo](#) inclui um exemplo de CSV com retornos de carriage. Faz o download desse ficheiro localmente e aponta a ferramenta `csv-linter` para esse ficheiro:

```
$ csv-linter carriage.csv
Warning: found carriage returns at index 0 of column 'notes':
'Aged in French, Hungarian, and American Oak barrel'
```

Para evitar que um campo extremamente longo seja impresso na saída, a mensagem de aviso mostra apenas os primeiros 50 caracteres. Esta

ferramenta de linha de comandos utiliza a estrutura Click para a funcionalidade da ferramenta de linha de comandos e o Pandas para a inspeção do CSV. Embora faça apenas três verificações e não tenha muito desempenho, teria sido inestimável para eu evitar problemas ao usar o conjunto de dados. Existem várias outras formas de garantir que um conjunto de dados está em condições aceitáveis, mas este é um excelente exemplo de como automatizar (e evitar) os problemas que encontras. A automação é a base do DevOps, e as ferramentas de linha de comando são uma excelente maneira de iniciar o caminho da automação.

Modularizando uma ferramenta de linha de comando

A ferramenta de linha de comandos anterior mostrou como usar as bibliotecas internas do Python para criar um script a partir de um único ficheiro Python. Mas é inteiramente possível usar um diretório com múltiplos ficheiros que compõem uma única ferramenta de linha de comandos. Esta abordagem é preferível quando o conteúdo de um único script começa a ser difícil de ler. Não existe um limite rígido para o que te deve fazer dividir um ficheiro longo em vários; eu recomendo agrupar código que partilha responsabilidades comuns e separá-los, especialmente quando existe uma necessidade de reutilização de código. Pode não haver um caso de uso para reutilização de código em algumas situações, mas dividir algumas partes ainda pode fazer sentido para melhorar a legibilidade e a manutenção.

Vamos reutilizar o exemplo da ferramenta *csv-linter* para adaptar o script de ficheiro único a vários ficheiros numa diretoria. O primeiro passo é criar uma diretoria com um ficheiro `__init__.py` e mover o ficheiro `csv_linter.py` para lá. Usando um ficheiro `__init__.py` diz ao Python para tratar esse diretório como um módulo. A estrutura deve agora parecer-se com isto:

```
$ tree .
.
├── csv_linter
│   ├── __init__.py
│   └── csv_linter.py
└── requirements.txt
```

```
└── setup.py  
1 directory, 4 files
```

Nesta altura, não há necessidade de repetir o nome da ferramenta no ficheiro Python, por isso muda o nome para algo mais modular e menos ligado ao nome da ferramenta. Normalmente sugiro usar *main.py*, por isso muda o nome do ficheiro:

```
$ mv csv_linter.py main.py  
$ ls  
__init__.py main.py
```

Tenta usar o comando `csv_linter` mais uma vez. A ferramenta deve estar num estado quebrado porque os ficheiros foram movidos:

```
$ csv-linter  
Traceback (most recent call last):  
  File ".../site-packages/pkg_resources/__init__.py", line 2451,  
in resolve  
    return functools.reduce(getattr, self.attrs, module)  
AttributeError: module 'csv_linter' has no attribute 'main'
```

Isto acontece porque o ficheiro *setup.py* está a apontar para um módulo que já não existe. Actualiza esse ficheiro para que encontre a função `main()` dentro do ficheiro *main.py*:

```
from setuptools import setup, find_packages  
  
setup(  
    name = 'csv-linter',  
    description = 'lint csv files',  
    packages = find_packages(),  
    author = 'Alfredo Deza',  
    entry_points=""",  
        [console_scripts]  
        csv-linter=csv_linter.main:main  
    """,  
    install_requires = ['click==7.1.2', 'pandas==1.2.0'],  
    version = '0.0.1',  
    url = 'https://github.com/paiml/practical-mlops-book',  
)
```

A mudança pode ser difícil de detetar, mas o ponto de entrada para `CSV-linter` é agora `csv_linter.main:main`. Essa mudança significa que o `setuptools` deve procurar um pacote `csv_linter` com um módulo `principal` com uma função `main()` nele. A sintaxe é um pouco complicada de lembrar (eu sempre tenho que procurar), mas entender os detalhes da mudança ajuda a visualizar como as coisas estão ligadas. O processo de instalação ainda tem todas as referências antigas, por isso tens de correr o `setup.py` novamente para que tudo funcione:

```
$ python setup.py develop
running develop
Installing csv-linter script to
/Users/alfredo/.virtualenvs/practical-mlops/bin
...
Finished processing dependencies for csv-linter==0.0.1
```

Agora que a ferramenta `csv-linter` está a funcionar novamente, vamos dividir o módulo `main.py` em dois ficheiros, um para as verificações e outro apenas para o trabalho da ferramenta de linha de comandos. Cria um novo ficheiro chamado `checks.py` e move as funções que fazem as verificações de `main.py` para este novo ficheiro:

```
# in checks.py

def carriage_returns(df):
    for index, row in df.iterrows():
        for column, field in row.iteritems():
            try:
                if "\r\n" in field:
                    return index, column, field
            except TypeError:
                continue

def unnamed_columns(df):
    bad_columns = []
    for key in df.keys():
        if "Unnamed" in key:
            bad_columns.append(key)
    return len(bad_columns)
```

```

def zero_count_columns(df):
    bad_columns = []
    for key in df.keys():
        if df[key].count() == 0:
            bad_columns.append(key)
    return bad_columns

```

E agora actualiza o *main.py* para importar as funções de verificação do ficheiro *checks.py*. O módulo principal recém-atualizado deve agora ter o seguinte aspeto:

```

import click
import pandas as pd
from csv_linter.checks import
    carriage_returns,
    unnamed_columns,
    zero_count_columns

@click.command()
@click.argument('filename', type=click.Path(exists=True))
def main(filename):
    df = pd.read_csv(filename)
    for column in zero_count_columns(df):
        click.echo(f"Warning: Column '{column}' has no items in
it")
    unnamed = unnamed_columns(df)
    if unnamed:
        click.echo(f"Warning: found {unnamed} columns that are
Unnamed")
    carriage_field = carriage_returns(df)
    if carriage_field:
        index, column, field = carriage_field
        click.echo((
            f"Warning: found carriage returns at index {index}"
            f" of column '{column}':")
        )
        click.echo(f'{field[:50]}')

```

A modularização é uma ótima maneira de manter as coisas curtas e legíveis. Quando uma ferramenta separa as preocupações desta forma, é mais fácil de manter e de raciocinar. Houve muitas vezes em que tive que trabalhar com scripts legados que tinham *milhares* de linhas sem nenhuma boa razão.

Agora que o script está em boa forma, podemos entrar em microsserviços e levar alguns desses conceitos adiante.

Microsserviços

Como mencionei no início deste capítulo, os microsserviços são um novo tipo de paradigma de aplicativo, totalmente oposto aos aplicativos monólitos do estilo antigo. Para operações de ML, em particular, é fundamental isolar as responsabilidades, tanto quanto possível, do processo de colocar modelos em produção. O isolamento de componentes pode então abrir caminho para a reutilização em outros lugares, não apenas vinculado a um processo específico para um único modelo.

Costumo pensar nos microsserviços e nos componentes reutilizáveis como peças de um puzzle Jenga. Uma aplicação monólita seria uma torre de Jenga extremamente alta com muitas peças a trabalhar em conjunto para a manter de pé, mas com uma grande falha: não tentes tocar em nada que possa deitar tudo abaixo. Por outro lado, se as peças forem colocadas juntas da forma mais robusta possível (como no início do jogo de puzzle), então remover as peças e colocá-las num local diferente é simples.

É comum que os engenheiros de software criem rapidamente utilitários que estão intimamente ligados à tarefa em questão. Por exemplo, uma lógica que remove alguns valores de uma cadeia de caracteres, que pode depois ser utilizada para a manter numa base de dados. Depois de as poucas linhas de código terem provado o seu valor, é útil pensar na reutilização de outros componentes da base de código. Eu costumo ter um módulo de utilitários nos meus projectos, onde os utilitários comuns vão para que outras partes da aplicação que precisam das mesmas facilidades possam importá-los e reutilizá-los.

Tal como a contentorização, os microsserviços permitem concentrar-se mais na solução em si (o código) do que no ambiente (por exemplo, o sistema operativo). Uma excelente solução para a criação de microsserviços é a utilização de tecnologias sem servidor. Os produtos sem servidores fornecedores decloud têm muitos nomes diferentes (funções lambda e

cloud, por exemplo). Ainda assim, todos eles se referem à mesma coisa: cria um único ficheiro com algum código e implementa instantaneamente na Cloud - sem necessidade de te preocupares com o sistema operativo subjacente ou as suas dependências. Basta selecionar um tempo de execução a partir de um menu pendente, como o Python 3.8, e clicar num botão. A maioria dos fornecedores de Cloud, de facto, permite-te criar a função diretamente no browser. Este tipo de desenvolvimento e aprovisionamento é bastante revolucionário e tem vindo a permitir padrões de aplicação interessantes que antes eram muito complicados de alcançar.

Outro aspecto crítico do serverless é que podes aceder à maioria das ofertas do fornecedor de Cloud sem grande esforço. Para o ML, isto é crucial: precisas de efetuar algumas operações de visão computacional? Uma implantação sem servidor pode fazer isso em menos de uma dúzia de linhas de código. Esta forma de alavancar as operações de ML na Cloud dá-te velocidade, robustez e reproduzibilidade: todos os componentes significativos dos princípios DevOps. A maioria das empresas não precisará de criar o seu próprio modelo de visão computacional a partir do zero. A frase "apoiar-se nos ombros de gigantes" é perfeita para entender as possibilidades. Há alguns anos, trabalhei numa agência de meios digitais com uma equipa de uma dúzia de funcionários de TI que tinham decidido gerir os seus servidores de correio eletrónico internamente. Gerir os servidores de correio eletrónico (corretamente) exige uma enorme quantidade de conhecimentos e um esforço contínuo. O correio eletrónico é um *problema difícil de resolver*. Posso dizer-te que o e-mail deixava de funcionar continuamente - na verdade, era quase uma ocorrência mensal.

Por fim, vamos ver quantas opções existem para criar microsserviços baseados em ML em provedores de Cloud. Normalmente, elas variam de mais IaaS (infraestrutura como serviço) a mais PaaS (plataformas como serviço). Por exemplo, na [Figura 11-1](#), o Kubernetes é uma tecnologia complexa e de nível inferior que implanta microsserviços. Em outros cenários, como o AWS App Runner, abordado anteriormente no livro, podes apontar o teu repositório do GitHub para o serviço e clicar em alguns

botões para obter uma plataforma de entrega contínua totalmente implantada. Em algum lugar no meio estão as funções de Cloud.

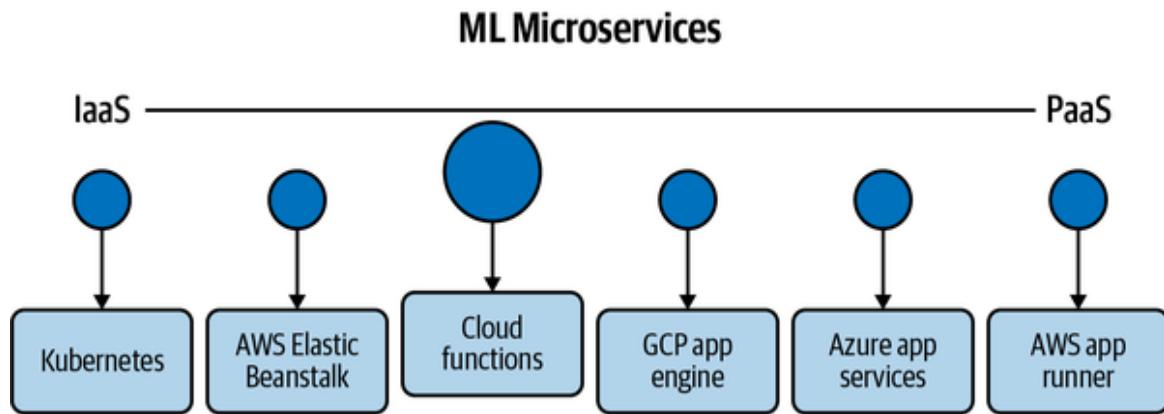


Figura 11-1. Microsserviços Cloud ML

Qual é a principal competência da tua empresa? Se não forem modelos de visão por computador de última geração, então não os crie tu mesmo. Da mesma forma, trabalha de forma inteligente, não com afinco, e constrói sobre sistemas de alto nível como o AWS App Runner ou o Google Cloud Run. Finalmente, resiste à vontade de reinventar a roda e aproveita os microsserviços Cloud.

Criar uma função sem servidor

A maioria dos fornecedores de Cloud expõe os seus serviços de ML nos seus ambientes sem servidor. Visão computacional, processamento de linguagem natural e serviços de recomendação são apenas alguns deles. Nesta secção, utilizarás uma API de tradução para tirar partido de uma das ofertas de processamento de linguagem mais potentes do mundo.

NOTA

Para este aplicativo sem servidor, usarei o [Google Cloud Platform](#) (GCP). Se ainda não te inscreveste, podes obter alguns créditos gratuitos para experimentar o exemplo desta secção, embora, com os limites actuais, devas conseguir implementar a função Cloud sem incorrer em quaisquer custos.

Depois de iniciar sessão no GCP, seleciona Cloud Functions na barra lateral esquerda, na secção Compute, conforme mostrado na [Figura 11-2](#).

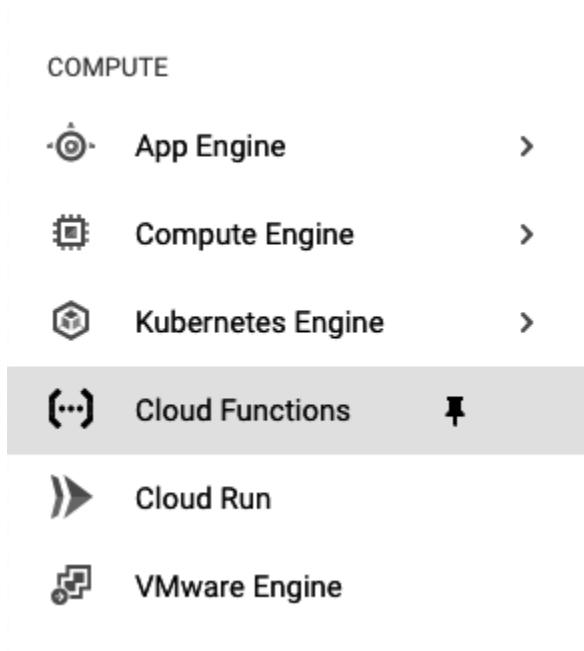


Figura 11-2. Barra lateral das funções Cloud

Se ainda não tiveres criado uma função, uma mensagem de saudação deverá mostrar-te uma ligação para criares uma. Se já tiveres implementado uma função, deverá estar disponível um botão Criar função. Criar e implantar uma função a partir da interface do usuário envolve apenas algumas etapas. [A Figura 11-3](#) é o formulário que deves esperar preencher.

1 Configuration — 2 Code

Basics

Function name * ?

Region ▼ ?

Trigger

HTTP

Trigger type ▼

URL

Authentication

- Allow unauthenticated invocations
Check this if you are creating a public API or website.
- Require authentication
Manage authorized users with Cloud IAM.

SAVE

CANCEL

VARIABLES, NETWORKING AND ADVANCED SETTINGS



Figura 11-3. Cria uma função Cloud

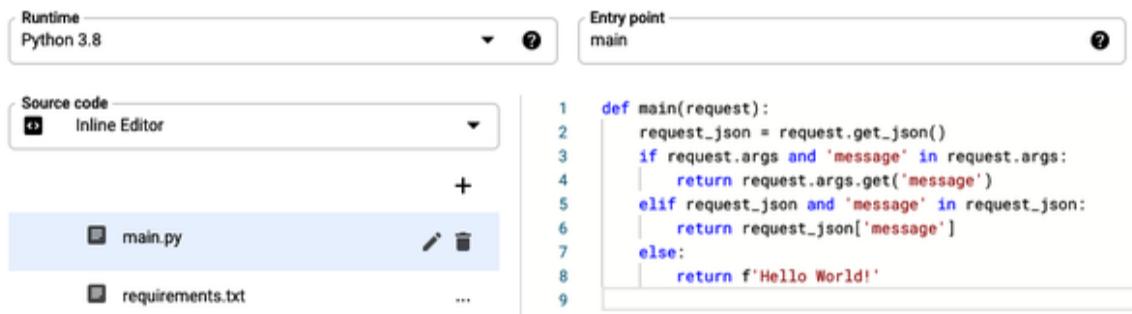
Os valores predefinidos para a secção Básicos são suficientes. Neste caso, o formulário vem pré-preenchido com *function-1* como o nome e usando *us-central1* como a região. Certifica-te de que defines o tipo de acionamento

como HTTP e que a autenticação é necessária. Clica em Guardar e depois no botão Seguinte na parte inferior da página.

AVISO

Embora as invocações não autenticadas a funções sejam permitidas (e tão simples como selecionar a opção no formulário Web), sugiro vivamente que nunca implementes uma função Cloud sem a autenticação activada. Os serviços expostos sobre HTTP que não são autenticados representam um risco de abuso. Uma vez que a utilização de uma função Cloud está diretamente ligada à tua conta e ao teu orçamento, pode ter um impacto financeiro substancial devido a uma utilização não autorizada.

Uma vez na secção Código, podes selecionar um tempo de execução e um ponto de entrada. Selecciona Python 3.8, altera o ponto de entrada para usar *main*, e actualiza o nome da função para usar *main()* em vez de *hello_world()* como mostrado na [Figura 11-4](#).



The screenshot shows the AWS Lambda function configuration interface. On the left, there are dropdown menus for 'Runtime' set to 'Python 3.8' and 'Entry point' set to 'main'. Below these are sections for 'Source code' (with an 'Inline Editor' button) and file lists for 'main.py' and 'requirements.txt'. The main area displays the Python code:

```
1 def main(request):
2     request_json = request.get_json()
3     if request.args and 'message' in request.args:
4         return request.args.get('message')
5     elif request_json and 'message' in request_json:
6         return request_json['message']
7     else:
8         return f'Hello World!'
9
```

Figura 11-4. Código das funções Cloud

A capacidade de escolher o ponto de entrada para a aplicação abre a possibilidade de criar outras funções para ajudar a função principal ou determinar outra convenção de nomes para interagir com o código. A flexibilidade é ótima, mas ter padrões e usar convenções é mais valioso. Depois de fazeres as alterações necessárias, clica no botão Implementar para colocar esta função num ambiente de produção. Depois de concluída, a função deve ser exibida no painel Funções da Cloud.

Em seguida, após a implantação, vamos interagir com ele enviando uma solicitação HTTP. Há muitas maneiras de fazer isso. Para começar, clica em Ações para a função selecionada e escolhe "Testar função". Uma nova

página é carregada e, embora possa ser difícil de ver no início, a secção "Triggering event" é onde adicionas o corpo do pedido a ser enviado. Como a função está à procura de uma chave "message", actualiza o corpo para incluir uma mensagem como mostra a Figura 11-5 e, em seguida, clica no botão Testar a função.

The screenshot shows a configuration interface for a Cloud Function. At the top, it says "Triggering event" with a question mark icon. Below that is a code editor with two lines of JSON:

```
1 {"message": "Practical MLOps in a function!"}
2
```

At the bottom is a blue button labeled "(...) TEST THE FUNCTION".

Figura 11-5. Código da função Cloud - Evento de acionamento

Deve levar apenas alguns segundos para obter a saída, que deve ser a saída do valor da chave "message". Além dessa saída, alguns logs devem aparecer, tornando esta uma forma muito simples de interagir com a função. Uma coisa que não foi necessária foi a realização de quaisquer passos de autenticação, embora a função tenha sido criada com a autenticação activada. Sempre que estiveres a depurar e quiseres testar rapidamente uma função implementada, esta é, de longe, a forma mais fácil.

Esta função aceita JSON (JavaScript Object Notation) como entrada. Embora não esteja claro que a função Cloud usa HTTP ao testar, é assim que a entrada é entregue à função. O JSON às vezes é chamado de *língua franca* (linguagem comum) do desenvolvimento da Web porque as linguagens de programação e outros serviços e implementações podem consumir e produzir JSON para construções nativas que essas linguagens e serviços entendem.

Embora as APIs HTTP possam restringir os tipos de pedidos e o formato do corpo, é relativamente comum usar JSON para comunicar. Em Python, podes carregar JSON em estruturas de dados nativas como listas e dicionários, que são fáceis de usar.

Antes de explorares outras formas de interagir com a função (incluindo a autenticação), vamos aproveitar os serviços de ML da Google utilizando o seu serviço de tradução. Por predefinição, todas as APIs da Cloud Platform da Google estão desactivadas. Se precisares de interagir com uma oferta Cloud, como traduções de idiomas, tens de ativar a API antes de a utilizares. Não é um grande problema se criares uma função Cloud (como neste caso) e te esqueceres de o fazer. O comportamento resultante seria um erro capturado nos logs e um HTTP 500 retornado como uma resposta de erro para o cliente que fez a solicitação. Este é um excerto dos registos de uma função que tentou utilizar a API de tradução sem a ativar primeiro:

```
google.api_core.exceptions.PermissionDenied: 403 Cloud  
Translation API has not  
been used in project 555212177956 before or it is disabled.  
Enable it by visiting:
```

```
https://console.developers.google.com/apis/api/translate.googleapis.com/  
then retry. If you enabled this API recently, wait a few minutes  
for the  
action to propagate to our systems and retry."
```

Habilita a **API Cloud Translation** antes de fazer qualquer modificação adicional na função. A maioria das APIs fornecidas pelo GCP precisa de ser activada de forma semelhante, acedendo à **ligação APIs e Serviços** e encontrando a API de que precisas na página Biblioteca.

NOTA

Se não fores um administrador na conta GCP e não vires uma API disponível, podes não ter as permissões necessárias para ativar uma API. Um administrador da conta precisa conceder-te as permissões adequadas.

Depois de ativar a API, volta à função clicando no seu nome para que o painel de controlo seja carregado. Uma vez no painel, encontra o botão Editar na parte superior da página para fazer alterações no código-fonte. A secção Editar apresenta-te primeiros opções para configurar a própria função

e depois o código. Não há necessidade de fazer alterações na configuração de implantação, então clica em Avançar para finalmente chegar à fonte. Clica no link *requirements.txt* que abre esse ficheiro para adicionar a biblioteca API que é necessária para interagir com o serviço de tradução:

```
google-cloud-translate==3.0.2
```

Agora clica em *main.py* para editar o conteúdo. Adiciona a instrução de importação para trazer o serviço translate e adiciona uma nova função que será responsável por fazer a tradução:

```
from google.cloud import translate

def translator(text="YOUR_TEXT_TO_TRANSLATE",
    project_id="YOUR_PROJECT_ID", language="fr"):

    client = translate.TranslationServiceClient()
    parent = f"projects/{project_id}/locations/global"

    response = client.translate_text(
        request={
            "parent": parent,
            "contents": [text],
            "mime_type": "text/plain",
            "source_language_code": "en-US",
            "target_language_code": language,
        }
    )
    # Display the translation for each input text provided
    for translation in response.translations:
        print(u"Translated text:\n{}".format(translation.translated_text))
    return u"Translated text:\n{}".format(response.translations[0].translated_text)
```

Esta nova função recebe três argumentos para interagir com a API de tradução: o texto de entrada, o ID do projeto e a língua de destino para a tradução (por defeito, francês). O texto de entrada tem como padrão o inglês, mas a função pode ser adaptada para usar outros idiomas (por exemplo, espanhol) como entrada e inglês como saída. Desde que a língua

seja suportada, a função pode usar a entrada e a saída com qualquer combinação.

A resposta ao pedido de tradução é iterável, pelo que é necessário um ciclo após a tradução estar concluída.

Agora modifica a função `main()` para passar o valor de "message" para a função `translator()`. Estou a utilizar o meu próprio ID de projeto ("gcp-book-1"), por isso certifica-te de que o actualizas com o teu se tentares o próximo exemplo:

```
def main(request):
    request_json = request.get_json()
    if request_json and 'message' in request_json:
        return translator(
            text=request_json['message'],
            project_id="gcp-book-1"
        )
    else:
        return f'No message was provided to translate'
```

A função `main()` ainda requer um valor "message" atribuído no pedido JSON de entrada, mas agora faz algo útil com ele. Testa-a na consola com um exemplo de entrada JSON:

```
{"message": "a message that has been translated!"}
```

A saída na página de teste (mostrada na [Figura 11-6](#)) deve ser quase imediata.

The screenshot shows the Google Cloud Platform Functions test interface. At the top, there's a status bar with 'Output' and a green checkmark labeled 'Complete'. Below it is a text input field containing the JSON payload: {"message": "a message that has been translated!"}. The main area displays the output: '\$ Translated text: un message qui a été traduit!'. Below this, there's a 'Logs' section with a green checkmark and the text 'Fetched (up to 100 entries). [View all logs](#)'. It shows a single log entry: 'Loading... Scanned up to 12/9/20, 2:59 AM. Scanned 4 GB.' At the bottom, there's another log entry: '▶ 2021-01-16 07:04:04.802 EST function-2 o3k33zyu1wjr Function execution started'. A horizontal line separates this from the rest of the logs.

Figura 11-6. Teste traduzido

Autenticação para funções Cloud

Vejo o acesso HTTP como uma democracia de acesso: uma enorme flexibilidade para que outros sistemas e linguagens interajam a partir das suas implementações num serviço separado num local remoto, utilizando as especificações HTTP. Todas as principais linguagens de programação podem construir pedidos HTTP e processar respostas de servidores. Reunir serviços com a ajuda do HTTP permite que estes serviços funcionem de novas formas, potencialmente de formas que não foram pensadas inicialmente. Pensa nas API HTTP como uma funcionalidade extensível que pode ser ligada a qualquer coisa ligada à Internet. Mas a ligação através da Internet tem implicações de segurança, como impedir o acesso não autorizado com pedidos autenticados.

Existem algumas formas de interagires remotamente com uma função Cloud. Vou começar com a linha de comando, usando o programa *curl*. Embora eu tenda a não usar o *curl* para interagir com solicitações autenticadas, ele oferece uma maneira direta de documentar todas as peças necessárias para que a solicitação seja enviada com êxito. Instala o [Google Cloud SDK](#) para o teu sistema e, em seguida, determina o ID do projeto e o nome da função que implementaste anteriormente. O exemplo a seguir usa *curl* com o SDK para autenticação:

```
$ curl -X POST --data '{"message": "from the terminal!"}' \
  -H "Content-Type: application/json" \
  -H "Authorization: bearer $(gcloud auth print-identity-token)" \
  https://us-central1-gcp-book-1.cloudfunctions.net/function-1
```

No meu sistema, utilizando o URL *da função-1*, obtenho a seguinte resposta:

Translated text: du terminal!

O comando parece muito complicado, mas apresenta uma imagem melhor do que é necessário para fazer uma solicitação bem-sucedida. Primeiro, declara que a solicitação usa um método POST. Esse método é

normalmente usado quando uma carga útil é associada a uma solicitação. Nesse caso, `curl` está enviando JSON do argumento para o sinalizador `--data`. Em seguida, o comando adiciona dois cabeçalhos de solicitação, um para indicar o tipo de conteúdo que está sendo enviado (JSON) e o outro para indicar que a solicitação está fornecendo um token. O token é onde o SDK entra em ação porque cria um token para a solicitação, que é exigido pelo serviço da função Cloud para verificar se a solicitação é autenticada. Por fim, o URL da função de nuvem é usado como destino para essa solicitação POST autenticada que envia JSON.

Tenta executar o comando SDK por si só para ver o que faz:

```
$ gcloud auth print-identity-token  
aIWQ06IClq5fNy1HWPHJRtoMu4IG0QmP84tnzY5Ats_4XQvC1ne-  
A9coqEciMu_WI4Tjnias3fJJali  
[...]
```

Agora que comprehendes os componentes necessários para o pedido, experimenta o SDK diretamente para fazer o pedido para chegar à função Cloud implementada:

```
$ gcloud --project=gcp-book-1 functions call function-1 \  
--data '{"message":"I like coffee shops in Paris"}'  
executionId: 1jgd75feo290  
result: "Translated text: J'aime les cafés à Paris"
```

Penso que é uma excelente ideia dos fornecedores de Cloud, como a Google, incluir outras facilidades para interagir com os seus serviços, como acontece aqui com a função Cloud. Se apenas tivesses conhecimento do comando SDK para interagir com uma função Cloud, seria difícil utilizar uma linguagem de programação para construir um pedido, por exemplo, com Python. Estas opções oferecem flexibilidade, e quanto mais flexível for um ambiente, melhores serão as hipóteses de o adaptar da forma mais razoável às necessidades do teu ambiente.

Agora vamos usar o Python para interagir com a função de tradução.

NOTA

O exemplo a seguir chamará diretamente o comando gcloud usando Python, facilitando a demonstração rápida de como criar código Python para interagir com a função Cloud. No entanto, não é uma forma robusta de lidar com a autenticação. Terás de [criar uma conta de serviço](#) e utilizar o *cliente google-api-python* para proteger corretamente o processo de autenticação.

Cria um ficheiro Python chamado *trigger.py* e adiciona o seguinte código para recuperar o token do comando gcloud:

```
import subprocess

def token():
    proc = subprocess.Popen(
        ["gcloud", "auth", "print-identity-token"],
        stdout=subprocess.PIPE)
    out, err = proc.communicate()
    return out.decode('utf-8').strip('\n')
```

A função `token()` irá chamar o comando `gcloud` e processa a saída para fazer o pedido. Vale a pena reiterar que esta é uma forma rápida de demonstrar como fazer pedidos para acionar a função a partir do Python. Deves considerar a criação de uma conta de serviço e OAuth2 a partir do *cliente google-api-python* se pretenderes implementar isto num ambiente de produção.

Agora cria o pedido utilizando esse token para comunicar com a função Cloud:

```
import subprocess
import requests

url = 'https://us-central1-gcp-book-1.cloudfunctions.net/function-1'

def token():
    proc = subprocess.Popen(
        ["gcloud", "auth", "print-identity-token"],
```

```

        stdout=subprocess.PIPE)
out, err = proc.communicate()
return out.decode('utf-8').strip('\n')

resp = requests.post(
    url,
    json={"message": "hello from a programming language"},
    headers={"Authorization": f"Bearer {token()}"}
)
print(resp.text)

```

Nota que adicionei a biblioteca *requests* (versão 2.25.1 no meu caso) ao script, por isso tens de a instalar antes de continuar. Agora executa o ficheiro *trigger.py* para o testar, assegurando que actualizaste o script com o teu ID de projeto:

```
$ python trigger.py
Translated text: bonjour d'un langage de programmation
```

Criando uma CLI baseada em Cloud

Agora que entendas os conceitos para construir uma ferramenta de linha de comando, empacotá-la e distribuí-la enquanto aproveita a Cloud para as suas ofertas de ML, é interessante ver tudo isto junto. Nesta secção, vou reutilizar todas as diferentes partes para criar uma. Cria um novo diretório e adiciona o seguinte a um ficheiro *setup.py* para que o empacotamento seja resolvido de imediato:

```

from setuptools import setup, find_packages

setup(
    name = 'cloud-translate',
    description = "translate text with Google's cloud",
    packages = find_packages(),
    author = 'Alfredo Deza',
    entry_points="""
    [console_scripts]
    cloud-translate=trigger:main
    """,
    install_requires = ['click==7.1.2', 'requests==2.25.1'],

```

```

        version = '0.0.1',
        url = 'https://github.com/paiml/practical-mlops-book',
)

```

O arquivo *setup.py* criará um executável *cloud-translate* mapeado para uma função *main()* dentro do arquivo *trigger.py*. Ainda não criámos essa função, por isso adiciona o ficheiro *trigger.py* que foi criado na secção anterior e adiciona a função:

```

import subprocess
import requests
import click

url = 'https://us-central1-gcp-book-
1.cloudfunctions.net/function-2'

def token():
    proc = subprocess.Popen(
        ["gcloud", "auth", "print-identity-token"],
        stdout=subprocess.PIPE)
    out, err = proc.communicate()
    return out.decode('utf-8').strip('\n')

@click.command()
@click.argument('text', type=click.STRING)
def main(text):
    resp = requests.post(
        url,
        json={"message": text},
        headers={"Authorization": f"Bearer {token()}"})

    click.echo(f"{resp.text}")

```

O ficheiro não é muito diferente do *trigger.py* inicial, onde corre diretamente com o Python. A estrutura Click permite-nos definir uma entrada *de texto* e depois imprime a saída para o terminal quando estiver concluída. Corre `python setup.py develop` para que tudo fique ligado, incluindo as dependências. Como esperado, a estrutura dá-nos o menu de ajuda:

```
$ cloud-translate --help
Usage: cloud-translate [OPTIONS] TEXT

Options:
  --help  Show this message and exit.
$ cloud-translate "today is a wonderful day"
Translated text: aujourd'hui est un jour merveilleux
```

Fluxos de trabalho CLI de aprendizagem automática

A distância mais curta entre dois pontos é uma linha reta. Da mesma forma, uma ferramenta de linha de comando é muitas vezes a abordagem mais simples para usar o aprendizado de máquina. Na [Figura 11-7](#), pode ver que existem muitos estilos diferentes de técnicas de ML. No caso da aprendizagem automática não supervisionada, pode treinar "on the fly"; noutros casos, pode querer utilizar um modelo treinado durante a noite e colocá-lo no armazenamento de objectos. Além disso, podes querer utilizar ferramentas de alto nível como o AutoML, AI APIS, ou modelos criados por terceiros noutros casos.

Repara que existem muitos domínios problemáticos diferentes em que a adição de ML melhora uma CLI ou é o objetivo completo da CLI. Esses domínios incluem texto, visão computacional, análise comportamental e análise de clientes.

É essencial salientar que existem muitos objectivos para a implementação de ferramentas de linha de comandos que incluem a aprendizagem automática. Um sistema de arquivos como o Amazon EFS, o GCP Filestore ou o Red Hat Ceph tem a vantagem de ser um ponto de montagem Unix centralizado para um cluster. O diretório *bin* pode incluir ferramentas ML CLI fornecidas através de um servidor Jenkins que monta este mesmo volume.

Outros alvos de entrega incluem o Repositório de Pacotes Python (PyPI) e Registros de Contêineres públicos como Docker, GitHub e Amazon. Ainda mais alvos incluem pacotes Linux como Debian e RPM. Uma ferramenta de

linha de comando que empacota o aprendizado de máquina tem uma coleção muito mais extensa de alvos de implantação do que até mesmo um microsserviço, porque uma ferramenta de linha de comando é um aplicativo inteiro.

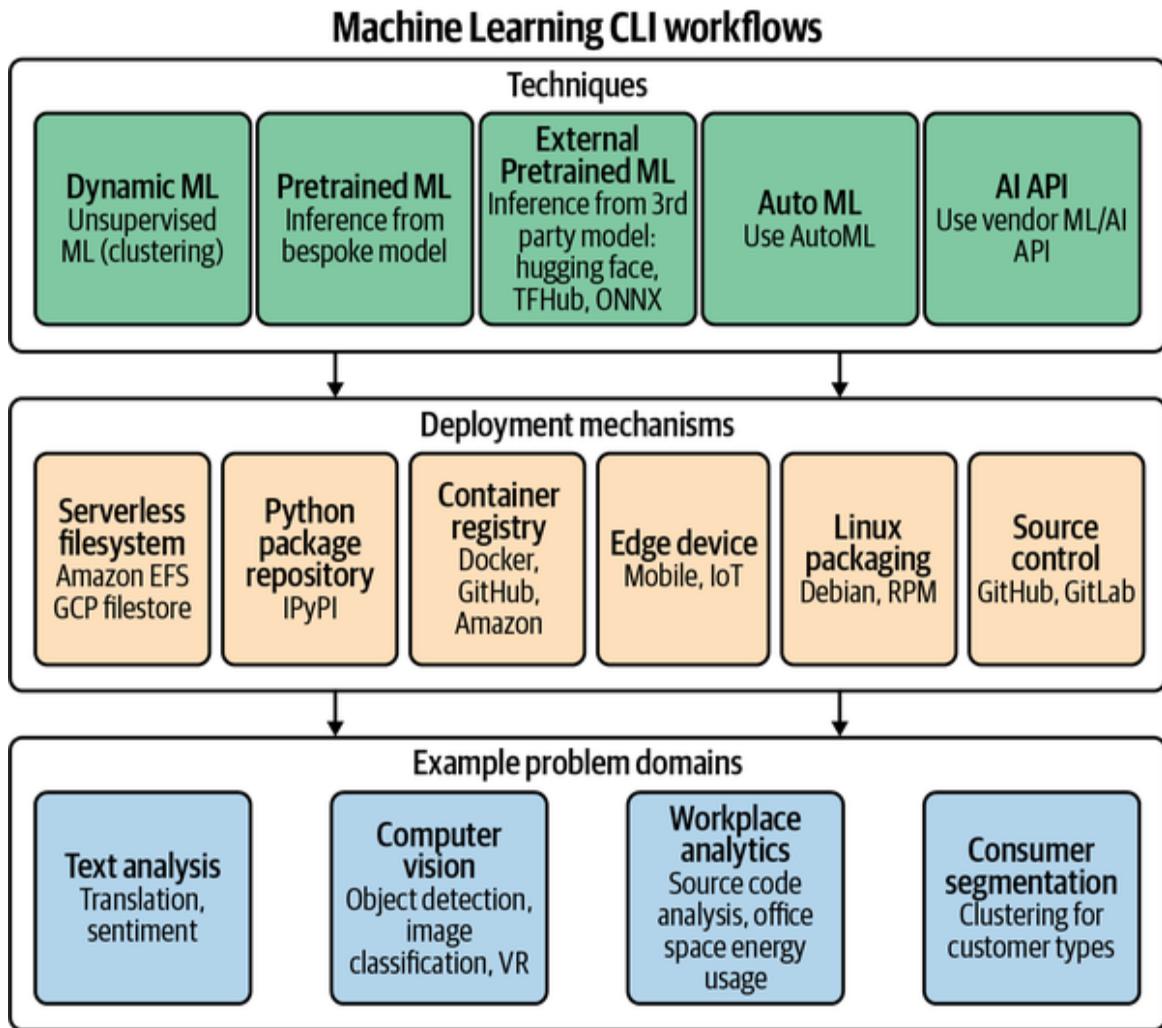


Figura 11-7. Fluxos de trabalho CLI de aprendizagem automática

Alguns bons exemplos de projectos adequados para a aprendizagem automática com um CLI incluem os seguintes recursos:

DevML

O DevML é um projeto que analisa as organizações do GitHub e permite que os profissionais de ML criem as suas próprias previsões de "ML secreto", talvez ligando-o ao streamlit ou incluindo relatórios de agrupamento de programadores no Amazon QuickSight.

Livro de receitas Python MLOps

O [repositório GitHub do Python MLOps Cookbook](#) contém um modelo simples de ML como um conjunto de utilitários. Este projeto tem cobertura detalhada no [Capítulo 7](#).

Aprendizagem automática do preço à vista

Outro exemplo de ML CLI é um projeto sobre Clustering de Aprendizagem Automática de Preço à Vista. [Neste repositório do GitHub](#), diferentes atributos das instâncias spot da AWS, incluindo memória, CPU e preço, são usados para criar clusters de tipos de máquinas semelhantes.

Com estes fluxos de trabalho CLI fora do caminho, vamos passar para a conclusão do capítulo.

Conclusão

Este capítulo descreveu como criar ferramentas de linha de comando desde o início e usar uma estrutura para criar ferramentas para realizar a automação rapidamente. Mesmo quando os exemplos podem parecer triviais, os detalhes e como as coisas funcionam juntas são aspectos essenciais. Ao aprender novos conceitos ou assuntos que os outros geralmente temem (como empacotamento em geral), é fácil sentir-se desencorajado e tentar contorná-los. Embora o Python tenha um longo caminho para um melhor empacotamento, começar não é assim tão difícil, e fazer o trabalho pesado com o empacotamento adequado das tuas ferramentas vai tornar-te inestimável em qualquer equipa. Com ferramentas de empacotamento e linha de comando, estás agora bem posicionado para começar a juntar diferentes serviços para automação.

Este capítulo fez isso aproveitando a Cloud e as suas muitas ofertas de ML: uma poderosa API de tradução da Google. É fundamental lembrar que não é

necessário criar todos os modelos a partir do zero e que deves aproveitar as ofertas dos fornecedores de Cloud sempre que possível, especialmente quando não é uma competência essencial da tua empresa.

Finalmente, quero enfatizar que ser capaz de criar novas soluções para problemas complicados é um superpoder do MLOps, baseado em saber como conectar serviços e aplicações. Como já sabes, utilizar HTTP, ferramentas de linha de comando e aproveitar as ofertas Cloud através dos seus SDKs é uma base sólida para fazer melhorias substanciais em quase todos os ambientes de produção.

No próximo capítulo, abordamos outros detalhes da engenharia de aprendizagem automática e um dos meus tópicos favoritos: estudos de caso. Os estudos de caso são problemas e situações do mundo real em que podes extrair experiências úteis e aplicá-las hoje.

Exercícios

- Adiciona mais algumas opções ao CLI que utilizam funções Cloud, como tornar o URL configurável.
- Descobre como a conta de serviço e o OAuth2 funcionam com o SDK do Google e integra-os em *trigger.py* para evitar a utilização do módulo *subprocess*.
- Melhora a função Cloud traduzindo uma fonte separada, como uma página da Wikipedia.
- Cria uma nova função Cloud que faça o reconhecimento de imagens e fá-la funcionar com uma ferramenta de linha de comandos.
- Faz um fork do repositório [do Python MLOps Cookbook](#) e constrói uma ferramenta CLI contentorizada ligeiramente diferente que publicas num registo público de contentores como o DockerHub ou o GitHub Container Registry.

Questões para discussão sobre pensamento crítico

- Indica uma consequência possível de funções Cloud não autenticadas.
- Quais são algumas das desvantagens de não utilizar um ambiente virtual?
- Descreve dois aspectos de boas técnicas de depuração e porque são úteis.
- Por que razão é útil conhecer as embalagens? Quais são os aspectos críticos da embalagem?
- É uma boa ideia utilizar um modelo existente de um fornecedor de Cloud? Porquê?
- Explica as soluções de compromisso na implementação de uma ferramenta CLI de código aberto alimentada por aprendizagem automática utilizando um registo de contentor público versus a utilização do Repositório de Pacotes Python.

Capítulo 12. Engenharia de aprendizagem automática e estudos de caso de MLOps

Este trabalho foi traduzido com recurso a IA. Agradecemos o teu feedback e comentários:
translation-feedback@oreilly.com

Por Noah Gift

Depois de acompanhar o Professor Loewi durante o seu procedimento, passei mais tempo nos seus cuidados pós-operatórios, durante os quais ele me deu mais palestras. Autografo o meu exemplar do seu pequeno livro de 62 páginas. Por cima da sua assinatura, com uma mão trémula, escreveu: "Factos sem teoria é caos, teoria sem factos é fantasia.

—Dr. Joseph Bogen

Um dos problemas fundamentais da tecnologia no mundo real é que é difícil saber a quem dar conselhos. Em particular, um tópico multidisciplinar como a aprendizagem automática é um desafio intrigante. Como é que podes encontrar a combinação certa de experiência do mundo real, competências actuais e relevantes e a capacidade de ensino para o explicar? Este capítulo tem como objetivo esta capacidade de ensino "unicórnio". O objetivo é destilar esses aspectos relevantes em sabedoria acionável para seus projetos de aprendizado de máquina, como mostrado na [Figura 12-1](#).

Outros domínios sofrem da maldição da complexidade ilimitada que advém de um campo multidisciplinar. Exemplos disso são a Ciência da Nutrição, a Ciência do Clima e as Artes Marciais Mistas. No entanto, uma thread comum é o conceito de um sistema aberto versus um sistema fechado. Um exemplo de brinquedo de um sistema essencialmente fechado é um copo

isolado. Nesse exemplo, é mais fácil modelar o comportamento de um líquido frio, uma vez que o ambiente tem um efeito mínimo. Mas se esse mesmo líquido frio for para o exterior num copo normal, as coisas tornam-se rapidamente confusas. A temperatura do ar exterior, a humidade, o vento e a exposição solar, por si só, criam uma complexidade em cascata na modelação do comportamento desse líquido frio.

Why learning is hard

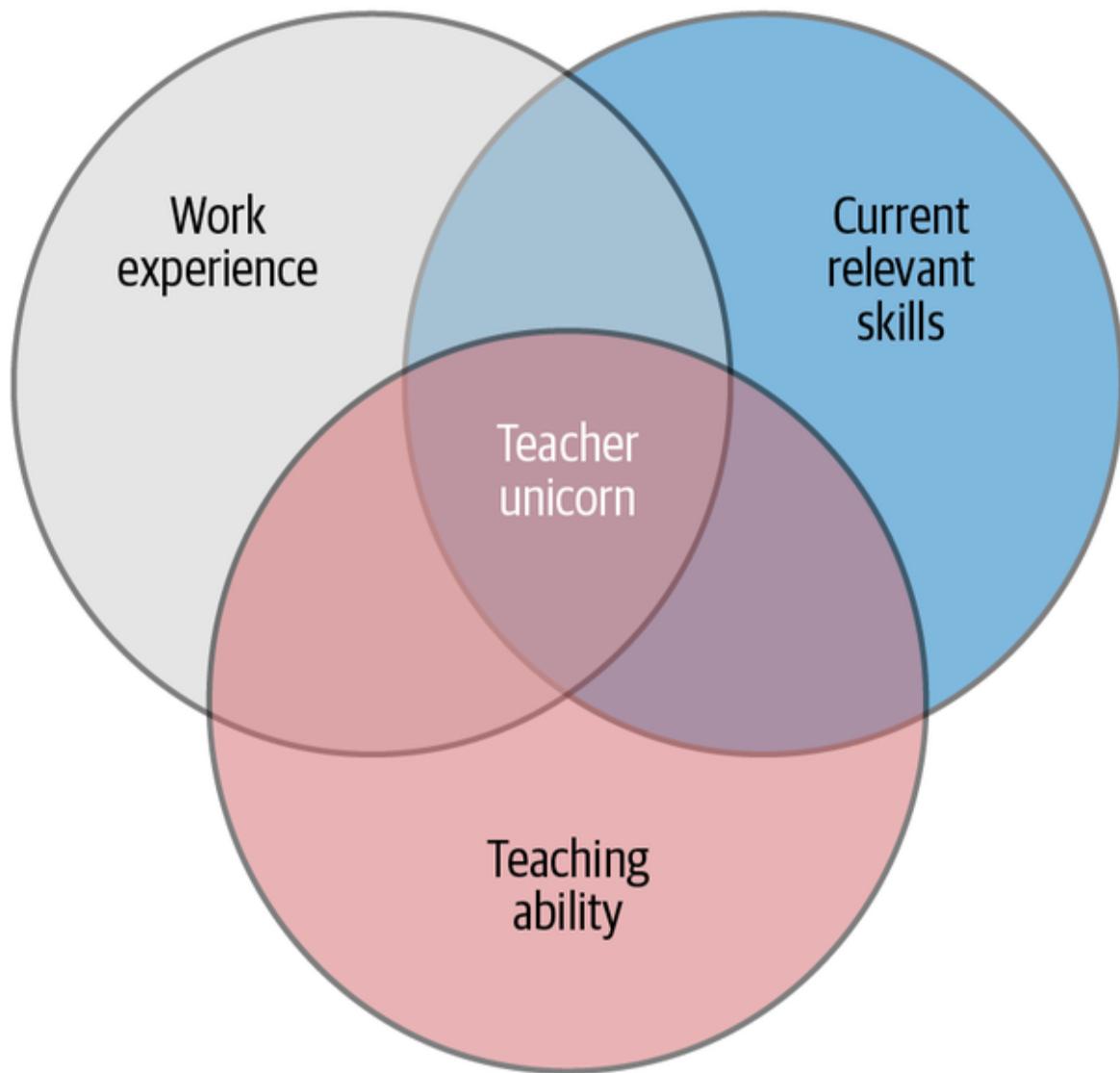


Figura 12-1. Ensina o unicórnio

Este capítulo explora a forma como os MLOps podem tirar partido das aprendizagens destes outros domínios. Explora também a forma como os

sistemas fechados e abertos influenciam o comportamento específico do domínio e, em última análise, como aplicar isto à operacionalização da aprendizagem automática.

Benefícios improváveis da ignorância na construção de modelos de aprendizagem automática

Há muitos benefícios improváveis na ignorância. A ignorância dá-te a coragem para tentares algo desafiante, que se soubesses o quanto difícil era, nunca o terias feito. Desde 2013, a ignorância tem desempenhado um papel crucial em duas coisas que tenho feito em simultâneo: criar um modelo de aprendizagem automática de produção com milhões de dólares em jogo, incluindo uma empresa de 100 pessoas; e aprender, treinar e competir no Jiu-Jitsu brasileiro com lutadores profissionais de topo e atletas olímpicos de luta livre e judo. De certa forma, as duas coisas estão tão interligadas que é difícil separar uma da outra na minha mente. De 2010 a 2013, passei três anos a ter todas as aulas de estatística, probabilidade e modelação que podia ter na UC Davis, no seu programa de MBA, enquanto trabalhava a tempo inteiro em startups em São Francisco. Desde 2017, também leciono aprendizagem automática e computação Cloud na UC Davis Graduate School of Management. Depois de me formar, estava pronto para assumir um cargo de Diretor Geral ou Diretor de Tecnologia e tornei-me um dos primeiros funcionários técnicos de uma rede social de desporto como CTO e Diretor Geral.

Parte da cultura da empresa era que os funcionários se exercitavam juntos num ginásio de artes marciais mistas que também dava aulas de fitness geral. Comecei accidentalmente a treinar Jiu-Jitsu brasileiro porque fiquei curioso ao ver os lutadores profissionais a lutar. Eventualmente, sem dar por isso, trabalhei ao lado de lutadores profissionais e aprendi as noções básicas de submissões. Algumas vezes, até fiquei inconsciente por acidente durante um treino. Lembro-me de uma vez ter pensado: "Se calhar, devia ceder a este estrangulamento de cabeça e braço." Mais tarde, perguntei-me onde

estava; parecia um ginásio, mas não sabia em que ano estávamos. Estaria eu no ginásio do meu liceu no sul da Califórnia? Que idade é que eu tenho? Quando o sangue voltou a entrar no meu cérebro, apercebi-me de que, ah, fui asfixiado e estou no ginásio de artes marciais em Santa Rosa, na Califórnia.

Nos meus primeiros anos de treino, também participei em dois torneios, tendo ganho o primeiro, uma categoria de "principiantes", e perdido uma categoria "intermédia" alguns anos mais tarde. Em retrospectiva, não comprehendia honestamente o que estava a fazer e corria o risco real de sofrer lesões graves. Vi muitas pessoas nos torneios sofrerem lesões graves, incluindo cabeçadas, ombros partidos e ligamentos do joelho rasgados. No meu segundo torneio, já com 40 anos, competi contra um jogador de futebol americano universitário de 20 anos na categoria de peso de 220 libras. No seu último combate, ele entrou numa luta a sério, levou uma cabeçada, estava a sangrar do nariz e estava bastante zangado. Estava apreensivo com o seu estado emocional quando entrei na competição e pensei: "Mas para que raio é que me inscrevi?"

Eu também me dei por feliz por não ter conhecimento do perigo real de competir em artes marciais nos meus trinta e quarenta anos. Continuo a gostar de treinar e aprender Jiu-Jitsu brasileiro, mas é provável que se soubesse o que sei hoje, não teria corrido os riscos que corri com tão pouca perícia como tinha. A ignorância deu-me a coragem de, francamente, correr riscos idiotas, mas também de aprender mais rapidamente.

Da mesma forma, em 2014, eu era igualmente ignorante quando comecei a construir uma infraestrutura de aprendizagem automática para a minha empresa. Hoje em dia, isto é conhecido como MLOps. Na altura, não havia muita informação sobre como operacionalizar a aprendizagem automática. Tal como no Jiu-Jitsu brasileiro, estava ansioso, mas ignorava o que realmente se iria passar e os riscos em jogo. Mais tarde, o terror que senti no Jiu-Jitsu brasileiro não foi nada comparado com o terror que sentiria ao construir sozinho modelos de previsão a partir do zero, com a responsabilidade de milhões de dólares por ano.

No primeiro ano, 2013, a nossa empresa construiu uma rede social de desporto e uma aplicação móvel. Mas, como muitos funcionários de uma startup sabem, construir software é apenas parte do desafio de uma startup; dois outros desafios vitais são a aquisição de utilizadores e a criação de receitas. No início de 2014, tínhamos uma plataforma, mas não tínhamos utilizadores nem receitas. Por isso, precisávamos de arranjar utilizadores rapidamente, ou a startup iria à falência.

A criação de tráfego ocorre geralmente de duas formas para uma plataforma de software. Uma maneira é construir um crescimento orgânico através do boca-a-boca. A segunda forma é comprar publicidade. O problema com a compra de publicidade é que esta pode rapidamente tornar-se uma afetação de custos permanente. O cenário de sonho era que a nossa empresa conseguisse que os utilizadores viessem à nossa plataforma sem comprar anúncios. Tínhamos uma relação com algumas estrelas do desporto, incluindo o antigo quarterback da NFL, Brett Favre. Esta "superestrela" inicial, influenciadora das redes sociais, deu-nos uma visão tremenda sobre a utilização do "growth hacking" orgânico para fazer crescer a nossa plataforma.

A certa altura, este ciclo de feedback de aprendizagem automática levou-nos a atingir os milhões de utilizadores activos mensais. Depois, a equipa jurídica do Facebook enviou-nos um comunicado a dizer que nos iriam "desplantar" metaforicamente. O nosso "crime" foi criar conteúdos desportivos únicos e originais com ligações à nossa plataforma. Há uma questão em curso sobre o potencial poder monopolista das Big Tech, e o crescimento do poder de pirataria do nosso algoritmo de influência chamou a atenção do Facebook. Este foi outro ponto de dados sobre o sucesso real do nosso sistema de previsão. Vamos ver como o fizemos a seguir.

Projectos MLOps na rede social Sqor Sports

Construir uma startup a partir do zero - ou seja, zero funcionários, zero utilizações e zero receitas - é uma tarefa stressante. De 2013 a 2016, passei muitas horas no parque por baixo do edifício Transamerica em São

Francisco, diretamente por baixo do meu escritório, a planear estas coisas. Em particular, o risco epistémico, o risco que eu desconhecia, de apostar milhões de dólares em previsões de aprendizagem automática era aterrador. Mas, em muitos aspectos, os desafios técnicos são muito mais confortáveis do que os psicológicos.

Aqui tens uma visão geral de como o sistema funcionava. Em poucas palavras, o utilizador publica o conteúdo original e depois nós publicamos o conteúdo noutras redes sociais como o Twitter e o Facebook. De seguida, recolhemos as visualizações de página geradas para o nosso site. As visualizações de página tornaram-se o alvo do nosso sistema de previsão de ML. **A Figura 12-2** mostra o pipeline de MLOps para nossa empresa de mídia social.

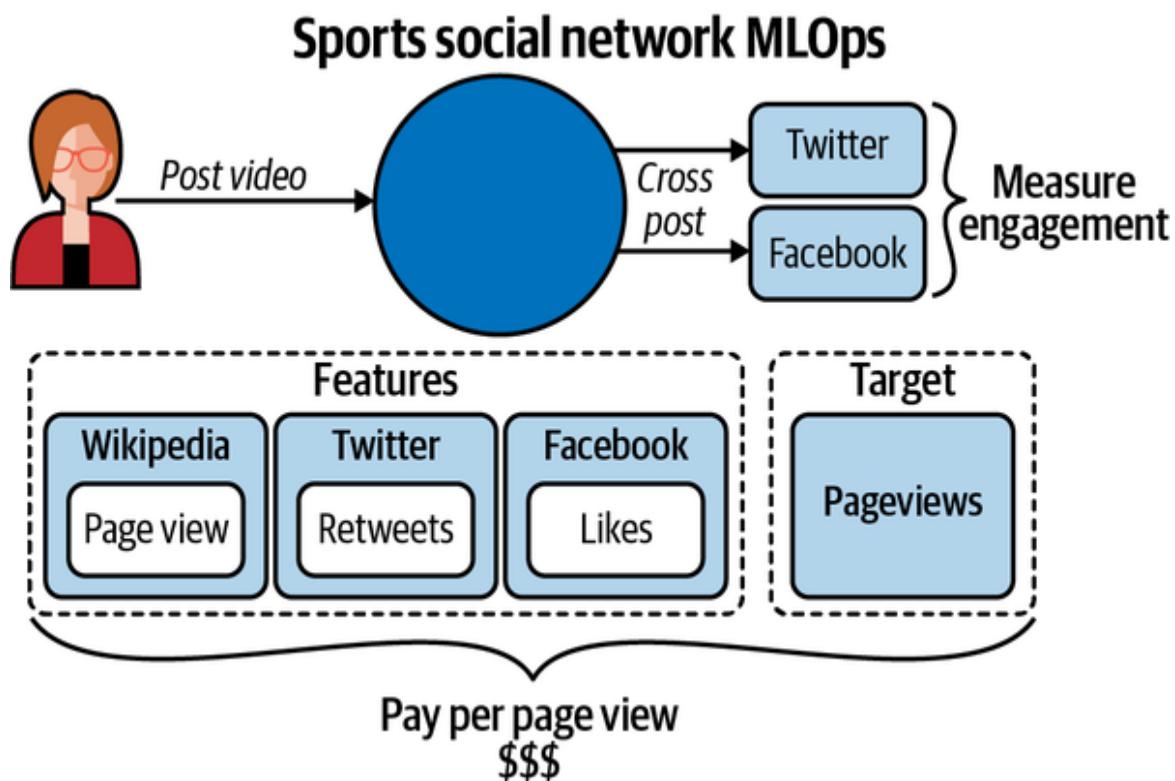


Figura 12-2. Pipeline MLOps para uma rede social de desporto

Mais tarde, recolhemos os sinais das redes sociais destes utilizadores, ou seja, a média de retweets, a média de gostos e as visualizações de páginas da Wikipédia. Estes tornaram-se as características e ajudaram-nos a eliminar o ruído dos dados falsos, como os seguidores "falsos". Depois, pagámos aos

criadores de conteúdos originais, como Conor McGregor, Brett Favre, Tim McGraw e Ashlyn Harris, pelo envolvimento que geraram no nosso site. Acontece que estávamos muito à frente ao fazer isto, e o **Snapchat paga milhões aos utilizadores que publicam conteúdo original na sua plataforma.**

As partes mais difíceis do problema foram a recolha fiável dos dados e a decisão de pagar milhões de dólares com base nas previsões. Ambas eram muito mais complicadas do que eu pensava inicialmente. Por isso, vamos analisar estes sistemas a seguir.

Etiquetagem de dados da Mechanical Turk

Inicialmente, descobrir que os sinais das redes sociais nos davam poder de previsão suficiente para "hackear o crescimento" da nossa plataforma foi um grande avanço. Mas, infelizmente, o desafio mais formidável ainda estava para vir.

Precisávamos de recolher os identificadores das redes sociais de forma fiável para milhares de utilizadores "famosos" das redes sociais. Infelizmente, inicialmente não correu bem e acabou por fracassar completamente. Nossa primeiro processo se parecia com a **Figura 12-3**.

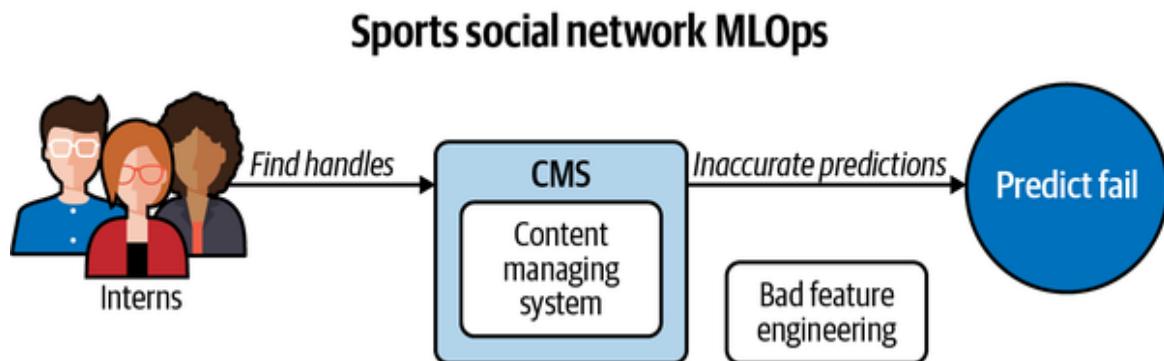


Figura 12-3. Má engenharia de características

Apesar de alguns dos estagiários serem eles próprios futuros jogadores da NFL, o problema crítico era que não tinham formação para introduzir os nomes das redes sociais de forma fiável. Como resultado, era fácil confundir Anthony Davis, o jogador da NFL, com Anthony Davis, o jogador da NBA, e trocar os identificadores do Twitter. Este problema de

fiabilidade da engenharia de características iria destruir a precisão do nosso modelo. Resolvemos isso adicionando automação na forma do Amazon Mechanical Turk. Treinámos um "quórum" de "turkers" para encontrar os identificadores de redes sociais dos atletas e, se 7/9 concordassem, verificámos que isso equivalia a uma precisão de cerca de 99,9999%. A Figura 12-4 mostra o sistema de etiquetagem do Mechanical Turk para a nossa empresa de redes sociais.

NOTA

Em 2014, não se sabia tanto sobre engenharia de dados e MLOps. O nosso projeto de sistema de etiquetagem foi executado por um incrível atleta, programador e antigo graduado da UC Davis, [Purnell Davis](#). Purnell desenvolveu este sistema a partir do zero, entre treinos com atletas como o jogador da NFL Marshawn Lynch ou lutadores profissionais de MMA de 90 quilos, que treinavam com a nossa empresa no mesmo ginásio durante o almoço ou de madrugada.

Mechanical Turk labeling system

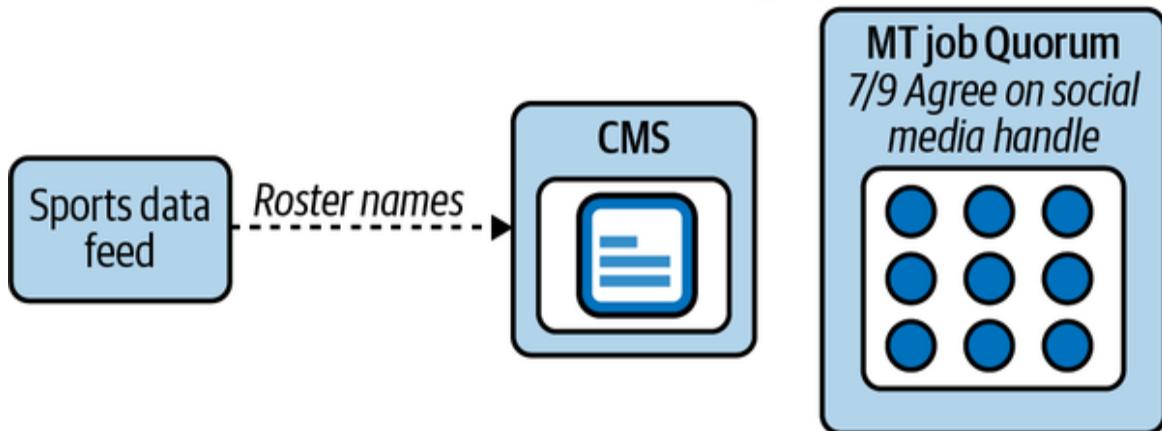


Figura 12-4. Etiquetagem da Mechanical Turk

Classificação do influenciador

Depois de conseguirmos etiquetar os dados, tivemos de recolher dados das API das redes sociais. Podes encontrar um excelente exemplo do tipo de código necessário para integrar isto num [motor de ML](#) neste repositório.

Por exemplo, as estatísticas de LeBron James no Twitter são as seguintes, de acordo com a nossa API de recolha de dados:

```
Get status on Twitter
```

```
df = stats_df(user="KingJames")
In [34]: df.describe()
Out[34]:
   favorite_count retweet_count
count    200.000000  200.000000
mean   11680.670000  4970.585000
std    20694.982228  9230.301069
min     0.000000  39.000000
25%   1589.500000  419.750000
50%   4659.500000  1157.500000
75%  13217.750000  4881.000000
max  128614.000000 70601.000000

In [35]: df.corr()
Out[35]:
   favorite_count retweet_count
favorite_count      1.000000  0.904623
retweet_count        0.904623  1.000000
```

NOTA

Se estás familiarizado com o programa *The Shop* da HBO, conheces Maverick Carter e LeBron James. Nós "quase" trabalhámos oficialmente com eles, mas as conversações não resultaram. Em vez disso, acabámos por desenvolver uma relação com o FC Bayern Munchen como parceiro crítico.

Estes dados de recolha são depois introduzidos num modelo de previsão. Uma palestra que dei no [encontro de R na Bay Area em 2014](#) mostra alguns dos nossos trabalhos manuais. O modelo inicial usou a biblioteca **Caret** do R para prever as visualizações de página. Na [Figura 12-5](#), os algoritmos de previsão originais para encontrar influenciadores foram feitos em R e este gráfico baseado em ggplot mostra a precisão da previsão.

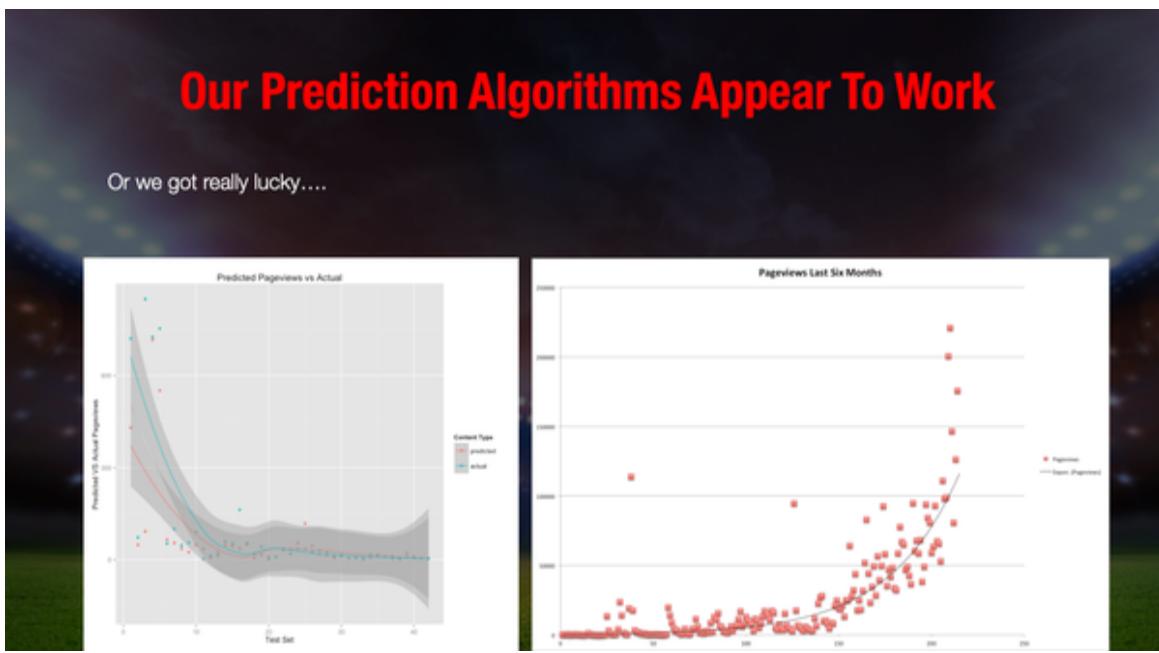


Figura 12-5. Visualizações de página previstas de influenciadores de redes sociais versus visualizações de página reais

Criámos então um modelo de sistema de pagamento utilizando as visualizações de página como métrica alvo. Em última análise, levou a um crescimento exponencial que alimentou uma rápida expansão para milhões de visualizações de páginas por mês. Mas, como referi anteriormente, a parte assustadora foram as noites sem dormir a pensar se estaria a levar a empresa à falência, uma vez que gastámos milhões de dólares nas previsões que ajudei a criar.

Inteligência do atleta (produto de IA)

Com um pipeline central de MLOps a servir previsões e a alimentar o nosso crescimento sem comprar anúncios, começámos a evoluir o nosso produto para um produto de IA chamado "Athlete Intelligence". Tínhamos dois gestores de produtos de IA que geriam este produto a tempo inteiro. A ideia geral do produto principal era que os "influenciadores" compreendessem o que podiam esperar em termos de pagamento e que as marcas utilizassem este painel de controlo para estabelecer parcerias diretamente com os atletas. Na [Figura 12-6](#), utilizámos a aprendizagem automática não

supervisionada para categorizar diferentes aspectos dos atletas, incluindo a sua influência nas redes sociais.

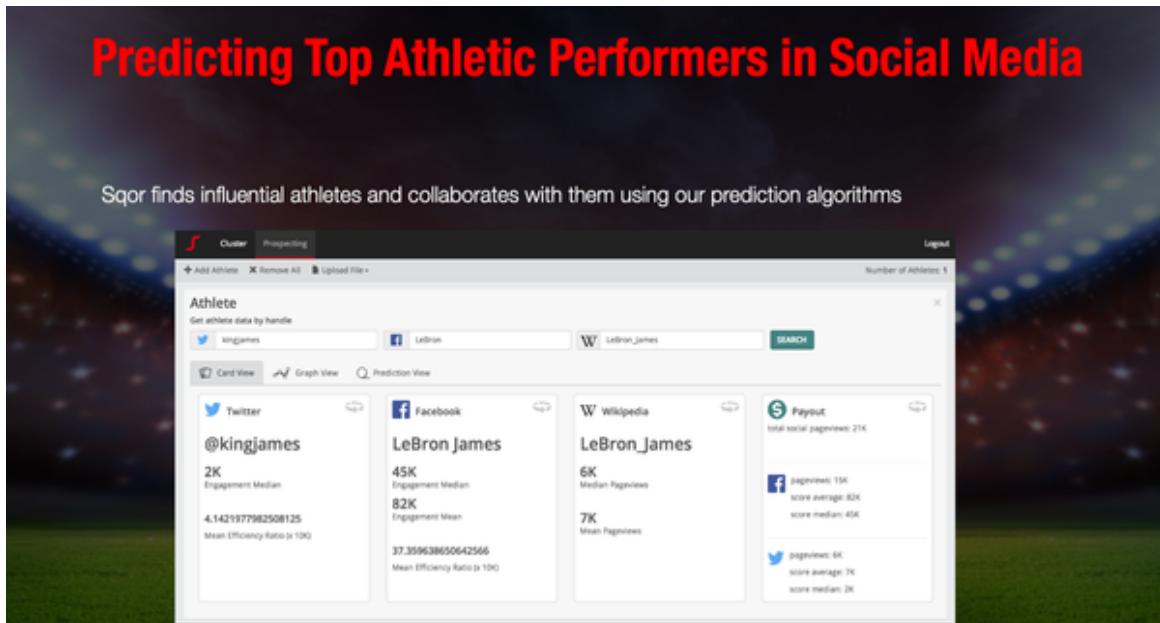


Figura 12-6. Inteligência do atleta

As características adicionais incluíram aprendizagem automática não supervisionada que "agrupou" perfis sociais de atletas semelhantes. Esta funcionalidade permitiu-nos agrupar diferentes tipos de atletas em pacotes de marketing de influenciadores(Figura 12-7).

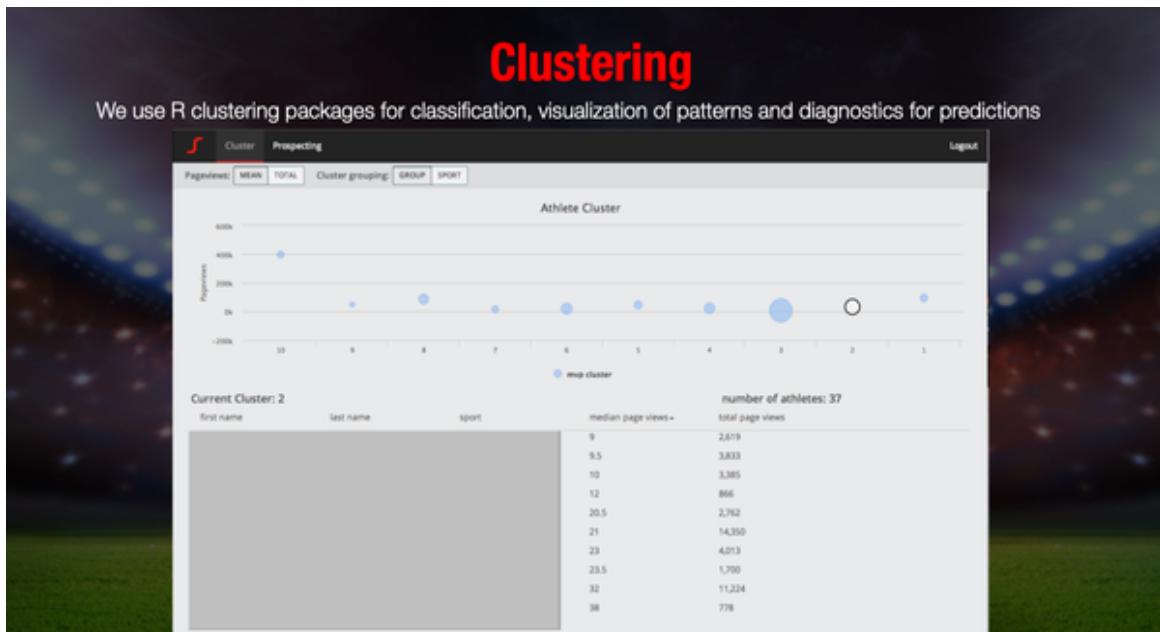


Figura 12-7. Painel de controlo do agrupamento de inteligência do atleta

Finalmente, este crescimento levou a dois novos produtos de receita para a nossa empresa. O primeiro foi uma plataforma de merchandising construída sobre o Shopify que se transformou num negócio de 500 mil dólares por ano e o segundo foi um negócio de marketing de influenciadores multimilionário. Contactámos diretamente as marcas e ligámo-las aos influenciadores na nossa plataforma. Um excelente exemplo disto é o anúncio "Game of War" que intermediámos com a **Machine Zone** e o **Conor McGregor**.

Em suma, ter um pipeline MLOps eficaz levou-nos a ter utilizadores e receitas. Sem o modelo de previsão, não teríamos utilizadores; sem os utilizadores, não teríamos receitas; sem receitas, não teríamos negócio. Os MLOps práticos pagam dividendos.

Vamos falar mais sobre o sistema aberto e o sistema fechado na próxima secção. As coisas parecem simples num ambiente controlado, como a ciência de dados académica, o ginásio de artes marciais num IG ou conselhos teóricos sobre nutrição, como calorias in versus calorias out. No entanto, os sistemas abertos ou do mundo real têm muitas influências adicionais, que são muito mais difíceis de controlar.

A técnica perfeita versus o mundo real

Será que um armlock (um ataque de braço que parte um braço através da hiperextensão do cotovelo) feito por um cinturão negro de Jiu-Jitsu brasileiro com o GI (uniforme de casaco de algodão pesado nas artes marciais) tem a mesma eficácia contra um adversário que não usa GI? Da mesma forma, na aprendizagem automática, o que é mais importante? É a precisão, a técnica, ou a capacidade de fornecer valor a um cliente ou conhecimento a uma situação?

NOTA

Se tens curiosidade em saber como é usar um armbar em competição, procura no Google "Ronda Rousey armbar" e verás um mestre técnico a trabalhar.

O que dizes de uma luta de rua contra um combate de grappling ou de um combate da UFC contra um combate de grappling? Mesmo os não especialistas concordam que quanto mais regras estiverem envolvidas numa técnica, menos hipóteses tem de ser bem sucedida num ambiente com poucas ou nenhuma regras, ou seja, o mundo real.

Aprendi uma lição ao viajar para academias aleatórias para treinar nos Estados Unidos quando estava em negócios. Em mais de uma ocasião, um faixa-marrom ou preta especialista em Jiu-Jitsu Brasileiro GI treinava comigo em NOGI (sem uniforme) e imediatamente tentava executar um armlock. Passei anos a treinar com lutadores profissionais que praticam NOGI. Depois de ter sido apanhado talvez centenas de vezes na submissão, aprendi a escapar-lhe facilmente deslizando o meu braço para fora de uma forma específica. Como é que eu percebi isso? Simplesmente copiei o que os lutadores profissionais fizeram e me ensinaram e pratiquei-o vezes sem conta.

Muitos especialistas em GI, de faixas pretas a castanhas e medalhistas olímpicos, disseram coisas semelhantes. "Não podes fazer isto na GI", ou "eu teria feito isto na GI", ou "não devias fazer isto", etc., com um ar horrorizado ou surpreendido. O seu "modelo perfeito" não funcionou e a sua visão do mundo ficou chocada. Queria isto dizer que eu era tão bom como estes especialistas? Não, significava que eles tinham uma técnica muito melhor do que a minha, num contexto que não se aplicava àquele em que nos encontrávamos atualmente, o grappling sem uniforme. Repara na **Figura 12-8** como fechar o sistema, ou adicionar regras, ao mundo exterior acrescenta mais controlo mas menos realismo.

Martial arts techniques in the real world

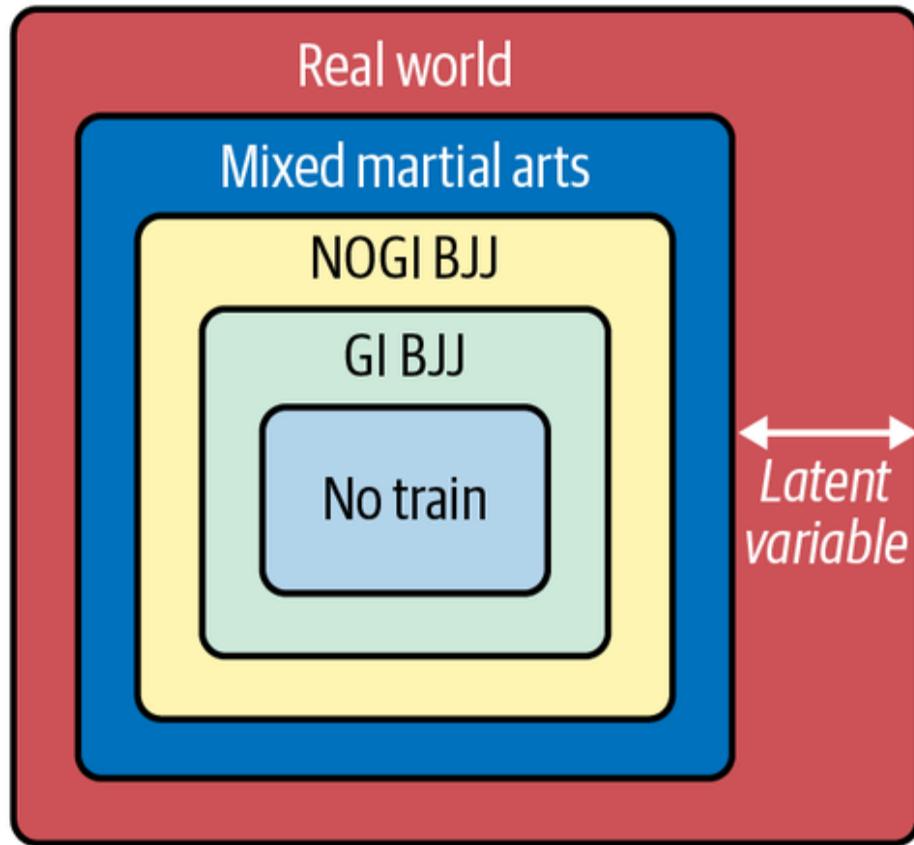


Figura 12-8. Técnicas de artes marciais no mundo real

No livro *Critical Thinking* (MIT Press), Jonathan Haber refere como Ann J. Cahill e Stephan Bloch-Schulman, da Universidade de Elon, fazem das suas aulas de ensino superior um estúdio de artes marciais:

Nessas aulas [de artes marciais], em cada nível sucessivo de avaliação, os alunos também têm de demonstrar que mantiveram as capacidades que alcançaram nos níveis de cinto anteriores. É importante notar que um sensei decente não atribui um cinto com base no esforço: o facto de um aluno se ter esforçado muito para dominar uma determinada ação não é relevante. A questão é: o aluno consegue dar o murro?

NOTA

Ao explicar essa mesma teoria de MMA (artes marciais mistas) e aprendizado de máquina em uma aula que dou no programa de ciência de dados da Northwestern, descobri que um dos meus alunos passou 14 anos na NFL. Ele então apontou o que admirava no MMA, porque os melhores atletas não têm agendas técnicas. Em vez disso, espera para ver o que acontece e aplica a técnica correta para a situação em questão.

ENTREVISTA: ARTES MARCIAIS E MODELAÇÃO ML

Entrevistei um cinturão negro de topo e antigo lutador de artes marciais mistas para discutir outras artes marciais do mundo real e a sua relação com a modelação de ML:

P: Quem és tu e qual é a tua experiência em artes marciais, desporto e luta?

R: O meu nome é Jacob Hardgrove. Treino Jiu-Jitsu brasileiro há cerca de 20 anos. Já competi em eventos de submission grappling e MMA profissional e tenho um recorde de MMA de 3-0. Para além do treino de Jiu-Jitsu, para me preparar para os eventos de MMA, passei uma quantidade significativa de tempo a estudar e a treinar outras formas de artes marciais, como o boxe ocidental, muay Thai, estilo livre e luta livre.

P: Porque é que algumas técnicas ensinadas de forma muito formal, como o GI BJJ, não funcionam tão bem num combate real, seja na rua ou na jaula?

R: O meu processo de reflexão sobre esta questão leva-me a considerar uma dicotomia entre a introdução de um conceito/técnica e a implementação bem sucedida desse conceito/técnica num ambiente/situação não controlado.

Para introduzir uma técnica ou um conceito a um principiante, o instrutor despoja a técnica do maior número possível de variáveis para criar uma estrutura facilmente repetível. Esta estrutura deve ser tão despojada quanto possível e consistir nos componentes primários e essenciais necessários para alcançar o efeito desejado. Esta estrutura, de acordo com a maioria das artes marciais tradicionais, é referida como "Kata".

Agora, vejo pessoas que acabam por se submergir no mundo do Jiu-Jitsu perderem de vista o facto de estarem a praticar habilidades marciais contra outros artistas marciais e não contra atacantes

violentos não treinados, mas potencialmente ainda viáveis. Infelizmente, esta verdade pode levar a um caso grave de "perder o objetivo".

Tal como Jacob, o cinturão negro ex-lutador de MMA, vejo problemas semelhantes na aprendizagem automática do mundo real. Como professor, dou aulas de aprendizagem automática em universidades de topo e sou também um profissional do sector. As ferramentas académicas como o [sklearn](#) e [o pandas](#) e os conjuntos de dados educativos no Kaggle são os "Kata". Estes conjuntos de dados e ferramentas são necessários para "despir" a complexidade do mundo real e ensinar a matéria. Um próximo passo fundamental é consciencializar o aluno de que os Kata de aprendizagem automática não são o mundo real. O mundo real é muito mais perigoso e complicado.

Desafios críticos nos MLOps

Vamos discutir alguns desafios específicos na introdução da aprendizagem automática na produção. Os três principais desafios são as consequências éticas e não intencionais, a falta de excelência operacional e o enfoque na precisão das previsões em detrimento do panorama geral. A excelência operacional na implementação da aprendizagem automática é muito mais importante do que a técnica no mundo real. Este ponto exato é claramente articulado no artigo de pesquisa "[Hidden Technical Debt in Machine Learning Systems](#)". Os autores do artigo descobriram que "...é comum incorrer em custos de manutenção contínuos e maciços nos sistemas de aprendizagem automática do mundo real". Vamos discutir algumas destas questões de seguida.

Consequências éticas e não intencionais

É fácil ficarmos "de mãos a abanar" ou auto-justificados em relação à ética e afastarmos as pessoas. Este facto não significa que o tema não deva ser

discutido. Muitas empresas que lidam com as redes sociais criaram armas de desinformação em massa. De acordo com [Tristan Harris](#), "64% dos grupos extremistas a que as pessoas aderiram deveram-se ao sistema de recomendação do próprio Facebook" e o YouTube "recomendou vídeos de Alex Jones, InfoWars e teorias da conspiração 15 mil milhões de vezes. Isso é mais do que o tráfego combinado do Washington Post, BBC, Guardian e Fox News juntos", e 70% de todo o tempo de visualização no YouTube são recomendações.

As consequências já não são teóricas. Muitas redes sociais e grandes empresas de tecnologia estão a mudar ativamente a sua abordagem aos sistemas baseados na aprendizagem automática. Estas actualizações vão desde a suspensão dos sistemas de reconhecimento facial até à avaliação mais rigorosa dos resultados dos motores de recomendação. As considerações éticas no mundo real também têm de fazer parte de uma solução MLOps, e não apenas o facto de a técnica ter poder de previsão.

Já aparecem algumas soluções para o problema das câmaras de eco - por exemplo, o [IEEE Spectrum](#) no artigo "[Smart Algorithm Bursts Social Networks Filter Bubbles](#)". Uma equipa de pesquisadores da Finlândia e da Dinamarca tem uma visão diferente de como as plataformas de redes sociais podem funcionar. Desenvolveram um novo algoritmo que aumenta a diversidade da exposição nas redes sociais, garantindo ao mesmo tempo que o conteúdo é amplamente partilhado."

É nesta situação que a ética entra em jogo. Se uma empresa ou um engenheiro de aprendizagem automática estiver 100% concentrado em aumentar o envolvimento e o lucro, pode não querer perder 10% do seu lucro para "salvar o mundo". Este dilema ético é um caso clássico de estudo de externalidades. Uma central nuclear pode proporcionar enormes benefícios energéticos, mas se despejar os resíduos nucleares no oceano, outras vítimas inocentes pagam o preço, mas não recebem qualquer recompensa.

Falta de excelência operacional

Outro antipadrão da engenharia de aprendizagem automática é a incapacidade de manter um modelo de aprendizagem automática de forma eficaz. Uma boa maneira de raciocinar sobre esta situação é considerar a construção de uma estante de madeira versus o cultivo de uma figueira. Se fizeres uma estante, provavelmente nunca mais a modificarás; uma vez concebida, o projeto termina. Por outro lado, uma figueira faz parte de um sistema aberto. Precisa de água, sol, solo, nutrientes, vento, poda dos ramos e proteção contra insectos e doenças. A figueira adapta-se ao ambiente ao longo do tempo, mas precisa de supervisão da pessoa que quer desfrutar dos figos mais tarde. Por outras palavras, a figueira precisa de manutenção para produzir frutos de alta qualidade. Um sistema de aprendizagem automática, como qualquer sistema de software, nunca está terminado como uma estante de livros. Em vez disso, precisa de ser alimentado como uma figueira.

De seguida, imaginemos uma empresa que prevê limites de crédito para cada novo cliente de uma loja de mobiliário. Digamos que um modelo inicial foi desenvolvido em janeiro de 2019. Desde então, muita coisa mudou. Pode haver um **desvio** substancial dos **dados** das características utilizadas para criar o modelo. Sabemos que, em 2020-2021, muitas lojas de retalho encerraram as suas actividades e que o modelo de negócio dos centros comerciais está seriamente ameaçado. A COVID-19 acelerou uma tendência de declínio do retalho físico, entre outras actualizações dos dados subjacentes. Como resultado, os dados "derivaram", ou seja, os padrões de compra são muito diferentes após a COVID-19, e o modelo de aprendizagem automática estático original pode não funcionar como pretendido.

Um engenheiro de aprendizagem automática seria sensato se tivesse em conta a constante reciclagem do modelo (alimentando a figueira), definindo alertas sobre as alterações de dados quando ocorre um limiar de desvio de dados. Além disso, a monitorização das métricas comerciais poderia enviar um alerta se os pagamentos em atraso ultrapassassem um determinado limite. Por exemplo, imaginemos que, normalmente, uma empresa que vende mobiliário tem 5% de clientes que pagam com 30 dias de atraso o seu

crédito. Se, de repente, 10% dos clientes pagarem com 30 dias de atraso, então o modelo de ML subjacente pode precisar de uma atualização.

O conceito de alertas e monitorização para a aprendizagem automática não deve ser uma surpresa para os engenheiros de software tradicionais. Por exemplo, monitorizamos a carga da CPU em servidores individuais, a latência para utilizadores móveis e outros indicadores-chave de desempenho. Com as operações de aprendizagem automática, aplica-se o mesmo conceito.

Concentra-te na precisão da previsão versus o panorama geral

Como discutido anteriormente no capítulo, até mesmo os praticantes de classe mundial podem, por vezes, concentrar-se na técnica em detrimento do quadro geral. Mas, tal como o "braço de ferro perfeito", que não funciona bem numa situação diferente, um modelo pode ser igualmente frágil. Um ótimo exemplo deste braço de ferro é o debate entre Nate Silver e Nassim Taleb. Issac Faber [tem uma excelente análise dos pontos críticos](#) num artigo no Medium. Isaac salienta que nem todos os modelos são réplicas perfeitas do mundo real, porque não conseguem eliminar a incerteza do que não sabemos, ou seja, o risco incomensurável. O debate entre Nate Silver e Nassim Taleb resume-se a saber se é possível modelar as eleições ou se é uma ilusão pensar que podemos modelá-las. Os acontecimentos eleitorais de 2020 parecem colocar pontos a favor de Nassim Taleb.

O cinturão negro Jacob diz a mesma coisa quando refere que um Kata é uma versão simplificada de uma técnica de treino. A complexidade do mundo real e o exercício de treino estão em conflito inerente. O modelo ou a técnica podem ser perfeitos, mas o mundo real pode não se importar. Por exemplo, no caso das eleições presidenciais de 2020, mesmo o melhor modelador não previu a probabilidade de uma insurreição ou de funcionários do Estado pressionados a anular os votos. Como Issac descreve este problema, é a diferença entre incerteza aleatória e epistémica. O risco aleatório pode ser medido, por exemplo, a probabilidade de cara ou

coroa, mas o risco epistémico não pode ser medido, como uma insurreição que poderia anular uma eleição presidencial.

O Dr. Steven Koonin, para quem trabalhei no Caltech durante vários anos, diz algo semelhante no seu livro sobre a ciência do clima. Tal como as previsões eleitorais, a ciência nutricional e outros sistemas complexos, o tema da ciência climática é instantaneamente polarizador. No livro *Unsettled* (BenBella Books), Koonin diz: "Não só podemos ser enganados por não termos uma visão global do clima ao longo do tempo, como também podemos ser enganados por não termos uma visão global do planeta". Além disso, continua: "As projecções de futuros acontecimentos climáticos e meteorológicos baseiam-se em modelos comprovadamente inadequados para o efeito". Quer acredites ou não na sua afirmação, vale a pena considerar a complexidade da modelação de um sistema complexo no mundo real e o que pode correr mal.

Um bom resumo desse dilema é ter cuidado com a confiança em uma previsão ou técnica. Um dos grapplers mais assustadores e talentosos com quem treinei, Dave Terrell, que lutou pelo campeonato da UFC, disse-me: nunca entres numa luta com vários adversários. Quanto mais habilidade e "pele no jogo" um praticante tem, mais se torna consciente do risco epistémico. Porquê arriscar a tua vida numa luta de rua com várias pessoas, mesmo que sejas um artista marcial de classe mundial? Da mesma forma, porquê ter mais confiança do que deveria quando prevê uma eleição, os preços das acções ou ossistemas naturais?

Na prática, a melhor forma de abordar este problema na produção é limitar a complexidade da técnica e assumir um menor conhecimento da incerteza epistémica. Esta conclusão pode significar que um modelo tradicional de aprendizagem automática com elevada capacidade de explicação é melhor do que um modelo complexo de aprendizagem profunda com uma precisão marginalmente melhor.

ENTREVISTA COM OS PRATICANTES DE MLOPS: PIERO MOLINO

Qual é a tua formação e como é que te envolveste na operacionalização da aprendizagem automática?

R: Estudei informática na Universidade de Bari, em Itália, onde me doutorei a trabalhar em resposta a perguntas de domínio aberto (na intersecção de PNL, ML e recuperação de informação). Desde que era estudante, tentava utilizar os sistemas de pesquisa que estava a construir para criar produtos que pudessem ser utilizados no mundo real, em vez de protótipos de laboratório que nunca foram utilizados na prática. Isto levou-me a trabalhar em PNL e ML aplicados em algumas empresas (Yahoo!, IBM Watson, Geometric Intelligence e Uber AI). Na maioria dos casos, fiz tanto pesquisa como aplicações, que é a intersecção onde gosto de trabalhar: novas ideias que acabam por ser utilizadas pelas pessoas. Na Uber AI, em particular, toquei em muitas aplicações de ML na empresa (apoio ao cliente, tempo previsto de entrega, recomendação de comida e restaurantes, sistemas de diálogo com motoristas) e desenvolvi o Ludwig para facilitar a minha vida ao saltar de um projeto para o outro, porque me permitiu evitar reinventar a roda (pré-processamento de dados, um ciclo de treino, funções de avaliação, etc.) de cada vez, mas tornou possível criar protótipos em minutos.

Quais são 3-5 dos aspectos mais importantes a ter em conta na implementação e manutenção de sistemas de aprendizagem automática em escala?

R: A minha experiência ensinou-me que o modelo em si é apenas uma peça relativamente pequena dentro de um sistema mais vasto quando o produzes, mas ao mesmo tempo é uma das mais críticas (se o modelo não fizer previsões satisfatórias, todo o resto é inútil) e uma das mais complexas de acertar, porque os dados, a infraestrutura e a monitorização (que são muito importantes) têm

padrões mais estabelecidos e são mais concebidos, pelo que há menos incerteza no resultado do que na tarefa de construção do modelo. Essa incerteza tem de ser tida em conta no processo de desenvolvimento do ML.

Outra aprendizagem importante é que a realidade da implementação de um sistema de ML e da observação do comportamento dos utilizadores quebra frequentemente os nossos pressupostos, pelo que o processo se torna iterativo. Por exemplo, os dados de entrada em tempo real que o sistema recebe podem acabar por ser muito diferentes dos dados de formação, a distribuição das entradas e saídas pode mudar ao longo do tempo e o facto de o sistema estar a ser utilizado em muitos casos terá impacto na distribuição dos dados que serão recolhidos (pensa num sistema de recomendação; os dados recolhidos após a sua primeira adoção serão certamente influenciados pelas sugestões do próprio sistema de recomendação). Por esta razão, a monitorização torna-se extremamente importante.

Finalmente, o processo de recolha de dados é também extremamente importante. Na minha experiência, trabalhei em projectos em que os dados de treino foram obtidos através de etiquetagem e em que os dados foram obtidos como subproduto de um processo que ocorreu dentro da organização. No primeiro caso, definir com precisão o processo de recolha de dados, dar instruções exactas aos anotadores, observar as suas anotações e o seu acordo e iterar o processo foi muito mais complicado do que simplesmente enviar os dados para serem anotados e recebê-los de volta. Neste último caso, compreender profundamente o processo que gera os dados era literalmente a única forma de lhes dar sentido, e ajudava a identificar os valores atípicos, a definir corretamente as expectativas do modelo e a compreender a incerteza das previsões do modelo. Na minha experiência, este grau de compreensão, juntamente com a análise das previsões do modelo, levou também a melhorar o próprio processo de recolha de dados, o que, por sua vez, conduziu

a dados menos ruidosos que foram utilizados para treinar melhores modelos, pelo que há aqui um ciclo virtuoso em jogo.

O que te entusiasma mais neste momento com a aprendizagem automática e porquê?

R: Há duas coisas que me entusiasmam mais. Por um lado, tens a percolação do AM noutros campos científicos, como a biologia, a química e a física, mas também em indústrias inteiramente novas, como a gráfica e os jogos de vídeo (AM para geração de conteúdos, renderização e animação, por exemplo). Por outro lado, tens aquilo em que estou a trabalhar ativamente (não trabalharia se não me entusiasmasse), que é a criação de ferramentas e abstrações que facilitem a utilização do AM para os seus fins por pessoas sem conhecimentos de AM. As duas coisas cruzam-se bem; se mais pessoas, talvez especialistas nos seus domínios, puderem aceder e utilizar o ML, a percolação será mais rápida.

Quais são as 3-5 principais coisas que uma pessoa que está a ler este livro pode fazer para ter sucesso na sua carreira nos MLOps?

R: Aprende a lidar com a incerteza associada ao processo de desenvolvimento do ML, aprende a trabalhar em equipas multifuncionais que incluem pessoas com pouca ou nenhuma experiência em ML, cria processos repetíveis, aprende a colocar-se no lugar dos utilizadores, esforça-se por evitar dívidas técnicas.

Como é que as pessoas podem entrar em contacto contigo e o que gostarias de partilhar sobre o que estás a fazer?

R: Não sou um utilizador ávido das redes sociais, mas ocasionalmente publico informações sobre o que estou a fazer no Twitter (@w4nderlus7) e no LinkedIn. Atualmente, estou a cogerir a **série de seminários MLSys de Stanford**, que julgo ser do interesse da maioria dos leitores. Também tenho um **site pessoal** que actualizo com os meus projectos, as minhas publicações e algumas

palestras que dou. O meu principal interesse atual é a construção de ferramentas e abstrações para tornar o ML mais acessível a pessoas com menos ou nenhum conhecimento de ML, para que possam aproveitar o seu conhecimento do domínio para treinar e usar modelos para atingir os seus objetivos.

ENTREVISTA COM PROFISSIONAIS DE MLOPS: FANCESCA LAZZERI

Qual é a tua formação e como é que te envolveste na operacionalização da aprendizagem automática?

R: Tenho formação em Economia. Antes de entrar para a Microsoft, fui bolseiro de pesquisa em Economia Empresarial na Universidade de Harvard, onde realizava análises estatísticas e econométricas na Unidade de Gestão de Tecnologia e Operações. Em Harvard, trabalhei em vários projectos baseados em dados de patentes, publicações e citações para investigar e medir o impacto das redes de conhecimento externas na competitividade e inovação das empresas. Esta experiência deu-me a oportunidade única de aprender a extraír conhecimentos de grandes volumes de dados, a construir modelos preditivos de raiz e a programar em R e Python. Este foi o meu primeiro passo no fantástico mundo da aprendizagem automática!

Alguns anos mais tarde, comecei a minha carreira na Microsoft como cientista de dados: na minha função na Microsoft, ajudava as empresas a transformar as suas operações através de algoritmos de aprendizagem automática e IA e, em particular, estava encarregue de passar os seus modelos do desenvolvimento para um ambiente de produção de uma forma robusta, rápida e repetível.

Quais são 3-5 dos aspectos mais importantes a ter em conta na implementação e manutenção de sistemas de aprendizagem automática em escala?

R: Muitas empresas encaram a implementação da aprendizagem automática como uma prática técnica. No entanto, é mais uma iniciativa orientada para o negócio que começa dentro da empresa; para se tornar uma empresa orientada para a IA, é importante que as pessoas que hoje operam com sucesso e compreendem o negócio colaborem estreitamente com as equipas que são responsáveis pelo

fluxo de trabalho de implementação da aprendizagem automática. É essencial manter uma interação constante para compreender o processo de experimentação do modelo em paralelo com as etapas de implementação e consumo do modelo. Além disso, os modelos de aprendizagem automática devem ser treinados com dados históricos, o que exige a criação de um pipeline de dados de previsão, uma atividade que requer várias tarefas, incluindo processamento de dados, engenharia de recursos e ajuste. Cada tarefa, até às versões das bibliotecas e ao tratamento dos valores em falta, deve ser exatamente duplicada do ambiente de desenvolvimento para o ambiente de produção. Por vezes, as diferenças entre as tecnologias utilizadas no desenvolvimento e na produção contribuem para dificultar a implantação de modelos de aprendizagem automática. Por último, as linguagens da ciência dos dados podem ser lentas. Python é uma das linguagens mais populares para aplicações de aprendizagem automática, mas os modelos de produção completos raramente são implantados nessas linguagens por razões de velocidade. Transferir um modelo Python para uma linguagem de produção como C++ ou Java é um desafio e resulta frequentemente numa redução do desempenho do modelo original treinado.

O que te entusiasma mais neste momento com a aprendizagem automática e porquê?

R: Estou muito entusiasmado com a adoção generalizada de estruturas de código aberto (como a [Fairlearn](#) e a [InterpretML](#)) para apoiar a interpretabilidade transparente e garantir a equidade dos algoritmos de aprendizagem automática. Os cientistas de dados sabem que a precisão já não é a única preocupação quando se desenvolvem modelos de aprendizagem automática; a interpretabilidade e a equidade também têm de ser consideradas. Para garantir que as soluções de aprendizagem automática são justas e que o valor das suas previsões é fácil de compreender e explicar, é essencial criar ferramentas de código aberto que os programadores e

os cientistas de dados possam utilizar para avaliar a equidade do seu sistema de aprendizagem automática e atenuar quaisquer problemas de equidade observados.

Quais são as 3-5 principais coisas que uma pessoa que está a ler este livro pode fazer para ter sucesso na sua carreira nos MLOps?

R: Podes:

1. Cria pipelines de ML reproduzíveis.
2. Captura os dados de governação para o ciclo de vida do ML de ponta a ponta.
3. Monitoriza as aplicações ML para questões operacionais e relacionadas com ML.

Como é que as pessoas podem entrar em contacto contigo e o que gostarias de partilhar sobre o que estás a fazer?

R: Podes seguir-me no Twitter [@frlazzeri](#), no [LinkedIn](#) e no [Medium](#)). Recentemente, escrevi um livro, *Machine Learning for Time Series Forecasting with Python* (Wiley, 2020), no qual podes encontrar exemplos do mundo real, recursos e estratégias concretas para explorar e transformar dados e desenvolver previsões de séries temporais práticas e utilizáveis. Na Microsoft, lidero uma equipa internacional (nos EUA, Canadá, Reino Unido e Rússia) de engenheiros e defensores de programadores Cloud, gerindo uma grande carteira de clientes. A minha equipa é responsável pela criação de conteúdo técnico e soluções automatizadas inteligentes no Azure, utilizando técnicas que vão desde a IoT, previsão de séries temporais, visão computacional, processamento de linguagem natural e estruturas de código aberto. Atualmente, também ensino "Introdução à IA com Python" na Universidade de Columbia. Podes ler mais sobre a minha filosofia e experiência de ensino no meu artigo, ["The Importance of Teaching Machine Learning"](#).

Recomendações finais para a implementação dos MLOps

Antes de terminarmos, queremos dar-te algumas dicas para implementares os MLOps na tua organização. Em primeiro lugar, apresentamos-te um conjunto de recomendações finais e globais:

- Começa com pequenas vitórias.
- Usa a Cloud, não lutes contra a Cloud.
- Certifica-te a ti e à tua equipa numa plataforma Cloud e numa especialização em ML.
- Automatiza desde o início de um projeto. Um excelente passo inicial de automatização é a integração contínua do teu projeto. Outra forma de dizer isto é que "se não for automatizado, está estragado".
- Pratica o Kaizen, ou seja, a melhoria contínua do teu pipeline. Este método melhora a qualidade do software, a qualidade dos dados, a qualidade do modelo e o feedback do cliente.
- Ao lidar com grandes equipas ou grandes dados, concentra-te na utilização de tecnologia de plataforma, como o AWS SageMaker, Databricks, Amazon EMR ou Azure ML Studio. Deixa a plataforma fazer o trabalho pesado para a tua equipa.
- Não te concentres apenas na complexidade das técnicas, ou seja, na aprendizagem profunda versus a resolução do problema com qualquer ferramenta que funcione.
- Leva a sério a governação dos dados e a cibersegurança. Uma forma de o fazer é utilizar o suporte empresarial para a sua

plataforma e realizar auditorias regulares à sua arquitetura e práticas.

Finalmente, há três leis da automação a considerar quando pensas nos MLOps:

1. Qualquer tarefa que fale em ser automatizada acabará por ser automatizada.
2. Se não for automatizado, está avariado.
3. Se um humano está a fazê-lo, uma máquina acabará por fazê-lo melhor.

Agora, vamos ver algumas dicas para lidar com as preocupações de segurança.

Governação de dados e cibersegurança

Existem dois problemas aparentemente contraditórios nos MLOps: o aumento das preocupações com a cibersegurança e a falta de aprendizagem automática na produção. Por um lado, demasiadas regras e nada é feito. Por outro lado, os ataques de resgate a infra-estruturas críticas estão a aumentar e seria sensato que as organizações prestassem atenção à forma como gerem os seus recursos de dados.

Uma forma de abordar ambas as questões em simultâneo é ter uma lista de verificação das melhores práticas. Segue-se uma lista parcial das melhores práticas para a governação de dados que irão melhorar a produtividade e a cibersegurança dos MLOps:

- Utiliza o PLP (Princípio do menor privilégio).
- Encripta os dados em repouso e em trânsito.
- Parte do princípio de que os sistemas que não são automatizados são inseguros.

- Utiliza plataformas Cloud, uma vez que têm um modelo de segurança partilhado.
- Utiliza o apoio da empresa e participa em auditorias trimestrais de arquitetura e segurança.
- Dá formação ao pessoal sobre as plataformas utilizadas, certificando-o.
- Envolve a empresa em acções de formação trimestrais e anuais sobre novas tecnologias e melhores práticas.
- Cria uma cultura empresarial saudável com padrões de excelência, funcionários competentes e uma liderança baseada em princípios.

Em seguida, vamos resumir alguns padrões de design MLOps que podem ser úteis para ti e para a tua organização.

Padrões de Design MLOps

Os exemplos seguintes reflectem uma lista parcial depadrões de conceção MLOps recomendados:

CaaS

O contêiner como um serviço (CaaS) é um padrão útil para MLOps porque permite que os desenvolvedores trabalhem em um microsserviço de ML em seu desktop ou em um editor Cloud e, em seguida, compartilhe-o com outros desenvolvedores ou com o público por meio de um comando `docker pull`. Além disso, muitas plataformas Cloud oferecem soluções PaaS (plataforma como serviço) de alto nível para implantar projetos em contêineres.

Plataforma MLOps

Todos os fornecedores de Cloud têm plataformas MLOps profundamente integradas. AWS tem o AWS SageMaker, o Azure tem o Azure ML Studio e o Google tem o vértice AI. Nos casos em que uma grande equipa, um grande projeto, grandes volumes de dados ou todos

os elementos acima referidos entram em jogo, a utilização da plataforma MLOps na Cloud que estás a utilizar poupar-te-á imenso tempo na construção, implementação e manutenção da tua aplicação de ML.

Sem servidor

A tecnologia sem servidor, como o AWS Lambda, é ideal para o rápido desenvolvimento de microsserviços de ML. Estes microsserviços podem chamar as APIs de IA da Cloud para realizar PNL, visão por computador ou outras tarefas ou utilizar um modelo pré-treinado desenvolvido por ti ou um que tenhas descarregado.

Centrado na Spark

Muitas organizações que lidam com Big Data já têm experiência com o Spark. Nessa situação, pode fazer sentido usar os recursos de MLOps da plataforma gerenciada do Spark Databricks ou do Spark gerenciado por meio da plataforma Cloud, como o AWS EMR.

Centrado em Kubernetes

Kubernetes é um "cloud in a box". Se a tua organização já está a utilizá-lo, pode fazer sentido utilizar a tecnologia focada em ML do Kubernetes, como o mlflow.

Para além destas recomendações, muitos recursos adicionais no [Anexo B](#) discutem tópicos que vão desde a governação de dados à certificação Cloud.

Conclusão

Este livro surgiu originalmente de uma discussão que tive com Tim O'Reilly e Mike Loukides no Foo Camp sobre como tornar o ML 10 vezes mais rápido. O consenso do grupo foi: sim, pode ser 10 vezes mais rápido! O livro de Matt Ridley "*How Innovation Works: And Why It Flourishes in Freedom* (Harper) ilustra que a resposta não intuitiva é que a inovação é

uma recombinação de ideias com uma melhor execução. Um colega atual e antigo professor, Andrew Hargadon, apresentou-me pela primeira vez estas ideias quando eu era estudante de MBA na UC Davis. No seu livro *How Breakthroughs Happen* (Harvard Business Review Press), Andrew refere que o que importa é o efeito de rede e a recombinação de ideias.

Para os MLOps, isto significa que a excelência operacional das ideias existentes é o molho secreto. As empresas que pretendem resolver problemas reais na aprendizagem automática e resolver esses problemas rapidamente podem inovar através da execução. O mundo precisa desta inovação para nos ajudar a salvar mais vidas através da medicina preventiva, como os exames de cancro automatizados de maior precisão, os carros autónomos e os sistemas de energia limpa que se adaptam ao ambiente.

Nada deve estar "fora de questão" para aumentar a excelência operacional. Se o AutoML acelera a prototipagem rápida, então utiliza-o. Se a computação em Cloud aumenta a velocidade de implementação de modelos de aprendizagem automática, então implementa-a. Como os acontecimentos recentes nos mostraram, com a pandemia da COVID-19 e as inovações tecnológicas resultantes, como a tecnologia CRISPR e a vacina contra a COVID-19, podemos fazer coisas incríveis com o devido sentido de urgência. O MLOps acrescenta rigor a esta urgência e permite-nos ajudar a salvar o mundo, um modelo de ML de cada vez.

Exercícios

- Cria uma aplicação de aprendizagem automática o mais rapidamente possível, que seja continuamente treinada e continuamente implementada.
- Implementa um modelo de aprendizagem automática utilizando a pilha Kubernetes.
- Implementa o mesmo modelo de aprendizagem automática utilizando a entrega contínua com AWS, Azure e GCP.

- Cria uma verificação de segurança automatizada dos contentores para um projeto de aprendizagem automática utilizando um sistema de construção nativo da Cloud.
- Treina um modelo utilizando um sistema AutoML baseado na Cloud e utilizando um sistema AutoML local como o Create ML ou o Ludwig.

Questões para discussão sobre pensamento crítico

- Como poderias construir um motor de recomendação que não tivesse tantas externalidades negativas como os actuais motores de recomendação das redes sociais? O que mudarias e como?
- O que pode ser feito para melhorar a precisão e a interpretabilidade da modelação de sistemas complexos como a nutrição, o clima e as eleições?
- Como é que a excelência operacional pode ser o ingrediente secreto para uma empresa que pretende ser líder em tecnologia relacionada com a aprendizagem automática?
- Se a excelência operacional é uma consideração crucial para os MLOps, quais são os critérios de contratação da tua organização para identificar o talento certo?
- Explica o papel da excelência operacional na aprendizagem automática no que diz respeito ao apoio empresarial à Cloud? É importante e porquê?

Apêndice A. Termos-chave

Por Noah Gift

Esta secção contém termos-chave selecionados que surgem frequentemente no ensino da computação em Cloud, MLOps e engenharia de aprendizagem automática:

Alertas

Os alertas são métricas de saúde que têm acções associadas. Um exemplo seria um alerta que envia uma mensagem de texto a um engenheiro de software quando um serviço Web devolve vários códigos de estado de erro.

Amazon ECR

O Amazon ECR é um registo de contentores que armazena contentores no formato Docker.

Amazon EKS

O Amazon EKS é um serviço Kubernetes gerido criado pela Amazon.

Escala automática

O escalonamento automático é o processo de aumentar ou diminuir a carga automaticamente com base na quantidade de recursos que os nós estão a utilizar.

AWS Cloud9

O AWS Cloud9 é um ambiente de desenvolvimento baseado na nuvem, executado na AWS. Tem ganchos especiais para desenvolver aplicações sem servidor.

AWS Lambda

Uma plataforma de computação sem servidor da AWS que tem capacidade FaaS.

Instâncias de contentores do Azure (ACI)

O Azure Container Instances é um serviço gerido da Microsoft que te permite executar imagens de contentores sem gerir servidores para os alojar.

Serviço de Kubernetes do Azure (AKS)

O Azure Kubernetes Service é um serviço Kubernetes gerido criado pela Microsoft.

black

A ferramenta **black** formata automaticamente o texto do código fonte Python.

Constrói um servidor

Um servidor de compilação é uma aplicação que funciona tanto no teste como na implementação de software. Os servidores de compilação populares podem ser SaaS ou de código aberto. Aqui estão algumas opções populares:

- **Jenkins** é um servidor de compilação de código aberto que pode ser executado em qualquer lugar, incluindo AWS, GCP, Azure, ou um contentor Docker ou no teu portátil.
- **CircleCI** é um serviço de construção SaaS que se integra com um popular fornecedor de alojamento Git como o GitHub.

CírculoCI

Um sistema de construção SaaS (software como um serviço) popular utilizado nos fluxos de trabalho DevOps.

Aplicações nativas da Cloud

As aplicações nativas da Cloud são serviços que utilizam as capacidades únicas da Cloud, como o serverless.

Contentor

Um contentor é um conjunto de processos que estão isolados do resto do sistema operativo. Eles geralmente têm o tamanho de megabytes.

Entrega contínua

A entrega contínua é o processo de entrega automática de software testado em qualquer ambiente.

Integração contínua

A integração contínua é o processo de testar automaticamente o software após o check-in no sistema de controlo da fonte.

Engenharia de dados

A engenharia de dados é o processo de automatização do fluxo de dados.

Recuperação de desastres

A recuperação de desastres é o processo de conceção de um sistema de software para recuperar apesar de um desastre. Este processo pode incluir o arquivamento de dados noutro local.

Contentor em formato Docker

Existem vários formatos para contentores. Uma forma emergente é o Docker, que envolve a definição de um *Dockerfile*.

Docker

A Docker é uma empresa que cria tecnologia de contentores, incluindo um motor de execução, uma plataforma de colaboração através do DockerHub e um formato de contentor chamado *Dockerfile*.

FaaS (função como um serviço)

Um tipo de computação em Cloud que facilita funções que respondem a eventos.

Google GKE

O Google GKE é um serviço Kubernetes gerido criado pela Google.

IPython

O interpretador `ipython` é um terminal interativo para Python. É o núcleo do notebook Jupyter.

JSON

JSON significa JavaScript Object Notation e é um formato de dados leve e legível por humanos, muito utilizado em serviços Web.

Clusters Kubernetes

Um cluster Kubernetes é uma implantação do Kubernetes que contém todo um ecossistema de componentes do Kubernetes, incluindo nós, pods, a API e contêineres.

Contentores Kubernetes

Um contentor Kubernetes é uma imagem Docker que é implementada num cluster Kubernetes.

Pods de Kubernetes

Um pod Kubernetes é um grupo de um ou mais contentores.

Kubernetes

Kubernetes é um sistema de código aberto para automatizar as operações de aplicações em contentores. O Google criou-o e abriu-o em 2014.

Teste de carga

O teste de carga é o processo de verificação das características de escala de um sistema de software.

Gafanhoto

Locust é uma estrutura de teste de carga que aceita cenários de teste de carga formatados em Python.

Registo

O registo é um processo de criação de mensagens sobre o estado de execução de uma aplicação de software.

Makefile

Um **Makefile** é um ficheiro que contém um conjunto de diretivas utilizadas para construir software. A maioria dos sistemas operativos Unix e Linux tem suporte incorporado para este formato de ficheiro.

Métricas

As métricas são a criação de KPIs (Key Performance Indicators) para uma aplicação de software. Um exemplo de um parâmetro é a percentagem de CPU utilizada por um servidor.

Microsserviço

Um microsserviço é um serviço leve e pouco acoplado. Pode ser tão pequeno como uma função.

Migra

Migrar é a capacidade de mover uma aplicação de um ambiente para outro.

Lei de Moore

A percepção de que, durante algum tempo, o número de transístores num microchip duplicou de dois em dois anos.

Operacionalização

O processo de preparar uma aplicação para a implementação em produção. Estas acções podem incluir monitorização, testes de carga e definição de alertas.

canalização

A ferramenta `pip` instala pacotes Python.

Portos

Uma porta é um ponto final de comunicação em rede. Um exemplo de uma porta é um serviço Web executado na porta 80 através do protocolo HTTP.

Prometeu

O Prometheus é um sistema de monitorização de código aberto com uma base de dados de séries temporais eficiente.

pylint

A ferramenta `pylint` verifica o código fonte Python em busca de erros de sintaxe.

PyPI

O Índice de Pacotes Python, onde os pacotes publicados estão disponíveis para instalação com ferramentas como `pip`.

pytest

A ferramenta `pytest` é uma estrutura para executares testes no código fonte Python.

Ambiente virtual Python

Um ambiente virtual Python é criado ao isolar um interpretador Python para um diretório e instalar pacotes nesse diretório. O interpretador Python pode executar esta ação através de `python -m venv yournewenv`.

Sem servidor

Serverless é uma técnica de construção de aplicações baseada em funções e eventos.

Fila de espera SQS

Uma fila de mensagens distribuída criada pela Amazon com leituras e gravações quase infinitas.

Swagger

Uma ferramenta swagger é uma estrutura de código aberto que simplifica a criação de documentação de API.

Máquina virtual

Uma máquina virtual é a emulação de um sistema operativo físico. Pode ter um tamanho de gigabytes.

YAML

O YAML é um formato de serialização legível por humanos frequentemente utilizado em sistemas de configuração. É facilmente transportável para o formato JSON.

Apêndice B. Certificações tecnológicas

Por Noah Gift

O termo "MLOps" implica diretamente uma ligação firme com TI, Operações e outras disciplinas tecnológicas tradicionais através da frase "Ops". Historicamente, a certificação tecnológica tem desempenhado um papel essencial na validação das competências dos profissionais do sector. O salário dos profissionais certificados é impressionante. **De acordo com o Zip Recruiter**, em fevereiro de 2021, o salário médio de um Arquiteto de Soluções AWS era de \$155k.

Uma forma de pensar sobre este tema é considerar a ideia de uma "Ameaça Tripla". No basquetebol, isto significa que um jogador é tão completo que consegue mais de 10 ressaltos, mais de 10 assistências e mais de 10 pontos num jogo de basquetebol. Podes aplicar esta mesma abordagem a uma carreira MLOps. Tem um portefólio de exemplos do teu trabalho, obtém uma certificação e tem experiência profissional ou uma licenciatura relevante.

Certificações AWS

Vamos abordar algumas opções de certificação da AWS.

Profissional de Cloud da AWS e Arquiteto de Soluções da AWS

Recomendo-te as certificações para especialistas em MLOps na Cloud da AWS. O AWS Cloud Practitioner é uma introdução mais suave ao mundo da certificação AWS, semelhante à certificação AWS Solutions Architect. Já ensinei muitas vezes esta certificação a alunos de muitos tipos diferentes:

Alunos de mestrado em ciência de dados, profissionais de negócios não técnicos e profissionais de TI atuais. Aqui estão algumas das perguntas mais comuns e as respostas relacionadas a ambas as certificações, com um olhar especial para uma pessoa com experiência em aprendizado de máquina que deseja passar no exame. Mesmo que não obtenhas a certificação AWS, estas perguntas são essenciais para o profissional de MLOps e vale a pena testar o teu nível de conhecimento.

P: Estou a ter problemas em ligar o RDS e partilhar ligações com um grupo de pessoas. Existe um método mais simples?

R: Podes achar mais simples utilizar o AWS Cloud9 como um ambiente de desenvolvimento para te ligares ao RDS. Podes ver um [passo a passo na Amazon](#).

P: O modelo de serviços Cloud é confuso. O que é PaaS e em que é que difere de outros modelos?

R: Uma forma de pensar nas ofertas Cloud é compará-las com a indústria alimentar. Podes comprar alimentos a granel numa loja como [a Costco](#). Tem uma escala considerável e pode passar os descontos no preço de compra para o cliente. No entanto, como cliente, também podes ter de levar a comida para casa, prepará-la e cozinhá-la. Esta situação é semelhante à da IaaS.

Agora olha para um serviço como o [Grubhub](#) ou [o Uber Eats](#). Não só não tens de conduzir até à loja para ir buscar a comida, como ela foi preparada, cozinhada e entregue para ti. Esta situação é semelhante à PaaS. Tudo o que tens de fazer é comer a comida.

Se olhares para a PaaS (plataforma como um serviço), o que significa que, enquanto programador, podes concentrar-te na lógica empresarial. Como resultado, muitas das complexidades da engenharia de software desaparecem. Um excelente exemplo de dois dos primeiros serviços PaaS é o [Heroku](#) e o [Google App Engine](#). Um exemplo perfeito de uma PaaS na AWS é o [AWS SageMaker](#). Resolve muitos dos problemas de infraestrutura envolvidos na criação e implementação da aprendizagem

automática, incluindo a formação distribuída e o fornecimento de previsões.

P: Qual é a definição exacta de uma localização de extremidade? Não é evidente.

R: Um **local de borda do AWS** é um local físico no mundo onde um servidor reside. As localizações periféricas são diferentes dos centros de dados porque têm um objetivo mais restrito. Quanto mais próximo o utilizador do conteúdo estiver da localização física do servidor, menor será a latência do pedido. Esta situação é crítica na entrega de conteúdos como o streaming de vídeos e música e também para jogar jogos. O serviço de borda mais comumente referido na AWS é o CloudFront. O CloudFront é um CDN (Content Delivery Network). As cópias em cache ou cópias do mesmo ficheiro de filme vivem nestes locais em todo o mundo através da CDN. Esta situação permite que todos os utilizadores tenham uma excelente experiência de transmissão deste conteúdo.

Outros serviços que utilizam localizações de borda incluem o **Amazon Route 53**, o **AWS Shield**, o **AWS Web Application Firewall** e o **Lambda@Edge**.

P: E se um dos centros de dados de uma zona de disponibilidade (AZ) for afetado por um incêndio? Como é que os centros de dados estão relacionados uns com os outros em termos de um desastre natural ou causado pelo homem? Como um sistema deve ser arquitetado para replicação de dados?

R: Como parte do **modelo de segurança partilhada**, a Amazon é responsável pela Cloud e o cliente é responsável pelo que está na Cloud. Esta situação significa que os dados estão seguros contra falhas catastróficas não planeadas, como incêndios. Além disso, se houver uma interrupção, os dados podem ficar indisponíveis durante a interrupção nessa região, mas acabam por ser restaurados.

Como arquiteto, é da responsabilidade do cliente tirar partido das arquitecturas multi-AZ. Um excelente exemplo disto é a **configuração multi-AZ do Amazon RDS**. Se ocorrer uma falha numa região, a base de dados secundária de ativação pós-falha já terá os dados replicados e tratará do pedido.

P: O que é o HA?

R: Um serviço HA (Highly Available) é construído com a disponibilidade em mente. Esta situação significa que a falha é uma expectativa, e o design suporta a redundância de dados e serviços. Um excelente exemplo de um serviço HA é o **Amazon RDS**. Além disso, o design do RDS Multi-AZ suporta uma interrupção mínima no serviço, permitindo várias versões da replicação da base de dados entre zonas de disponibilidade.

P: Como é que decides entre o spot e o on-demand?

R: Tanto as instâncias à vista como a pedido são facturadas com um montante fixo para o primeiro minuto e, em seguida, facturadas ao segundo. As instâncias Spot são as mais económicas porque podem proporcionar uma poupança de até 90%. Utiliza **as instâncias Spot** quando não importa quando uma tarefa é executada ou interrompida. Na prática, isto cria um caso de utilização crítico para uma instância pontual. Eis alguns exemplos:

- Experimenta um serviço AWS
- Treina trabalhos de aprendizagem profunda ou de aprendizagem automática
- Dimensiona um serviço Web ou outro serviço em combinação com instâncias a pedido

As instâncias on-demand funcionam quando uma carga de trabalho é executada num estado estável. Assim, por exemplo, um serviço Web em produção não quereria utilizar apenas instâncias pontuais. Em vez disso,

poderia começar com casos a pedido e, quando a utilização do serviço for calculada (ou seja, 2 instâncias c4.large), então **as instâncias reservadas** devem ser compradas.

P: Como é que a hibernação pontual funciona?

R: Existem alguns motivos para as **interrupções de instâncias spot**, incluindo preço (oferta superior ao preço máximo), capacidade (não existem instâncias spot suficientes não utilizadas) e restrições (ou seja, o tamanho alvo da Zona de disponibilidade é demasiado grande). Para hibernar, tem de ter um volume de raiz EBS.

P: Para que serve uma etiqueta no EC2?

R: Na **Figura B-1**, uma instância EC2 tem uma etiqueta aplicada a ela. Esta etiqueta pode agrupar tipos de instâncias num grupo lógico como "servidores Web".

The screenshot shows the 'Step 5: Add Tags' section of the AWS EC2 instance creation wizard. At the top, a navigation bar lists steps 1 through 7: Choose AMI, Choose Instance Type, Configure Instance, Add Storage, Add Tags (which is highlighted in blue), Configure Security Group, and Review. Below the navigation bar, the title 'Step 5: Add Tags' is displayed. A descriptive text explains that a tag consists of a key-value pair and can be applied to instances or volumes. It also mentions that tags will be applied to all instances and volumes. A 'Learn more' link is provided for tagging resources. The main interface shows a table where a single tag is being added. The 'Key' column is labeled 'Name' and contains 'ec20demo'. The 'Value' column is empty. To the right of the table are two buttons: 'Instances' and 'Volumes', each with a small icon. At the bottom of the form, there is a button labeled 'Add another tag' and a note '(Up to 50 tags maximum)'. At the very bottom of the screenshot, there is a footer with buttons for 'Cancel', 'Previous', 'Review and Launch' (which is highlighted in blue), and 'Next: Configure Security Group'.

Figura B-1. Etiquetas EC2

A principal razão para utilizar uma etiqueta para um recurso EC2 é anexar metadados a um grupo de máquinas. Aqui está um cenário. Digamos que 25 instâncias do EC2 estão a executar um site de compras e não têm etiquetas. Mais tarde, um utilizador cria outras 25 instâncias

EC2 para executar temporariamente uma tarefa como o treino de um modelo de aprendizagem automática. Pode ser um desafio na consola determinar que máquinas são temporárias (e podem ser eliminadas) e as máquinas de produção.

Em vez de adivinhar as funções das máquinas, é melhor atribuir etiquetas que permitam a um utilizador identificar rapidamente uma função. Esta função pode ser: Key="role", Value="ml" ou pode ser Key="role", Value="web". Na consola do EC2, um utilizador pode consultar por etiqueta. Esse processo permite operações em massa, como o encerramento de instâncias. As tags também podem desempenhar um papel importante na análise do custo. Se as funções de máquina contiverem marcas, os relatórios de custo poderão determinar se determinados tipos de máquina são muito caros ou usam muitos recursos. Podes ler a [documentação oficial das etiquetas na Amazon](#).

P: O que é o PuTTY?

R: Na Figura B-2, a ferramenta SSH PuTTY permite o acesso remoto à consola a partir do Windows para uma máquina virtual Linux.

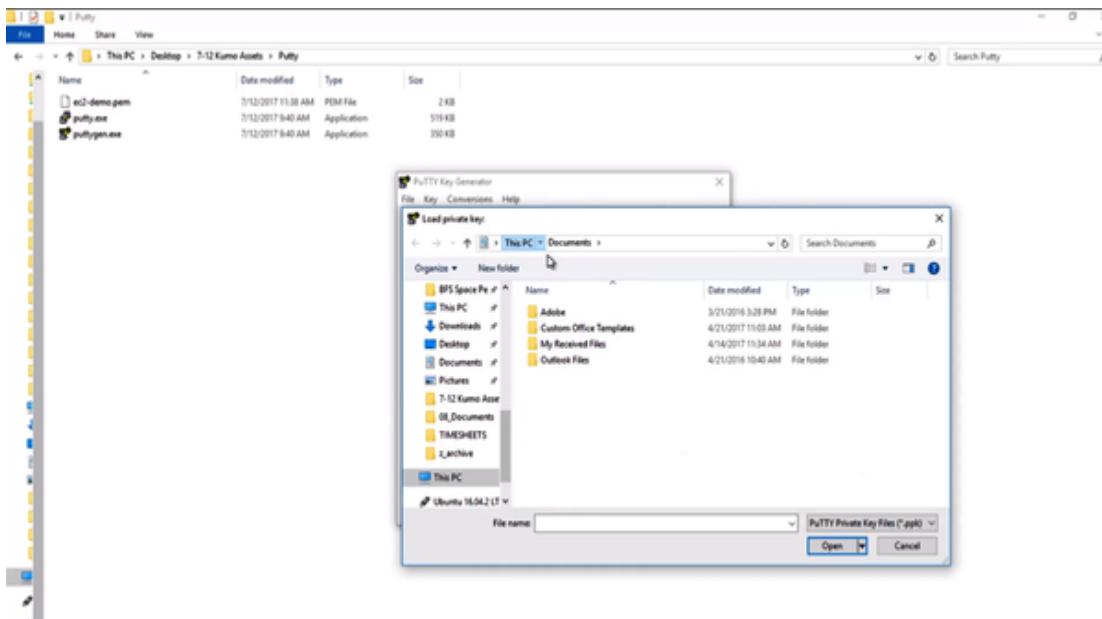


Figura B-2. PuTTY

O PuTTY é um cliente SSH gratuito utilizado no sistema operativo Windows. O MacOS e o Linux têm suporte incorporado para SSH. O que é o SSH? É um protocolo de rede criptográfico para realizar operações de rede. O SSH serve para iniciar sessão numa máquina remota e gerir o dispositivo através de comandos de terminal.

Podes ler a [documentação oficial do PuTTY aqui](#).

P: Em que é que o Lightsail é diferente do EC2 ou de outros serviços?

R: O Lightsail é uma PaaS ou plataforma como um serviço. Esta plataforma significa que um programador só tem de se preocupar com a configuração e o desenvolvimento da aplicação WordPress. O EC2 é de nível inferior e é chamado de IaaS (infraestrutura como serviço). Existe um espelho na computação em nuvem em que os serviços de nível inferior são fornecidos, tal como os ingredientes a granel na Costco. Esses ingredientes a granel dão origem a refeições, mas requerem competências. Da mesma forma, uma pessoa pode encomendar refeições para serem entregues em sua casa. Estas refeições são mais caras, mas requerem pouca experiência. A PaaS é semelhante; o utilizador paga mais por serviços de nível superior.

Outras soluções PaaS (fora da AWS) são o [Heroku](#) e o [Google App Engine](#). Podes ler mais sobre os tipos de serviços Cloud no [capítulo "Cloud Computing"](#) do livro *Python for DevOps* (O'Reilly).

P: Li sobre um caso de uso para a AMI que tem a seguinte descrição: "usa-o para copiar isso para uma frota de máquinas (cluster de aprendizagem profunda)." O que significa uma frota de máquinas?

R: Uma frota funciona da mesma forma que uma empresa de aluguer de automóveis. Quando pedes uma reserva de carro, pedem-te para escolheres um grupo: compacto, sedan, luxo ou camião. Não te é garantido um modelo específico, apenas um determinado grupo. Da mesma forma, como as instâncias pontuais são um mercado aberto, é possível que uma determinada máquina, por exemplo C3.8XLarge, não esteja disponível, mas uma combinação semelhante esteja. Podes

solicitar um grupo de recursos idênticos em termos de CPU, memória e capacidades de rede, selecionando uma frota. Podes ler mais sobre o [EC2 Fleet no blogue da Amazon](#).

P: O que significa spike para o dimensionamento de instâncias on-demand?

R: Uma carga de trabalho "espinhosa" pode ser um site que, de repente, recebe 10 vezes mais tráfego. Digamos que este sítio Web vende produtos. Normalmente, tem uma quantidade de tráfego estável durante o ano, mas por volta de dezembro, o tráfego aumenta para 10 vezes o tráfego habitual. Este cenário seria um caso de utilização justo para que as instâncias "a pedido" fossem escaladas para satisfazer este requisito. O padrão de tráfego esperado deve utilizar instâncias reservadas, mas para o pico, deve utilizar instâncias a pedido. Podes ler mais sobre o [tráfego em picos e as instâncias reservadas no blogue da Amazon](#).

P: O que significa o "SUBSECOND" para uma das vantagens do AWS Lambda?

R: Isto significa que pode conceber um serviço eficiente e apenas incorrer em encargos pela duração do seu pedido em intervalos de 100 ms. Esta situação é diferente de uma instância EC2 em que és cobrado por uma instância em execução contínua a cada segundo. Com uma função Lambda, podes conceber um fluxo de trabalho baseado em eventos, em que uma lambda é executada apenas em resposta a eventos. Uma boa analogia seria uma luz tradicional que se apaga e acende. É fácil usar mais eletricidade porque a luz tem um interruptor manual. Uma abordagem mais eficiente é a iluminação com deteção de movimento. A iluminação desliga-se e liga-se de acordo com o movimento. Esta abordagem é semelhante ao AWS Lambda; em resposta a eventos, liga, executa uma tarefa e depois sai. Podes ler mais sobre o [Lambda na documentação da Amazon](#). Também podes construir um [projeto Python AWS Lambda no GitHub](#).

P: No AWS S3, há várias classes de armazenamento. O nível de armazenamento IA (acesso infrequente) inclui o IA padrão e o IA de uma zona, ou há mais tipos? Vejo apenas standard e one-zone na secção de ACESSO INFREQUENTE no [site da AWS](#).

R: Existem dois tipos de IA (acesso infrequente). O IA padrão, que armazena em três AZ (Zonas de Disponibilidade) e o One Zone. Uma das principais diferenças da One Zone é a disponibilidade. Tem 99,5% de disponibilidade, menos disponível do que a AI de três zonas e a Standard. O custo mais baixo reflecte esta menor disponibilidade.

P: Como funciona o Elastic File System (EFS)?

R: O EFS funciona conceitualmente de forma semelhante ao Google Drive ou ao Dropbox. Podes criar uma conta Dropbox e partilhar dados com vários computadores ou amigos. O EFS funciona de forma muito semelhante. O mesmo sistema de ficheiros está disponível para as máquinas que o montam. Este processo é muito diferente do EBS (Elastic Block Storage), que pertence a uma instância de cada vez.

P: Relativamente ao caso de utilização do ELB, não comprehendo estes dois casos de utilização: 1) O que significa "ponto de acesso único"? Está a dizer que, se puderdes controlar o teu tráfego entrando através de uma porta ou servidor, então é mais seguro? 2) o que significa "desacoplar o ambiente da aplicação"?

R: Vejamos um sítio Web como exemplo. O site será executado na porta 443, que é a porta para tráfego HTTPS. Este site pode ser <https://example.com>. O ELB é o único recurso exposto ao mundo exterior. Quando um navegador da Web se conecta a <https://example.com>, ele se comunica apenas com o ELB. Ao mesmo tempo, o ELB solicita as informações aos servidores Web por trás dele. Em seguida, envia essas informações de volta para o navegador da Web.

O que é uma analogia no mundo real? Seria como um caixa de banco num drive-through. Diriges até à janela mas só te ligas à caixa do banco.

Dentro do banco, muitas pessoas estão a trabalhar, mas tu estás a interagir apenas com uma pessoa. Podes ler publicações sobre ELB no [blogue da AWS](#).

P: Porque é que o caso de utilização do ELB é o mesmo que o do Classic Load Balancer?

R: Partilham as mesmas características: acesso através de um único ponto de entrada, ambiente de aplicação dissociado; proporcionam uma elevada disponibilidade e tolerância a falhas e aumentam a elasticidade e a escalabilidade.

Elastic Load Balancing refere-se a uma categoria de平衡adores de carga. Existe o Application Load Balancer, o Network Load Balancer e o Classic Load Balancer. A um nível elevado, o Classic Load Balancer é um balanceador de carga mais antigo que tem menos funcionalidades do que o Application Load Balancer. Funciona em situações em que as instâncias EC2 mais antigas estão em serviço.

Estas são as chamadas instâncias clássicas do EC2. Num novo cenário, um Application Load Balancer seria ideal para algo como um serviço HTTP. Podes ler publicações de blogue sobre [comparações de funcionalidades ELB na documentação da Amazon](#).

AWS Certified Machine Learning - Especialidade

As escolas de gestão e as escolas de ciência de dados adoptaram completamente o ensino de certificações. Na UC Davis, ensino aos alunos material tradicional para crédito, como aprendizagem automática e certificações Cloud, incluindo o AWS Cloud Practitioner e o AWS Certified Machine Learning - Specialty. Com a AWS, trabalho em estreita colaboração com muitas partes da sua organização, incluindo a AWS Educate, a AWS Academy e a [AWS ML Hero](#).

Como deves calcular, sim, recomendo a obtenção da certificação AWS Machine Learning. Conhecendo muitas das pessoas envolvidas na criação da certificação ML da AWS, e talvez com alguma influência na sua criação,

o que mais me agrada é o grande enfoque em MLOps. Por isso, vamos aprofundar o assunto.

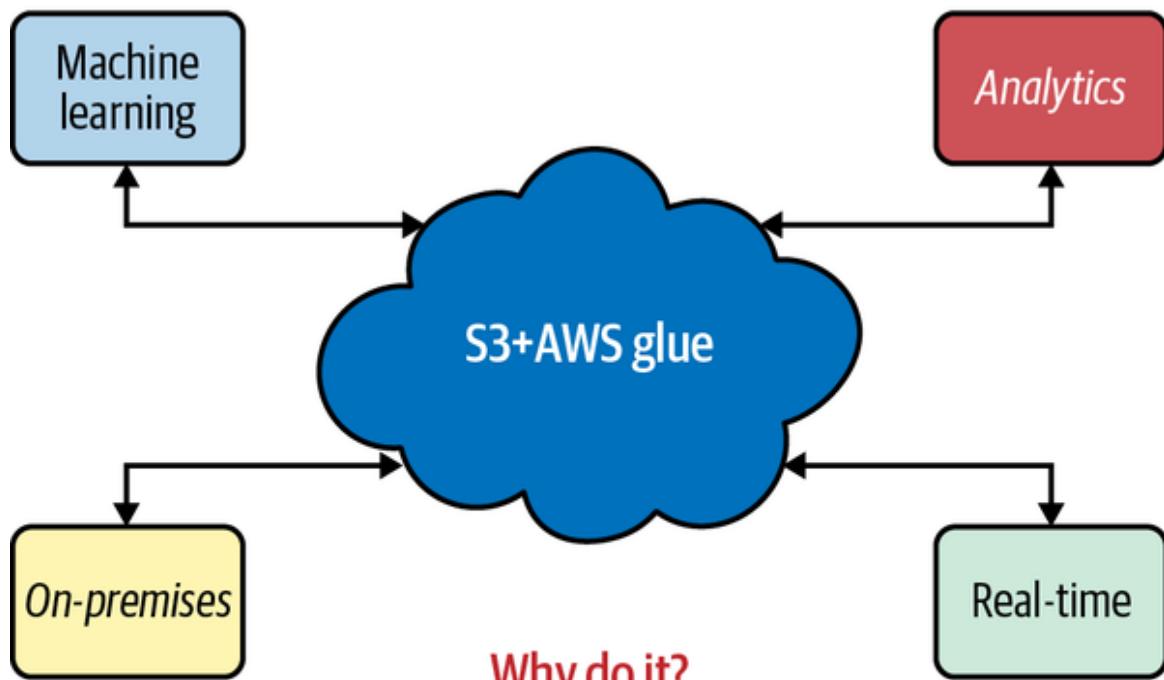
O **candidato recomendado** tem conhecimentos e experiência de, pelo menos, 1-2 anos a desenvolver, arquitetar ou executar cargas de trabalho de ML/aprendizagem profunda. Na prática, isso significa a capacidade de expressar a intuição por trás dos algoritmos básicos de ML, realizar a otimização básica de hiperparâmetros, experiência com frameworks de ML e deep learning, seguir as práticas recomendadas de treinamento de modelos e seguir as práticas recomendadas operacionais e de implantação. Em suma, a leitura deste livro é um excelente passo de preparação para a obtenção da certificação!

A estrutura do exame divide-se em vários domínios: Engenharia de dados, análise exploratória de dados, modelação e implementação e operações de aprendizagem automática. A última secção do exame, em particular, é sobre o que é essencialmente MLOps. Por isso, vamos analisar estas secções e ver como se aplicam aos conceitos abordados neste livro.

Engenharia de dados

Um componente central da engenharia de dados, análise e aprendizado de máquina na AWS é o lago de dados, que também é o Amazon S3. Por que usar um lago de dados(**Figura B-3**)? Os principais motivos são os seguintes. Primeiro, fornece a capacidade de lidar com dados estruturados e não estruturados. Um lago de dados também permite cargas de trabalho analíticas e de ML. Em terceiro lugar, podes trabalhar com os dados sem os mover, o que pode ser crítico com os grandes volumes de dados. E, finalmente, o seu baixo custo.

AWS data lake



- Structured and unstructured data
- Analytics and ML
- Work on data without data movement
- Low cost storage

Figura B-3. Lago de dados

Outro tópico importante sobre a engenharia de dados da AWS é o lote versus dados de fluxo contínuo. Primeiro, vamos definir dados de fluxo contínuo. Os dados de fluxo contínuo são normalmente pequenos dados enviados de várias fontes. Exemplos incluem arquivos de log, métricas e dados de séries temporais, como informações de negociação de ações. O principal serviço da AWS que lida com streaming é o Kinesis. Eis alguns cenários ideais: problemas de análise de séries temporais, dashboards em tempo real e métricas em tempo real.

A comparação entre lote e streaming tem efeitos significativos no desenvolvimento do pipeline de ML. Existe um maior controlo da formação do modelo em lote, uma vez que é possível decidir quando voltar a treinar o modelo de ML. O retreinamento contínuo de um modelo pode fornecer

melhores resultados de previsão, mas com maior complexidade. Por exemplo, o teste A/B é usado para testar a precisão do novo modelo? O teste A/B do modelo está disponível nos pontos de extremidade do SageMaker, portanto, isso deverá ser levado em consideração na arquitetura.

Para o processamento em lote, várias ferramentas podem funcionar como mecanismos de processamento em lote. Entre elas estão o Amazon EMR/Spark, o AWS Glue, o AWS Athena, o AWS SageMaker e o **serviço AWS Batch**, apropriadamente chamado. Em particular, para a aprendizagem automática, o AWS Batch resolve um problema único. Por exemplo, imagina que queres escalar milhares de trabalhos de agrupamento k-means simultâneos e individuais. Uma forma de o fazeres seria com o AWS Batch. Além disso, podes fornecer uma interface simples para o sistema de processamento em lote, escrevendo uma ferramenta de linha de comandos Python. Aqui está um trecho de código que mostra como isso poderia ser na prática:

```
@cli.group()
def run():
    """AWS Batch CLI"""

    @run.command("submit")
    @click.option("--queue", default="queue", help="Batch Queue")
    @click.option("--jobname", default="1", help="Name of Job")
    @click.option("--jobdef", default="test", help="Job Definition")
    @click.option("--cmd", default=["whoami"], help="Container
Override Commands")
    def submit(queue, jobname, jobdef, cmd):
        """Submit a job to AWS Batch Service"""

        result = submit_job(
            job_name=jobname,
            job_queue=queue,
            job_definition=jobdef,
            command=cmd
        )
        click.echo(f"CLI: Run Job Called {jobname}")
    return result
```

Outro aspecto crítico do tratamento da engenharia de dados é a utilização do AWS Lambda para processar eventos. É importante observar que o AWS Lambda é uma cola que se integra profundamente à maioria dos serviços da AWS. Portanto, ao fazer engenharia de dados com base na AWS, provavelmente encontrará o AWS Lambda em algum momento.

O CTO da AWS acredita que "uma base de dados de tamanho único não serve a ninguém". O que ele quer dizer com isso está claramente descrito na Figura B-4.

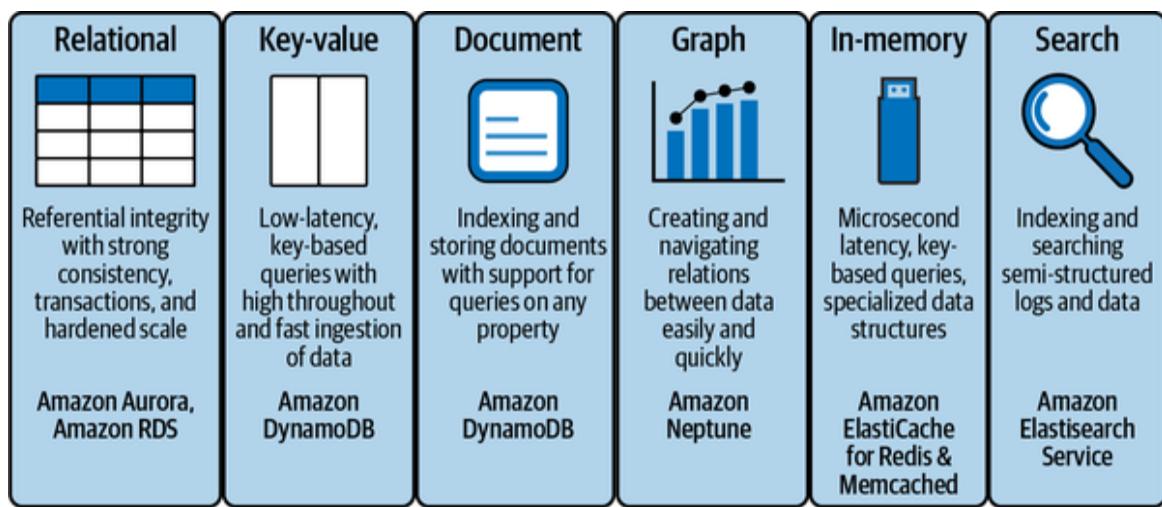


Figura B-4. Base de dados de tamanho único

Outra forma de dizer isto é utilizar a melhor ferramenta para o trabalho. Pode ser uma base de dados relacional; noutras casos, é uma base de dados de chave/valor. O exemplo a seguir mostra como funciona uma API simples baseada no DynamoDB em Python. A maior parte do código é de registo.

```
def query_police_department_record_by_guid(guid):
    """Gets one record in the PD table by guid

In [5]: rec = query_police_department_record_by_guid(
    "7e607b82-9e18-49dc-a9d7-e9628a9147ad"
)

In [7]: rec
Out[7]:
{'PoliceDepartmentName': 'Hollister',
 'UpdateTime': 'Fri Mar 2 12:43:43 2018',
 'guid': '7e607b82-9e18-49dc-a9d7-e9628a9147ad'}
```

```

"""
db = dynamodb_resource()
extra_msg = {"region_name": REGION, "aws_service": "dynamodb",
             "police_department_table": POLICE_DEPARTMENTS_TABLE,
             "guid": guid}
log.info(f"Get PD record by GUID", extra=extra_msg)
pd_table = db.Table(POLICE_DEPARTMENTS_TABLE)
response = pd_table.get_item(
    Key={
        'guid': guid
    }
)
return response['Item']

```

Três itens finais a serem discutidos na engenharia de dados são ETL, Segurança de dados e Backup e recuperação de dados. Com o ETL, os serviços essenciais incluem o AWS Glue, o Athena e o AWS DataBrew. O AWS DataBrew é o serviço mais recente e resolve uma etapa necessária na criação de modelos de aprendizado de máquina de produção, ou seja, automatiza as etapas confusas na limpeza de dados. Por exemplo, na [Figura B-5](#), um conjunto de dados, nomes de bebés, é perfilado sem escrever uma única linha de código.

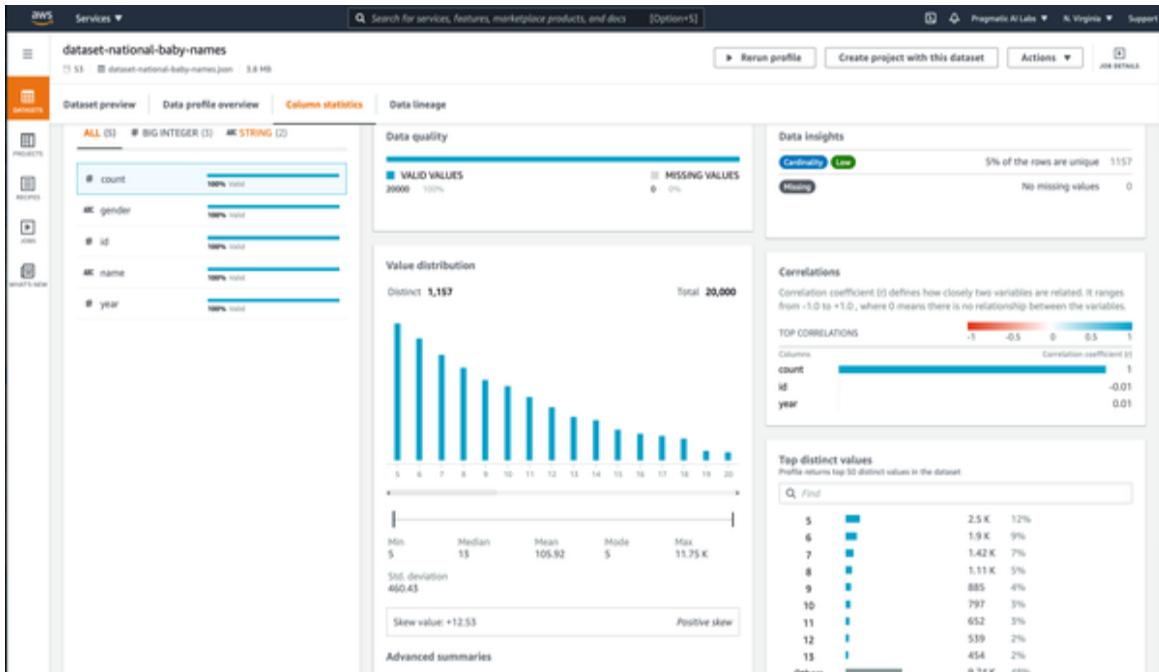


Figura B-5. DataBrew

Mais tarde, este mesmo conjunto de dados pode ser uma parte vital de um projeto MLOps. Uma funcionalidade útil é a capacidade de seguir a linhagem do conjunto de dados, a sua origem e as acções que envolvem o conjunto de dados. Esta funcionalidade está disponível através do separador "Data lineage" (Linhagem de dados) (ver [Figura B-6](#)).

A governação de dados é uma forma concisa de abordar as preocupações relacionadas com a segurança dos dados e com o backup e a recuperação de dados. AWS, através do KMS (Key Management Service), permite uma estratégia de encriptação integrada. Este passo é fundamental porque permite a implementação de encriptações em repouso e em trânsito, bem como de PLP (Princípio do privilégio mínimo).

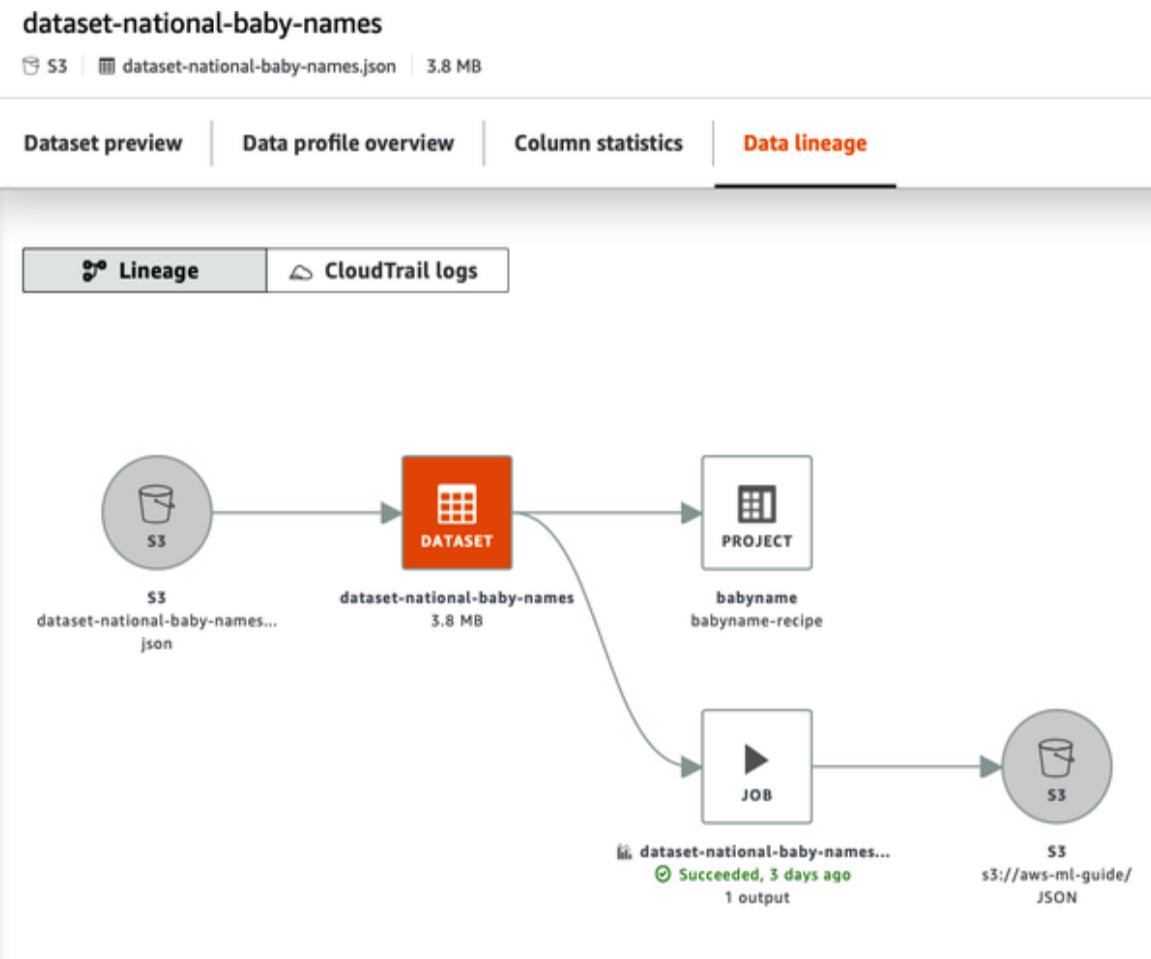


Figura B-6. Linhagem DataBrew

Outro aspecto da segurança dos dados é a capacidade de registar e auditar o acesso aos dados. A auditoria regular do acesso aos dados é uma forma de identificar riscos e mitigá-los. Por exemplo, podes assinalar um utilizador que olha regularmente para um AWS Bucket que não tem nada a ver com o seu trabalho e depois apercebes-te que isso representa uma falha de segurança significativa que tens de fechar.

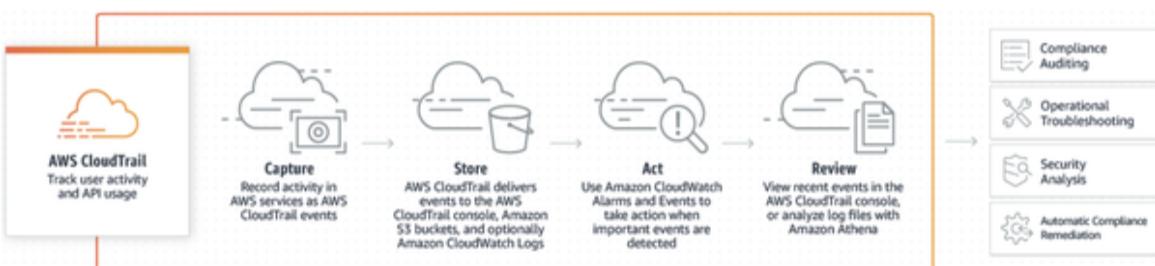


Figura B-7. AWS Cloud-Trail

Por fim, o backup e a recuperação de dados é talvez um dos aspectos mais importantes da governança de dados. A maioria dos serviços AWS tem capacidades de snapshot, incluindo RDS, S3 e DynamoDB. Conceder uma cópia de segurança, uma recuperação e um estilo de vida úteis para os dados que são arquivados no Amazon Glacier é essencial para a conformidade com as melhores práticas na engenharia de dados.

Análise Exploratória de Dados (EDA)

Antes de poderes fazer aprendizagem automática, primeiro, os dados precisam de ser explorados. Na AWS, várias ferramentas podem ajudar-te. Elas incluem o exemplo do DataBrew de antes, bem como o AWS QuickSight. Vamos ver o que podes fazer com o AWS QuickSight na Figura B-8.

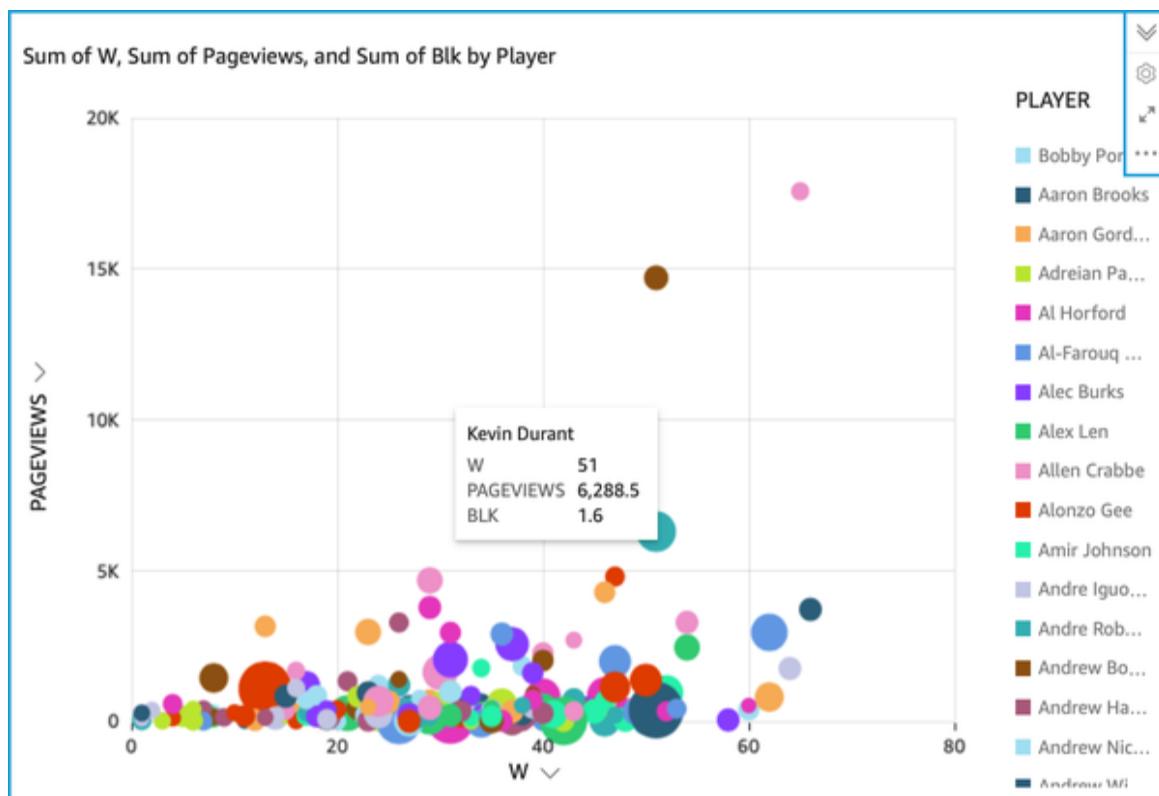


Figura B-8. AWS QuickSight

Repara como esta abordagem sem código/baixo código expõe uma relação de lei de potência entre vitórias e popularidade nas redes sociais, ou seja, visualizações de páginas da Wikipédia. Os fãs podem gravitar mais perto

dos "vencedores", prestar atenção neles e querer ler mais informações sobre esses jogadores. Este passo inicial da EDA pode levar imediatamente ao desenvolvimento de um modelo de aprendizagem automática que utilize o comportamento dos adeptos para prever quais as equipas com maior probabilidade de ganhar a época da NBA.

Replicar este gráfico é simples: transfere o **ficheiro CSV**, diz ao QuickSight para efetuar uma nova análise, cria um novo conjunto de dados utilizando o ficheiro CSV e, em seguida, seleciona "criar uma análise".

É essencial compreender o papel da EDA nos MLOps. A EDA ajuda a detetar valores atípicos, a encontrar padrões ocultos (através de agrupamento), a ver as distribuições de dados e a criar características. Ao utilizar o agrupamento, é essencial lembrar que os dados precisam de ser escalados. O escalonamento de dados normaliza a magnitude. Por exemplo, se dois amigos correram "50", é vital considerar a importância. Um amigo pode ter corrido 50 milhas, e outro pode ter corrido 50 pés. São coisas imensamente diferentes. Sem escalonamento, os resultados do aprendizado de máquina são distorcidos pela magnitude de uma variável ou coluna. O exemplo a seguir mostra como o escalonamento é visto na prática:

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

scaler = StandardScaler()

print(scaler.fit(numerical_StandardScaler(copy=True,
                                             with_mean=True, with_std=True))

# output
# [[ 2.15710914  0.13485945  1.6406603 -0.46346815]]
```

Outro conceito na EDA é a ideia de pré-processamento de dados. O pré-processamento é um termo lato que se pode aplicar a mais do que um cenário. Por exemplo, a aprendizagem automática exige que os dados sejam numéricos, pelo que uma forma de pré-processamento consiste em codificar variáveis categóricas para um formato numérico.

NOTA

A codificação de dados categóricos é uma parte essencial da aprendizagem automática. Existem muitos tipos diferentes de variáveis categóricas.

- Variáveis categóricas (discretas)
 - Conjunto finito de valores: {verde, vermelho, azul} ou {falso, verdadeiro}
- Tipos categóricos:
 - Ordinal (ordenado): {Large, Medium, Small}
 - Nominal (não ordenado): {verde, vermelho, azul}
- Representado como texto

Outra forma de pré-processamento é a criação de novas características. Vejamos este exemplo das idades dos jogadores da NBA. Um gráfico mostra que há uma distribuição normal da idade com a mediana em torno de 25 anos([Figura B-9](#)):

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv(
    r"https://raw.githubusercontent.com/noahgift/socialpowernba" \
    r"/master/data/nba_2017_players_with_salary_wiki_twitter.csv")

sns.distplot(df.AGE)
plt.legend()
plt.title("NBA Players Ages")
```

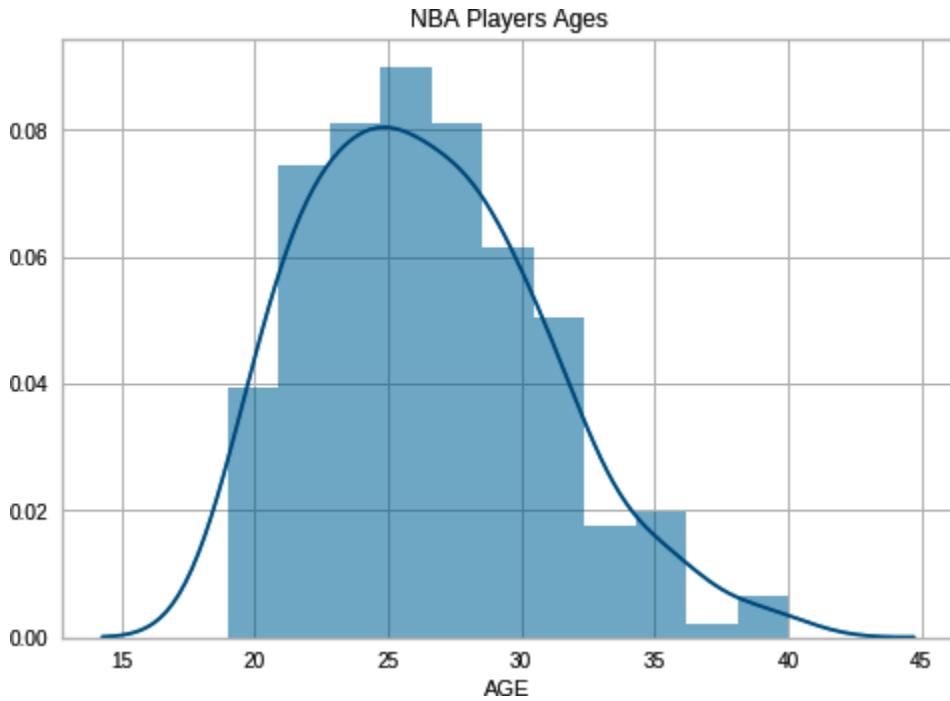


Figura B-9. Idade dos jogadores da NBA

Podemos utilizar este conhecimento para criar uma nova funcionalidade. A funcionalidade pode converter a idade em várias categorias: Novato, Primeiro, Pós-Primeiro e Pré-Reforma. Estas categorias podem conduzir a futuros conhecimentos:

```
def age_brackets (age):
    if age >17 and age <25:
        return 'Rookie'
    if age >25 and age <30:
        return 'Prime'
    if age >30 and age <35:
        return 'Post Prime'
    if age >35 and age <45:
        return 'Pre-Retirement'
```

Podemos então utilizar estas novas categorias para agrupar os dados e calcular o salário médio de cada escalão:

```
df["age_category"] = df["AGE"].apply(age_brackets)
df.groupby("age_category")["SALARY_MILLIONS"].media
```

Repara que há diferenças dramáticas no salário médio. Quando os jogadores começam a jogar, são os que recebem menos, mas o seu salário quase triplica no seu auge. Quando se reformam, há uma mudança dramática, pois o seu salário é metade do que recebiam no seu auge:

```
age_category
Post-Prime    8.550
Pre-Retirement 5.500
Prime          9.515
Rookie         2.940
Name: SALARY_MILLIONS, dtype: float64
```

Implementação e operações de aprendizagem automática (MLOps)

Vamos discutir alguns dos componentes críticos do MLOps no conceito do exame de certificação AWS ML. Os conceitos-chave a seguir são essenciais para considerar ao criar modelos:

- Monitorização
- Segurança
- Reciclagem de modelos
- Testes A/B
- TCO (Total Cost of Ownership)

O que achas do próprio MLOps? É fundamental ter em conta os seguintes aspectos:

- Estás a utilizar um modelo suficientemente simples?
- Estás a usar o lago de dados ou estás ligado diretamente ao SQL DB de produção?
- Tens alertas configurados para falhas nos limiares de previsão?
- Tens ambientes de desenvolvimento, fase e produção?

Por fim, vale a pena discutir dois tópicos: solução de problemas da implantação de produção e eficiência dos sistemas de ML. Para uma implantação de produção, esses conceitos incluem o uso do CloudWatch, a pesquisa nos logs do CloudWatch, o alerta sobre eventos críticos, o uso de recursos de dimensionamento automático e o uso do suporte empresarial.

Para o custo e a eficiência dos sistemas ML, tanto no exame como no mundo real, é essencial compreender os seguintes conceitos-chave:

- Instâncias Spot (mostra o código spot)
- Utilização correta dos recursos da CPU e da GPU
- Aumentar e diminuir a escala
- Tempo de colocação no mercado
- API de IA versus "Do it Yourself"

Outras certificações Cloud

Para além da AWS, a Azure e a GCP têm certificações notáveis.

Cientista de dados e engenheiro de IA do Azure

O Azure tem [algumas certificações](#) que vale a pena analisar, incluindo o [Azure Associate Data Scientist](#) e o [Azure AI Engineer Associate](#). Estas certificações estão divididas em três níveis (por ordem de especialização): fundamentos, associado e especialista. Além disso, existem vários *percursos de aprendizagem* relacionados com a plataforma de IA do Azure que estão intimamente relacionados com os MLOps:

- O guia de iniciação
- Cria modelos de aprendizagem automática
- Cria modelos preditivos sem código com o Azure Machine Learning

- Cria soluções de IA com o Azure Machine Learning

Um bom ponto de partida é consultar [o guia de formação](#), que tem uma lista convincente de jornadas (por exemplo, profissionais de dados e IA) e certificações explícitas como o [Azure AI Fundamentals](#). Estas *jornadas* permitem-te decidir a melhor estratégia para perseguir o objetivo que funciona melhor.

Além disso, existem recursos gratuitos (ou quase gratuitos) para te ajudar a começar a utilizar o Azure se fores [estudante](#) ou [membro do corpo docente](#). As ofertas tendem a mudar com o tempo, mas podes inscrever-te sem um cartão de crédito e começar imediatamente a utilizar o Azure a partir deste momento. Se fores um educador, existem [ofertas e recursos](#) adicionais disponíveis que também podem ser úteis.

GCP

Algumas certificações notáveis para profissionais de MLOps incluem o [Professional Machine Learning Engineer](#) e o [Professional Cloud Architect](#). Finalmente, uma certificação muito específica relacionada com MLOps é o [TensorFlow Developer Certificate](#). Permite-te demonstrar a tua proficiência na utilização do TensorFlow para resolver problemas de aprendizagem profunda e de ML.

Certificações relacionadas com SQL

Para ter sucesso com MLOps, é necessário um conhecimento mínimo de SQL. No entanto, mais tarde, recomenda-se que aprofunde os seus conhecimentos de SQL e que os domine de forma teórica e aplicada. Aqui tens alguns recursos recomendados:

- Certifica-te como Programador Associado da Databricks para o Apache Spark 3.0
 - Material de estudo: Site da Databricks e Plataforma de aprendizagem O'Reilly

- Plataforma de aprendizagem O'Reilly: Aprende a usar o **Spark**
 - Plataforma de aprendizagem O'Reilly: **Spark: O Guia Definitivo**
- Certificação Microsoft: Fundamentos de Dados do Azure
 - Material de estudo: Coursera, Microsoft Learn e O'Reilly Learning Platform
 - **Curso Coursera Preparação para o Exame Microsoft Azure DP-900 Data Fundamentals**
 - Plataforma de aprendizagem O'Reilly: **Exame Ref DP-900**
- Certificação Oracle Database SQL Certified Associate
 - Material de estudo: Plataforma de aprendizagem da O'Reilly e Oracle
 - Plataforma de aprendizagem O'Reilly: **Guia do exame OCA Oracle Database SQL**
 - Oracle: **Certificação Oracle Database SQL Certified Associate**
- Profissional de análise de dados do Google
 - Material de estudo: Coursera e O'Reilly Learning Platform
 - Plataforma de aprendizagem O'Reilly: **Governação de dados: O Guia Definitivo**
 - Plataforma de aprendizagem O'Reilly: **Ciência de dados na Google Cloud Platform**

- Plataforma de aprendizagem O'Reilly:[Google BigQuery: O guia definitivo](#)
- Coursera:[Profissional de análise de dados do Google](#)

Podes encontrar mais referências no[Coursera](#).

Apêndice C. Trabalho à distância

Por Noah Gift

No mundo pós-COVID-19, é fundamental ter um escritório estável em casa onde possas trabalhar. Outro fator é que o remote-first optimiza os resultados. Um problema significativo com os ambientes presenciais é a "aparência" de progresso versus o progresso real. Ser arrastado para reuniões durante horas que não têm qualquer resultado é um bom exemplo. Ter a equipa de "vendas" a interromper os programadores que estão a escrever código num escritório aberto é outro exemplo. Quando o foco está estritamente nos resultados, então o remote-first começa a fazer muito sentido.

Nos últimos anos, tenho estado continuamente a "hackear" o meu escritório em casa para poder ensinar em todo o mundo e nas principais universidades de topo, bem como fazer engenharia de software e consultoria à distância. Podes ver a minha configuração na [Figura C-1](#). Quero mostrar-te como podes criar o teu próprio espaço de trabalho para seres produtivos.



Figura C-1. Trabalha em casa

Equipamento para trabalhar à distância

Network+

Aqui está uma lista breve e não exaustiva de coisas a considerar se trabalhares remotamente. Uma rede doméstica fiável é provavelmente o item mais crítico em qualquer lista de verificação de trabalho remoto. Idealmente, podes obter uma ligação de fibra de baixo custo por menos de 100 dólares. A fibra é ideal porque obténs a mesma velocidade tanto para baixo como para cima. Nota que não só a fibra de 1 GB é padrão em muitas partes da América, como a fibra de 2 GB também está a tornar-se amplamente disponível.

Há alguns detalhes necessários a que deves prestar atenção ao configurar a tua rede doméstica. Vamos abordar estes pormenores a seguir.

Rede doméstica física

É melhor ligar a tua estação de trabalho à rede doméstica de fibra ótica ou por cabo através de Ethernet. Este passo elimina toda uma série de problemas que podem interferir com o trabalho remoto, nomeadamente questões relacionadas com a rede sem fios. Uma excelente forma de o fazeres é comprares um switch de rede barato com capacidade de 2,5 GB ou superior e ligá-lo com cabos de rede Cat6 (que oferecem velocidades até 10 Gbps). Assim, se tiveres uma ligação de fibra de 2 GB, podes aproveitar diretamente a velocidade total.

Para além da rede com fios, uma rede em malha para a rede sem fios pode render enormes dividendos. Uma configuração sem fios recomendada seria a utilização de um router WiFi 6 em malha. Estes permitem-te cobrir a tua casa, mesmo em áreas com mais de 5.000 pés quadrados, com velocidades sem fios que podem exceder 1 Gbps. As redes mesh modernas também permitem centenas de ligações simultâneas, o que deve ser mais do que suficiente para uma rede doméstica.

Gestão de energia e Network+ doméstico

Quando a tua casa se torna um escritório permanente, há dois aspectos essenciais a considerar: custo e fiabilidade. Por exemplo, se tiveres cortes

de energia frequentes, podes perder um rendimento comercial significativo. Da mesma forma, as tuas contas de eletricidade podem aumentar substancialmente.

Podes ajudar a salvar o ambiente, a reduzir os custos dos serviços públicos domésticos e a assegurar a continuidade do negócio durante as falhas de energia. Com uma instalação solar em casa, podes fazer o seguinte. Em primeiro lugar, utiliza uma UPS (fonte de alimentação ininterrupta) para tempestades e falhas de energia e liga-lhe o teu importante escritório em casa e o teu equipamento de rede doméstica.

Finalmente, uma bateria Tesla Powerwall, ou baterias semelhantes, pode fornecer dias de energia de reserva à medida que recarrega a partir da energia solar. Este tipo de configuração permite-te trabalhar apesar das grandes tempestades. Para tornar o negócio ainda mais agradável, há poupanças fiscais significativas para um projeto solar.

Área de trabalho em casa

Uma secretária de pé, um monitor de ecrã panorâmico, um bom microfone e uma boa câmara são muito úteis. A ideia geral é comprar equipamento que te leve a um dia mais produtivo e, esperemos, mais curto. Uma coisa importante a considerar é comprar as melhores ferramentas que podes pagar, uma vez que elas te permitem ganhar dinheiro e ser produtivo.

Saúde e espaço de trabalho

Podes incorporar uma secretária de pé para diminuir as lesões na zona lombar e promover mais atividade física? Que tal fazeres 100 balanços com kettlebell em cinco séries de 20 por dia? Podes programar uma caminhada diária quando o teu cérebro começa a sentir-se sobrecarregado? Estas coisas parecem insignificantes, mas as pequenas coisas fazem mudanças significativas para melhorar a tua saúde, produtividade e felicidade.

Finalmente, podes eliminar a maior parte dos hábitos corporativos de junk food e substituí-los por jejum intermitente associado a alimentos saudáveis?

Lê o [Apêndice F](#) sobre o jejum intermitente para saberes mais sobre a minha viagem e pesquisa sobre o mesmo.

Configuração do estúdio virtual do teu espaço de trabalho em casa

Ter um fundo que apareça por detrás da câmara pode acrescentar um enorme nível de profissionalismo. Esta configuração significa ter uma boa iluminação, alguma "encenação" e outros itens que contribuem para o fundo. Nota que a automatização da voz pode ser uma grande vantagem nestas configurações, uma vez que as luzes grandes não são necessárias até estares na câmara.

Na [Figura C-2](#), o meu fundo está configurado para ter um aspeto apelativo para chamadas de videoconferência. Este é um aspeto frequentemente esquecido do trabalho remoto.



Figura C-2. Fundo do estúdio

Localização, localização, localização

Podes mudar-te para uma região de baixo custo onde haja acesso à natureza e boas escolas? No mundo pós-pandémico, podemos esperar que uma nova

realidade permita que os trabalhadores remotos que pensam na ciência dos dados optimizem a saúde, a qualidade de vida e os custos. Um fator determinante é o custo da habitação própria. Em certas regiões dos Estados Unidos, não faz sentido construir uma força de trabalho, como a Bay Area. O JCHS (Joint Center for Housing Studies) da Universidade de Harvard tem muitas visualizações interactivas de dados **que explicam isto**. Trabalhas para viver ou vives para trabalhar?

Quando optimizas a tua vida em torno do equilíbrio trabalho-vida, família e saúde, podes achar que uma hipoteca cara pode ser contraproducente para os teus objetivos. Trabalhar remotamente permite-te ter em conta a tua despesa mais significativa: onde vives. Um bom site que te ajuda a decidir onde viver é o [Numbeo](#); podes ter em conta o clima, o custo de vida, a criminalidade, a educação e outros factores. Além disso, há muitos sítios incríveis para viver fora dos centros tecnológicos tradicionais, como Nova Iorque e São Francisco.

Apêndice D. Pensa como um VC para a tua carreira

Por Noah Gift

Uma consideração fundamental para construir uma carreira na aprendizagem automática é pensar como um VC (capitalista de risco). O que é que um VC faz de diferente de um empregado? Uma coisa que faz é preparar-se para o fracasso, investindo numa série de empresas. Então, porque não fazer o mesmo na tua carreira? Imagina que pensas numa estratégia multi-empresa desde o início. Nesse caso, podes sempre concentrar-te no processo a longo prazo de adquirir mais competências, construir projectos paralelos e carteiras de trabalho, bem como rendimentos fora do teu emprego.

Além disso, a compreensão das receitas e das despesas permite-te criar autonomia - autonomia para, digamos, deixares o que estás a fazer durante o verão e mergulhares a fundo na próxima grande tecnologia MLOps. Vamos ver como pensar nas receitas e despesas.

Estratégia de receitas da pera

Vivemos numa nova era, em que é possível começar um negócio com um computador portátil e uma ligação à Internet. Como consultor e empresário de longa data, desenvolvi uma estrutura que funciona para mim. Quando avalio com quem trabalhar e em que projeto trabalhar, penso em PPEAR ou "pera":

- *P (Passivo)*
- *P (Positivo)*
- *E (Exponencial)*

- *A (Autonomia)*
- *R (Regra dos 25%)*

Eis como podes utilizar este quadro "side gigs".

Passivo

Muitas pessoas saltam de emprego em emprego com o objetivo de obter um salário mais elevado, mas os salários são fixos: por muito bom que seja o trabalho que produzes, continuas a receber o mesmo salário. O rendimento passivo é uma forma de investir em resultados exponenciais. Todos os trabalhadores da área tecnológica devem ter uma mistura de salário e algum investimento num ativo que possa gerar resultados exponenciais. Idealmente, este resultado está diretamente ligado ao teu trabalho:

- Esta ação conduz a um rendimento passivo: livros, produtos, investimentos?
- És dono do cliente? O ideal é que te concentres em manter o cliente.
- Qual é a relação de direitos de autor?

Predador (20% ou menos)

Deve haver uma razão muito convincente para trabalhares com um predador. Talvez te dêem visibilidade ou te arrisquem. A desvantagem dos predadores é que muitas vezes têm um processo burocrático. Com quantas camadas de pessoas tens de interagir para conseguir fazer alguma coisa? Quanto tempo demoras a fazer alguma coisa? Pode ser 10 a 100 vezes mais demorado do que trabalhares sozinho.

Parceiro (50% ou mais)

Há muito a gostar numa parceria igualitária. O parceiro tem "pele no jogo" em termos de dinheiro e do seu tempo.

Plataforma (80% ou superior)

Há prós e contras na utilização de uma plataforma. As vantagens de uma plataforma são que podes reter a maior parte das receitas se fores autossuficiente. As desvantagens são que podes não ter ainda um ponto de referência. Podes não ter um enquadramento para o que é "bom". Podes querer trabalhar com um predador e ver como ele faz as coisas antes de ires diretamente para a plataforma.

NOTA

Nem toda a gente quer ser um autor ou criador, mas toda a gente pode ser um investidor. Talvez isso signifique colocar 50% do teu rendimento do W2 num fundo de índice ou arrendar uma casa.

Positivo

Quando trabalhas num projeto ou com um parceiro, a experiência deve ser positiva. Mesmo que recebas um bom salário, se o ambiente for tóxico, acaba por se tornar aborrecido. Algumas perguntas a fazer são:

- Sou feliz todos os dias?
- Respeito as pessoas com quem trabalho todos os dias?
- As pessoas com quem estou a trabalhar são pessoas com um historial de sucesso?
- A minha saúde aumenta ou mantém-se ao nível do sono, da forma física e da nutrição?
- Uma vez que és a média das cinco pessoas com quem passas mais tempo, como podes estar rodeado de pessoas positivas?

Exponencial

Outra questão importante sobre um projeto ou sobre o trabalho com um parceiro é o potencial exponencial. Talvez tenhas decidido trabalhar com um parceiro predador devido ao potencial exponencial do projeto. Por outro lado, se estás a trabalhar com um predador, mas o projeto não tem potencial exponencial, então talvez não seja um bom projeto.

Este projeto ou parceria conduz a uma reação exponencial em termos de receitas, utilizadores, tráfego, imprensa ou prestígio?

Autonomia

Outra questão importante num projeto ou no trabalho com um parceiro é a autonomia. Se és bom no que fazes, precisas de liberdade. Tu sabes o que é ser bom; o teu parceiro talvez não. Qual é o teu grau de autonomia? És capaz de apostar em ti próprio ou o sucesso está nas mãos de outras pessoas?

Alguns exemplos de perguntas são:

- Esta ação aumenta a autonomia ou cria uma dependência?
- Aprendo e cresço - uma nova competência, um novo prestígio ou uma afiliação a uma marca?
- É automatizável ou manual? Evita as tarefas que não podem ser automatizadas.

Regra dos 25%

De que cor é o dinheiro que ganhas? **A Figura D-1** mostra três formas de considerar o rendimento: empregado, consultor ou investidor.

Ser empregado pode ser valioso para ti porque aprendes competências e constróis uma rede de contactos. Mas não te esqueças de que se trata de dinheiro "vermelho". Este dinheiro vermelho pode desaparecer a qualquer momento. Não tens o controlo.

What color is your money?

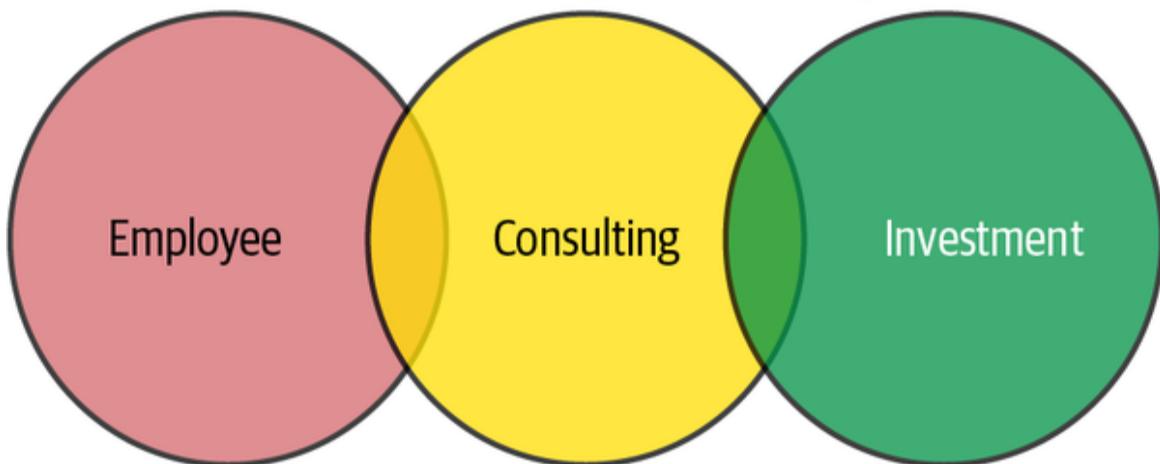


Figura D-1. De que cor é o teu dinheiro?

A consultoria é dinheiro "amarelo". É um grande passo na direção certa. Podes fazer consultoria enquanto trabalhas por conta de outrem, o que te retira alguns dos riscos de ser trabalhador por conta de outrem. No entanto, como consultor, tens de ter cuidado para nunca teres um cliente que represente mais de 25% do teu rendimento total e, idealmente, não mais de 25% do teu rendimento de consultoria. A familiaridade gera desprezo. As melhores relações acontecem quando as pessoas se comportam bem e sabem que a ligação só existe para resolver um problema.

Investimentos como imóveis, fundos de índice e produtos digitais são "dinheiro verde". Este fluxo de rendimentos pagar-te-á sempre. O cenário ideal é fazeres 80% do teu rendimento com dinheiro verde e limitares a consultoria ou o emprego a 20% do teu rendimento.

Notas

As seguintes notas e recursos foram úteis para a elaboração deste apêndice:

- Warren Buffet tem uma frase famosa sobre este assunto. Ele disse: "Se não encontrares uma forma de ganhar dinheiro enquanto dormes, vais trabalhar até morreres".
- Há bons conselhos relacionados no artigo "[1.000 fãs verdadeiros? Tenta 100](#)".

- Por último, agradeço os comentários e as ideias inspiradoras de [Andrew Hargadon](#) e [Dickson Louie](#).

Apêndice E. Construindo um Portfólio Técnico para MLOps

Por Noah Gift

Uma coisa que recomendo a toda a gente é tornares-te uma "ameaça tripla". No basquetebol, uma posição de ameaça tripla significa alguém que pode marcar, recuperar ou passar. O defensor tem de se proteger contra as três opções. Também podes ser uma ameaça tripla do ponto de vista da carreira, o que significa que podes passar pelos porteiros da contratação porque tens três ameaças: um portfólio técnico, uma certificação relevante e experiência profissional ou um diploma relevante.

Aconselho sobre Certificações Cloud no [Apêndice B](#), mas aqui vamos mergulhar no teu portfólio de tecnologia. A construção de um projeto de portfólio melhora o teu currículo, que terá um link para um código-fonte de alta qualidade e um vídeo de demonstração, em comparação com outro currículo que não o tenha. Quem chamarias para uma entrevista? Chamaria o candidato que já me tivesse mostrado o que sabe fazer antes de nos conhecermos.

Ao elaborar um projeto, escolhe algo que aches que vai "fazer mexer a agulha" nos próximos dois anos. Considera as seguintes ideias:

- Projeto GitHub com código fonte e um *README.md* que explique o projeto. O *README.md* deve ser de qualidade profissional e usar um estilo de escrita empresarial.
- Cadernos de notas ou código fonte 100% repetíveis.
- Trabalho original (não uma cópia de um projeto Kaggle).
- Paixão autêntica.

- Vídeo de demonstração final de cinco minutos que mostra como funciona.
- A demonstração tem de ser muito técnica e mostrar precisamente como realizar uma tarefa, ou seja, tens de ensinar alguém passo a passo. (Pensa num programa de culinária em que um chefe demonstra como fazer bolachas com pepitas de chocolate. Este nível de detalhe tem de ser semelhante).
- O vídeo deve ser, no mínimo, de 1080p com um formato 16:9.
- Considera a possibilidade de gravar com um microfone externo de baixo custo.

Algo a considerar na construção de um notebook Jupyter é dividir os passos de um projeto de ciência de dados. Normalmente, os passos são:

- Ingerir
- EDA (Análise Exploratória de Dados)
- Modelação
- Conclusão

Para projectos de portefólio MLOps, eis algumas ideias que resultaram na obtenção de emprego por parte dos estudantes em grandes empresas de tecnologia, ou seja, FAANG, e startups de ponta como engenheiros de ML, engenheiros de dados e cientistas de dados.

Projeto: Entrega contínua da API de engenharia de dados Flask/FastAPI numa plataforma PaaS

Uma excelente forma de testares os teus conhecimentos sobre a construção de APIs é o seguinte projeto:

1. Cria uma aplicação Flask ou Fast numa plataforma Cloud e envia o código fonte para o GitHub.
2. Configura um servidor de compilação nativo da Cloud (AWS App Runner, AWS Code Build, etc.) para implementar alterações no GitHub.
3. Cria uma API de engenharia de dados realista.

Podes consultar este [passo-a-passo](#) da O'Reilly sobre a utilização de funções com a FastAPI para criar microsserviços para obteres mais ideias, ou este [projeto de exemplo do Github](#) para um projeto de arranque completo do AWS App Runner para a FastAPI.

Projeto: Projeto Docker e Kubernetes Container

Muitas soluções Cloud envolvem contentores em formato Docker. Vamos aproveitar os contentores em formato Docker no seguinte projeto:

1. Cria um contentor Docker personalizado a partir da versão atual do Python que implementa uma aplicação Python ML.
2. Envia a imagem para o DockerHub, Amazon ECR, Google Container Registry ou outro Cloud Container Registry.
3. Baixa a imagem e executa-a numa plataforma Cloud Shell: Google Cloud Shell ou AWS Cloud9.
4. Implementa uma aplicação num cluster Kubernetes gerido na Cloud, como o GKE (Google Kubernetes Engine), ou o EKS (Elastic Kubernetes Service), etc.

Projeto: Pipeline de engenharia de dados de IA sem servidor

Reproduz a arquitetura do projeto de engenharia de dados sem servidor de exemplo mostrado na [Figura E-1](#). Em seguida, aprimora o projeto ampliando a funcionalidade da análise de PLN: adicionando extração de entidade, extração de frase-chave ou algum outro recurso de PLN.

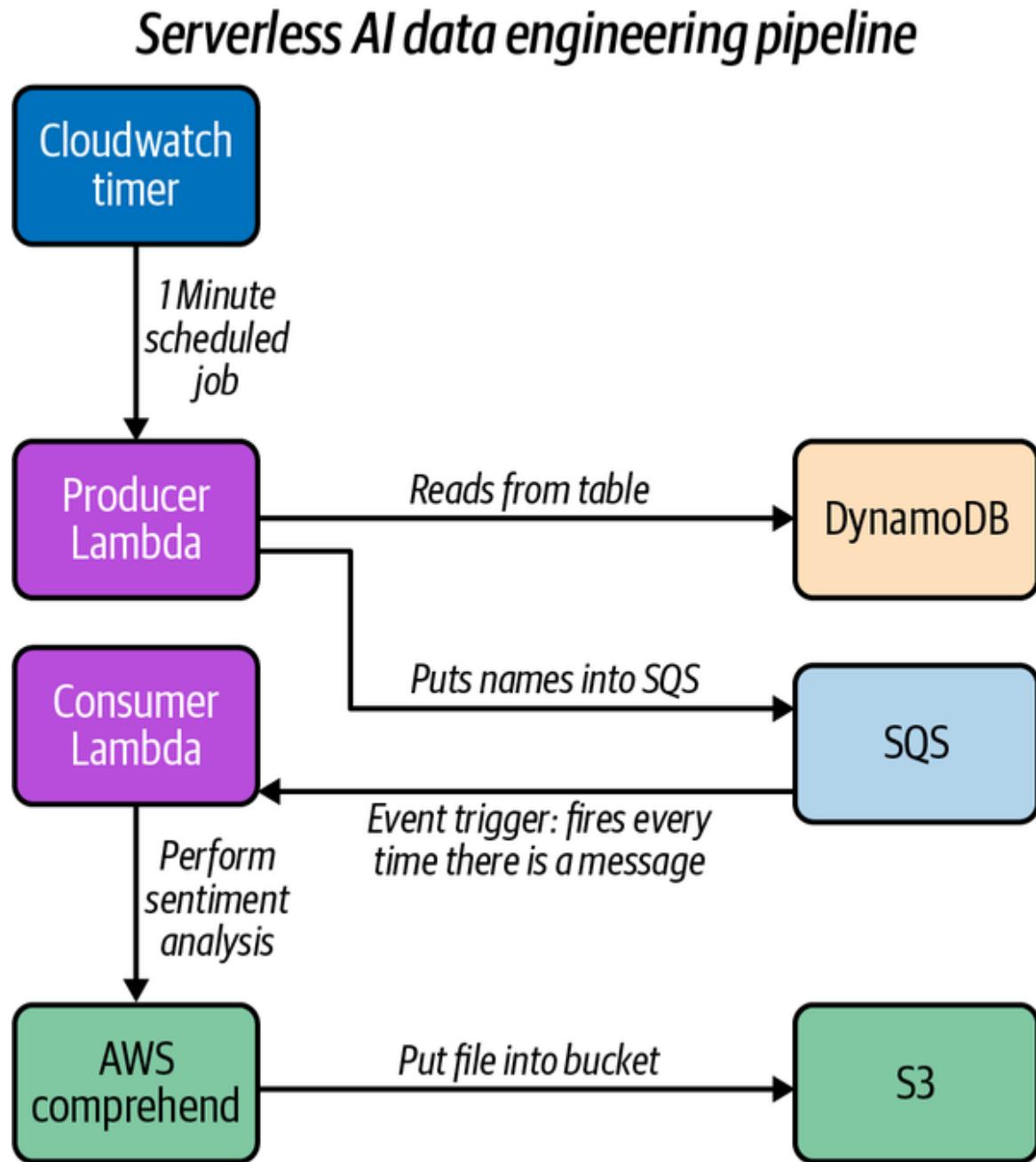


Figura E-1. Engenharia de dados sem servidor

Um bom recurso para este projeto é o seguinte repositório:

<https://github.com/noahgift/awslambda>.

Projeto: Constrói a solução Edge ML

Constrói e implementa uma solução de visão por computador baseada na extremidade utilizando uma das tecnologias abordadas no livro:

- Intel Movidius Neural Compute Stick 2
- AWS DeepLens
- Coral AI
- SmartPhone (iOS, Android)
- Raspberry Pi

Aqui estão os resultados do teu projeto:

- Publica os teus resultados como um bloco de notas Jupyter, uma pasta de blocos de notas Colab, ou ambos num repositório GitHub.
- Escreve uma sinopse de duas páginas de um projeto de pesquisa em formato PDF.
- Inclui um link para os teus trabalhos publicados na tua sinopse de duas páginas.
- Cria um vídeo de demonstração de 60 segundos do teu projeto de visão por computador a fazer inferência (previsões).
- Submete o teu projeto à Intel ou à AWS Community Projects.

Projeto: Cria uma aplicação ou API ML nativa na Cloud

Constrói uma aplicação analítica nativa da nuvem alojada no Google Cloud Platform (GCP), AWS, Azure ou outra nuvem ou tecnologia (ou seja, Kubernetes). Este projeto tem como objetivo dar-te a capacidade de criar soluções realistas e funcionais que funcionem com técnicas modernas.

Antes de começares, lê "["Hidden Technical Debt in Machine Learning Systems"](#) de Sculley et al. (2015).

Uma boa ideia para limitar a complexidade é utilizar um conjunto de dados públicos. Um exemplo pode ser um projeto que usa dados públicos de um conjunto de dados do Google BigQuery se estiver usando o GCP. Como alternativa, se usares o AutoML, os dados podem ser dados de tutorial ou dados personalizados.

A ideia principal é que penses em começar a criar um portefólio. Aqui está uma lista de requisitos de projeto sugeridos para um projeto GCP; podes modificar a pilha tecnológica para uma Cloud diferente.

- Código fonte armazenado no GitHub
- Implementação contínua a partir do CircleCI
- Dados armazenados no GCP (BigQuery, Google Cloud Storage, etc.)
- Previsões de ML criadas e fornecidas (AutoML, BigQuery, etc.)
- Stackdriver instalado para monitorização
- O Google App Engine envia pedidos HTTP através da API REST com um payload JSON
- Implementado no ambiente GCP
- Um documento de duas páginas, a um só espaço, descrevendo o projeto tal como um consultor o descreveria a um cliente durante a fase de entrega

Segue-se um exemplo de uma lista de verificação do projeto final a ter em conta.

- Realiza previsão/inferência de ML?
- Existem ambientes separados?
- Existe uma monitorização e alertas abrangentes?

- É utilizado o armazenamento de dados correto, ou seja, relacional, gráfico, chave/valor?
- Aplica-se o princípio do menor privilégio?
- Os dados são encriptados em trânsito e em repouso?
- Fizeste um teste de carga da aplicação para verificar o desempenho?

Arranja um emprego: Não invadas o castelo, entra pela porta das traseiras

A indústria tecnológica tem sempre um título profissional de "sonho". Estes títulos vão e vêm. Aqui tens alguns: Administrador de sistemas Unix, administrador de rede, webmaster, desenvolvedor web, desenvolvedor móvel, cientista de dados. O que acontece quando estes títulos de emprego aparecem é que as empresas entram em pânico com a contratação destes cargos.

Depois, todos os progressos são interrompidos e aparecem cada vez mais obstáculos. Um exemplo clássico é pedir a alguém dez anos de experiência numa tecnologia, quando ela existe há um ano. Finalmente, torna-se impossível entrar no "castelo". A frente do castelo tem guardas, óleo quente, lanças e um monstro à espera no fosso. Em vez disso, pensa numa porta dos fundos. Muitas vezes, uma porta dos fundos é um cargo menos prestigiado.

Como aprender

As carreiras relacionadas com o software são únicas em comparação com outras profissões. As profissões relacionadas com software têm muito mais em comum com atletas profissionais, artistas marciais ou músicos. O processo de alcançar a mestria envolve o sofrimento e o gosto por cometer erros.

Cria os teus próprios 20% de tempo

Nunca confies numa empresa para ser a única fonte do que precisas de aprender. Tens de traçar o teu caminho de aprendizagem. Uma forma de o fazer é passar algumas horas por dia a aprender novas tecnologias como um hábito. Pensa nisso como um exercício para a tua carreira.

Aceita a mentalidade de erro

É comum evitarmos erros e querermos a perfeição. Por exemplo, tentamos evitar acidentes de viação, compras deixadas cair e outros erros, e a maioria dos estudantes quer obter um "A" perfeito num exame.

Para aprenderes a ser um engenheiro de software competente, é melhor inverteres esta ideia. Um fluxo constante de erros significa que estás no caminho certo. Quantos erros cometeste esta semana? Quantos cometeste hoje? William Blake disse isto da melhor forma em 1790, quando afirmou: "Se o tolo persistisse na sua loucura, tornar-se-ia sábio".

Encontrar passatempos paralelos para testar a tua aprendizagem

Se perguntasses a um quórum de engenheiros de software bem sucedidos com mais de 10 anos de experiência, irias ouvir uma versão do seguinte: "Eu sou uma máquina de aprendizagem". Então, como é que uma máquina de aprendizagem se torna melhor a aprender? Um método é escolher um desporto que demora anos a dominar, no qual és um completo principiante, e observar-te a ti próprio.

Dois jogos em particular que se adequam bem a esta atividade são a escalada e o Jiu-Jitsu brasileiro. O Jiu-Jitsu brasileiro tem o efeito secundário adicional de te ensinar autodefesa prática. Descobrirás que tens pontos cegos na aprendizagem que poderás corrigir no teu trabalho "real".

Apêndice F. Estudo de caso de ciência de dados: Jejum intermitente

Por Noah Gift

No início dos anos 90, frequentei a Cal Poly San Luis Obispo e formei-me em ciências da nutrição. Escolhi este curso porque estava obcecado em ser um atleta profissional. Senti que estudar ciências da nutrição poderia dar-me uma vantagem extra. Primeiro encontrei uma pesquisa sobre restrição calórica e envelhecimento.

Também participei numa auto-experimentação na minha aula de bioquímica nutricional. Centrifugámos o nosso sangue e calculámos os níveis de LDL, HDL e colesterol total. No mesmo curso, tomámos um suplemento de megadoses de vitamina C e depois recolhemos a nossa urina para ver o que era absorvido. Descobrimos que nada foi absorvido numa população saudável de estudantes universitários porque o corpo responde de forma inteligente à absorção de nutrientes, aumentando a sensibilidade à absorção quando os níveis são baixos. Os suplementos vitamínicos são muitas vezes um desperdício de dinheiro.

Fiz um ano de anatomia e fisiologia e aprendi a dissecar o corpo humano. Aprendi sobre o ciclo de Krebs e como funciona o armazenamento de glicogénio.¹ O corpo produz insulina para aumentar o açúcar no sangue e armazena-o no fígado e nos tecidos musculares. Se essas áreas estiverem "cheias", coloca esse glicogénio no tecido adiposo, ou "gordura". Da mesma forma, quando o corpo não tem glicogénio ou quando está em atividade aeróbica, o tecido adiposo é o principal combustível. Este armazenamento é o nosso depósito de combustível "extra".

Também passei um ano na Cal Poly como decatleta da Divisão I. Uma das coisas que aprendi da maneira mais difícil foi que fazer demasiado

levantamento de pesos era ativamente prejudicial para o desempenho em desportos como a corrida. Eu tinha 1,80 m de altura, 90 kg e conseguia levantar 225 pesos cerca de 25 vezes (semelhante ao desempenho de um defesa da NFL no levantamento de pesos). Também corria os 1500 metros (cerca de uma milha) em 4 minutos e 30 segundos e liderava regularmente o grupo em treinos de três milhas com corredores de longa distância da Divisão I. Também conseguia afundar uma bola de basquetebol perto da linha de lance livre e corria os 100 metros em 10,9.

Resumindo, eu era um bom atleta e bem preparado, mas trabalhei ativamente durante anos fazendo os tipos errados de exercícios (musculação). A minha ética de trabalho era fora de série, mas também extremamente ineficaz e contraproducente para o desporto que escolhi. Também sobreestimei a minha capacidade de entrar num desporto da I Divisão onde nem sequer tinha feito muitas atividades, como o salto com vara. Também quase entrei para a equipa - havia uma pessoa à minha frente. Mas, nesta parte da minha vida, o "quase" não contava. Esta experiência foi a primeira vez que dei a algo toda a minha atenção e esforço, mas acabei por falhar. Foi uma experiência de humildade que foi bom tirar do caminho logo no início da vida. Aprendi a lidar com o fracasso, o que me serviu muito bem na engenharia de software e na ciência de dados.

Como antigo engenheiro de software de Silicon Valley, descobri mais tarde uma palavra para este comportamento: YAGNI. YAGNI significa "You Ain't Gonna Need It" (Não vais precisar disso). Tal como nos anos em que passei a engordar 40 quilos de músculo extra que acabaram por diminuir o meu desempenho desportivo, podes trabalhar nas coisas erradas em projectos de software. Exemplos disto incluem a construção de funcionalidades que não usas numa aplicação ou abstrações demasiado complexas como a programação avançada orientada a objectos. Estas técnicas são literalmente "peso morto". São ativamente prejudiciais porque levam tempo a desenvolver, que poderia ser gasto em coisas valiosas, e atrasam permanentemente o projeto. Tal como na minha experiência de atletismo, algumas das pessoas mais motivadas e talentosas podem ser as que mais abusam da adição de complexidade desnecessária a um projeto.

O campo da ciência nutricional também tem um problema YAGNI, e o jejum intermitente é um excelente exemplo de uma técnica de simplificação. Funciona de forma muito semelhante à forma como apagar metade de um ensaio de 2.000 palavras pode torná-lo melhor. Acontece que as décadas de "complexidade" adicional na alimentação podem ser ignoradas e eliminadas: lanches frequentes, pequeno-almoço e alimentos ultraprocessados.²

Não precisas de tomar o pequeno-almoço nem de lanchar. Para simplificar ainda mais, não precisas de comer muitas vezes por dia. É uma perda de tempo e de dinheiro. Também não precisas de alimentos ultra-processados: cereais de pequeno-almoço, barras de proteínas ou qualquer outro alimento "feito pelo homem". Afinal, a YAGNI volta a atacar com a nossa dieta. Também não precisas de comprar uma ferramenta única para teres uma alimentação saudável, como livros, suplementos ou planos de refeições.

Há um problema bem conhecido chamado problema do caixeiro viajante,³ que coloca a seguinte questão: Dada uma lista de cidades e as distâncias entre cada par de cidades, qual é o percurso mais curto possível que visita cada cidade exatamente uma vez e regressa à cidade de origem? O problema é essencial porque não existe uma solução perfeita. Em linguagem corrente, isto significa que uma solução é demasiado complexa para ser implementada no mundo real. Além disso, demoraria cada vez mais tempo a criar uma resposta relativa aos dados. Por isso, em vez disso, a informática resolve estes problemas utilizando heurísticas. Escrevi uma solução heurística na pós-graduação que não é particularmente inovadora, mas dá uma resposta razoável.⁴ A forma como funciona é escolher uma cidade aleatoriamente e, em seguida, escolhe sempre o percurso mais curto quando lhe são apresentados percursos possíveis. No final da solução, calcula a distância total. Depois, repete esta simulação com o tempo que tiveres e escolhes a distância mais curta.

O jejum intermitente é tão eficaz porque também ultrapassa a complexidade insolúvel de contar calorias para perder peso. O jejum intermitente é uma heurística eficaz. Em vez de contares as calorias, não comes durante blocos do dia.⁵ Estes blocos podem ser os seguintes:

Jejuns diários:

- Janela de alimentação de 8 horas ou 16:8
 - 12h00 - 20h00
 - Das 7h às 15h.
- Janela de alimentação de 4 horas ou 20:4
 - 18:00 - 22:00
 - Das 7h às 11h.

Jejuns mais longos com padrões mais complexos:

- 5:2
 - Cinco dias de alimentação normal e dois dias de restrição calórica, normalmente 500 calorias.
- Jejum de dias alternados
 - Come normalmente num dia e restringe as calorias noutro, normalmente 500 calorias.

Fiz experiências principalmente com jejuns diários de 16 horas ou 20 horas. Como cientista de dados, nutricionista e ainda atleta de competição, também tenho dados. Tenho dados de 2011-2019 sobre o meu peso corporal.⁶ De agosto de 2019 a dezembro de 2019, tenho estado maioritariamente numa rotina 12:8 IF.

Na **Figura F-1**, posso utilizar a recolha de dados da minha balança para fazer ciência de dados no meu próprio corpo e descobrir o que funciona e o que não funciona.



Figura F-1. Peso corporal

Uma coisa que aprendi ao analisar o peso corporal e ao fazer experiências com dados é que algumas pequenas coisas fazem uma grande diferença:

- Evita os alimentos "feitos pelo homem"
- Dorme 8 horas (os MBA e as startups provocaram um aumento de peso devido à perda de sono)
- Exercício diário
- Jejum intermitente
- Não podes fazer exercício para te livrares de uma má dieta (o ritmo cardíaco estava nos 40)

A figura F-2 mostra um exemplo de uma refeição aprovada pelo YAGNI.



Figura F-2. Comida saudável: omeleta de abacate

Aqui tens a receita de uma omeleta de cogumelos com abacate:

- Ovos
- Cogumelos Shiitake
- Queijo
- Abacate

- Salsa

Demora apenas alguns minutos a fazer, as gorduras e os alimentos integrais fazem-te sentir saciado e é barato.

Quando tive "excesso de peso" foi nos períodos em que não segui os conselhos anteriores: trabalhar horas loucas em startups, comer comida feita por "humanos". Trabalhar em jejum demora um pouco a habituar-me, mas descobri que aumenta o desempenho em muitos desportos que pratico: boulder, levantamento de pesos, treino HIIT e Jiu-Jitsu brasileiro. Da mesma forma, sou muito produtivo a escrever software, a escrever livros e a fazer trabalho intelectual. O meu principal "hack" é o consumo regular de café frio e água.

A minha conclusão é que o jejum intermitente é uma das melhores formas de melhorar drasticamente a vida de uma pessoa. Não custa nada e é simples de fazer, principalmente se o praticares diariamente, e é apoiado pela ciência. Além disso, muitas pessoas têm dificuldade em encontrar projectos de ciência de dados e de aprendizagem automática interessantes para trabalhar. Porque não te usas a ti próprio como caso de teste, como mostra este estudo de caso?

Notas sobre o jejum intermitente, a glicemia e a alimentação

De acordo com o *New England Journal of Medicine* (NEJM), "Há cada vez mais provas de que comer em 6 horas e jejuar durante 18 horas pode desencadear uma mudança metabólica da energia baseada na glicose para a energia baseada na cetona, com maior resistência ao stress, maior longevidade e menor incidência de doenças, incluindo cancro e obesidade".

De acordo com o NHS (Nurse's Health Study), "Vários comportamentos relacionados com o estilo de vida podem influenciar o facto de uma pessoa conseguir ou não manter o equilíbrio energético a longo prazo. Por exemplo, o consumo de bebidas açucaradas, doces e alimentos processados

pode dificultar essa tarefa, ao passo que o consumo de cereais integrais, frutas e legumes pode facilitar essa tarefa."

Isto também mostra uma abordagem de ciência de dados e de aprendizagem automática para resolver o problema da obesidade. Aumenta o número de porções de frutos secos, fruta e iogurte. Diminui ou elimina as batatas fritas, as batatas e as bebidas açucaradas (nota que existe uma ligação entre os alimentos ultraprocessados e os picos de insulina). Estes são os principais alimentos que contribuíram para o aumento de peso:

- Batatas fritas
- Batatas
- Bebidas açucaradas

Estes são os principais alimentos que estão inversamente associados ao aumento de peso (perda de peso):

- Nozes
- Frutas
- Iogurte

Mudanças de estilo de vida como a IF são certamente mais fáceis de tentar quando já consegues ver os dados por detrás delas!

1 Vê a página do ciclo do ácido cítrico na [Wikipédia](#).

2 Ver "Eating More Ultra-Processed Foods May Shorten Life Span" na [Harvard Health Publishing](#).

3 Vê a descrição na [Wikipédia](#).

4 Está disponível no [GitHub](#).

5 Para mais informações, consulta o [DietDoctor](#) e o [New England Journal of Medicine](#).

6 Está disponível no [GitHub](#).

Apêndice G. Recursos educativos adicionais

Por Noah Gift

Todos nós sabemos que um livro, um curso ou um diploma não é suficiente - em vez disso, um processo de aprendizagem contínua é o melhor método para te manteres atualizado. Uma forma de continuar a aprender é formar um grupo de estudo no teu trabalho, com os teus amigos ou na escola para continuar a crescer. Os seguintes recursos das aulas que lecciono em engenharia de ML, MLOps e visão computacional aplicada podem ajudar-te a começar.

Perguntas adicionais sobre o pensamento crítico dos MLOps

- Um antigo gestor de engenharia de uma startup menciona que a gestão de projectos "Agile" por si só não é suficiente para enviar um produto mínimo viável (MVP). Muitas vezes, também é necessário um calendário semanal de 3 meses (ou seja, planeamento em cascata). Discute a tua resposta a esta opinião.
- Que problemas resolve um sistema de integração contínua (IC)?
- Porque é que um sistema de IC é uma parte essencial do software SaaS?
- Porque é que as plataformas Cloud são o alvo ideal para as aplicações analíticas?
- Como é que a aprendizagem profunda beneficia com a Cloud?

- Quais são as vantagens dos serviços geridos como o Google BigQuery?
- Em que é que o Google BigQuery difere de um SQL tradicional?
- Como é que a previsão de aprendizagem automática (ML) diretamente do BigQuery acrescenta valor à plataforma Google?
- Que vantagens poderá ter para a engenharia de aplicações analíticas?
- Como é que o AutoML tem um custo total de propriedade (TCO) mais baixo?
- Como é que pode ter um TCO mais elevado?
- Que problemas é que os diferentes ambientes resolvem?
- Que problemas é que um ambiente diferente cria?
- Como podes gerir corretamente os custos inesperados na Cloud?
- Quais são as três ferramentas que te ajudam a gerir os custos no Google Cloud Platform?
- Quais são as três ferramentas que te ajudam a gerir os custos na plataforma AWS?
- Quais são as três ferramentas que te ajudam a gerir os custos na Plataforma Azure?
- Porque é que o registo em JavaScript Object Notation (JSON) pode muitas vezes ser melhor do que o registo não estruturado?
- Quais são as desvantagens dos alertas que disparam demasiado?
- Cria um plano de aprendizagem ao longo da vida, respondendo às seguintes perguntas: Que competências vais aprender neste trimestre, porquê e como? Que competências vais aprender até ao final deste ano, porquê e como? Que competências vais aprender

até ao final do próximo ano, porquê e como? Que competências vais aprender até ao final de cinco anos, porquê e como?

- Que problemas é que um sistema de entrega contínua (CD) resolve?
- Porque é que um sistema de entrega contínua é uma parte essencial da engenharia de dados?
- Qual é a diferença fundamental entre integração contínua e entrega contínua?
- Explica como a monitorização e o registo desempenham um papel fundamental na engenharia de dados.
- Explica o que pode correr mal nos exames de saúde.
- Explica por que razão a "governação de dados" é o "herói desconhecido" da cibersegurança.
- Explica como os testes desempenham um papel fundamental na engenharia de dados.
- Explica como a automatização e os testes estão tão intimamente ligados.
- Escolhe uma framework de linha de comandos Python favorita e escreve um exemplo do mundo real, e partilha. Podes explicar porque o escolhestes?
- Explica como a computação em Cloud está a ter impacto na engenharia de dados.
- Explica como o serverless está a ter impacto na engenharia de dados.
- Partilha uma função simples do Python AWS Lambda e explica o que faz.
- Explica o que é a engenharia de aprendizagem automática.

- Cria e partilha um `Dockerfile` simples que executa uma aplicação Flask. Explica como funciona.
- Explica o que é a engenharia de dados.
- Trunca e baralha um grande conjunto de dados, carrega-o no Pandas e partilha o teu trabalho. Explica a abordagem que utilizaste.
- Explica o que é o DevOps e como melhora um projeto de engenharia de dados.
- Qual é o problema que a Cloud resolve quando consideras os problemas de visão computacional do mundo real?
- Como é que os blocos de notas Colab e os Jupyter Notebooks podem ser utilizados para trocar ideias ou construir portefólios de pesquisa?
- Quais são as diferenças fundamentais entre a visão biológica e a visão artificial?
- Quais são alguns casos de utilização real da modelação generativa?
- Explica como as máquinas de jogo podem utilizar a visão por computador para ganhar.
- Quais são os prós e os contras da utilização de APIs de visão computacional para resolver problemas do mundo real?
- Como é que o AutoML está a afetar a ciência dos dados atualmente e como irá afectá-la no futuro?
- Quais são os casos de utilização real da aprendizagem automática com base nos limites?
- Quais são algumas das plataformas de aprendizagem automática baseadas em tecnologia de ponta?

- Como é que as plataformas integradas se enquadram nas estratégias de ML das empresas existentes?
- Como é que o SageMaker pode mudar a criação de modelos de aprendizagem automática nas organizações?
- Quais são os casos de utilização práticos do AWS Lambda?
- Explica casos práticos de utilização da aprendizagem por transferência. Explica como poderias utilizá-la num projeto.
- Como é que podes levar um modelo de ML que construíste para uma fase mais avançada de funcionalidade?
- O que é a IAC e que problema resolve?
- Como é que uma empresa deve decidir qual o nível de abstração da Cloud a utilizar para um projeto: SaaS, PaaS, IaaS, MaaS, sem servidor?
- Quais são as diferentes camadas de segurança de rede na AWS e quais os problemas únicos que cada uma resolve?
- Que problema é que as instâncias AWS Spot resolvem e como é que as podes utilizar nos teus projectos?
- Cria um contentor em formato Docker e recomenda a sua utilização num projeto.
- Avalia os serviços de gestão de contentores, como o Kubernetes e o Kubernetes alojado, e cria uma solução com eles.
- Resume os registos de contentores e como os utilizar para criar contentores personalizados.
- O que são contentores?
- Que problema é que os contentores resolvem?
- Qual é a relação entre Kubernetes e contentores?

- Avalia com precisão os desafios e oportunidades da computação distribuída e aplica estes conhecimentos a projectos reais.
- Resume como a consistência eventual desempenha um papel nas aplicações nativas da Cloud.
- Como é que o Teorema CAP desempenha um papel no design para a Cloud?
- Quais são as implicações da lei de Amdahl para os projectos de aprendizagem automática?
- Recomenda casos de utilização adequados para o ASICS.
- Considera as implicações do fim da Lei de Moore.
- Quais são os problemas de uma abordagem "tamanho único" às bases de dados relacionais?
- Como é que um serviço como o Google BigQuery pode mudar a forma como lidas com os dados?
- Que problema é que uma base de dados "sem servidor" como a Athena resolve?
- Quais são as diferenças fundamentais entre o armazenamento em bloco e o armazenamento de objectos?
- Quais são os problemas fundamentais que um lago de dados resolve?
- Quais são as soluções de compromisso com uma arquitetura sem servidor?
- Quais são as vantagens de desenvolver com a Cloud9?
- Que problemas é que o Google App Engine resolve?
- Que problemas é que o ambiente Cloud Shell resolve?

Materiais educativos adicionais do MLOps

Para além dos recursos deste livro, estes são recursos adicionais actualizados frequentemente que podes utilizar para continuar a melhorar:

- Muitos outros vídeos relacionados com MLOps são actualizados semanalmente na Plataforma O'Reilly pela [Pragmatic AI Solutions](#). Também podes rever os Katacodas na Plataforma de Aprendizagem O'Reilly.
- No [sítio Web da Pragmatic AI Labs](#) estão disponíveis vários livros gratuitos.
- Subscreve o [canal de YouTube da Pragmatic AI Labs](#).
- Faz a especialização da Duke e da Coursera [Building Cloud Computing Solutions at Scale](#).

Educação - Perturbação

Uma perturbação é uma rutura, interrupção ou alteração do modelo ou processo estabelecido. Esta secção apresenta as minhas reflexões sobre a perturbação educativa e a forma como esta afecta a aprendizagem das técnicas MLOps.

A disruptão é sempre fácil de detetar em retrospectiva. Considera a questão dos taxistas e dos serviços actuais como o Lyft e o Uber. Como é que exigir que os motoristas paguem [um milhão de dólares por um medalhão de táxi](#) fazia sentido como mecanismo para facilitar o serviço público de táxi([Figura G-1](#))?



Figura G-1. Medalhão de táxi¹

Quais foram os problemas que empresas como a **Lyfte** e **Uber** resolveram?

- Preço mais baixo
- Empurrar versus Puxar (o condutor vem até ti)
- Serviço previsível
- Ciclo de feedback para a criação de hábitos
- Assíncrono por conceção
- Digital versus analógico
- Fluxo de trabalho não linear

Consideraremos essas mesmas ideias no que diz respeito à educação.

Estado atual do ensino superior que vai sofrer alterações

A educação está a sofrer uma perturbação semelhante. De acordo com a Experian, a dívida dos estudantes atingiu o seu ponto mais alto de sempre, com uma taxa de crescimento linear desde 2008, como mostra a Figura G-2.

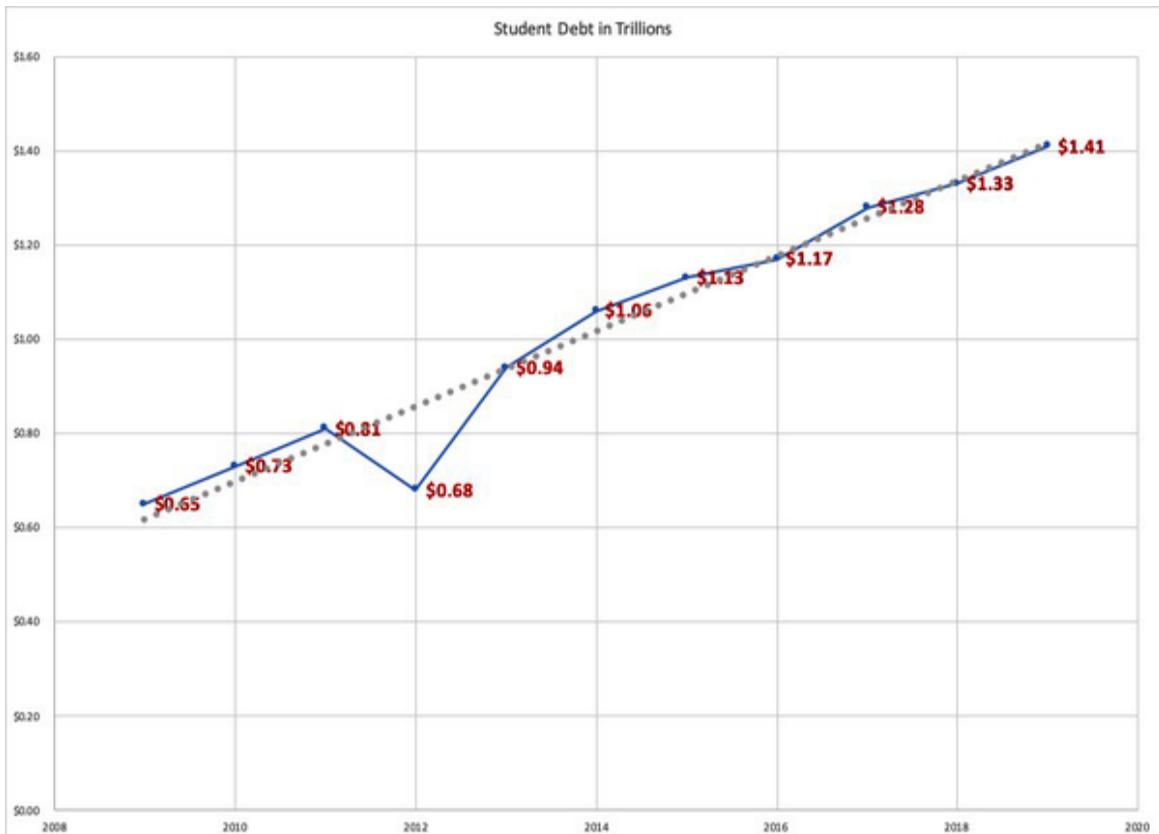


Figura G-2. Dívida estudantil da Experian

Simultaneamente a esta tendência preocupante, há uma estatística igualmente preocupante: 4/10 dos licenciados em 2019 estavam num emprego que não exigia o seu diploma (Figura G-3).

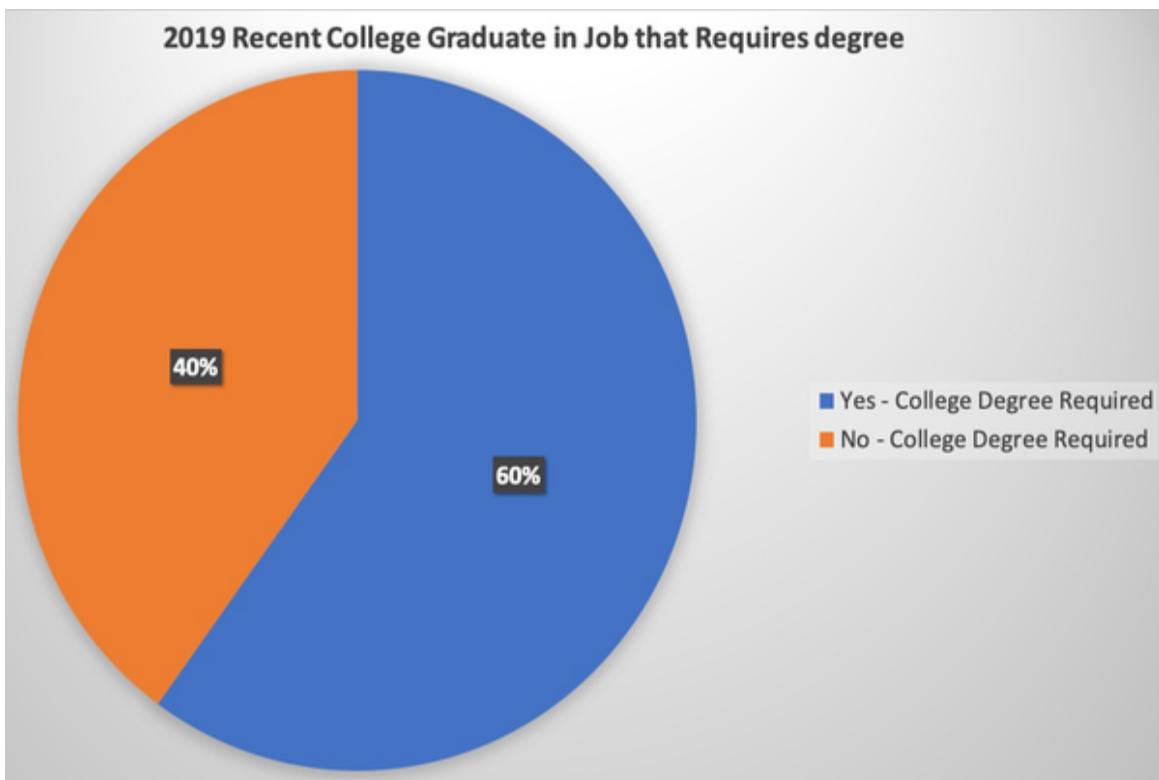


Figura G-3. Empregos que exigem diploma

Este processo não é sustentável. A dívida dos estudantes não pode continuar a crescer todos os anos e, ao mesmo tempo, produzir quase metade dos resultados que não conduzem diretamente a um emprego. Porque é que um estudante não há-de passar quatro anos a aprender um passatempo como fitness pessoal, desporto, música ou artes culinárias, se o resultado é o mesmo? Pelo menos, nessa situação, não estaria endividado e teria um passatempo divertido que poderia usar para o resto da vida.

No livro *Zero to One* de Peter Thiel (Currency), menciona a regra dos 10X. Afirma que uma empresa terá de ser 10 vezes melhor do que o seu concorrente mais próximo para ter sucesso. Poderá um produto ou serviço ser 10 vezes melhor do que o ensino tradicional? Sim, pode.

Educação 10 vezes melhor

Então, como seria na prática um sistema educativo 10X?

Aprendizagem integrada

Se o objetivo de um programa educativo fosse o emprego, por que não fazer formação profissional enquanto se está na escola?

Concentra-te no cliente

Grande parte do atual sistema de ensino superior centra-se no corpo docente e na sua pesquisa. Quem está a pagar por isso? O cliente, o estudante, é que paga. Um critério essencial para os educadores é a publicação de conteúdos em revistas de prestígio. Existe apenas uma ligação indireta com o cliente.

Entretanto, empresas como a **Udacity**, a Coursera, a O'Reilly e a **Edx** estão a oferecer diretamente aos clientes estes produtos. Esta formação é específica para o trabalho e é continuamente actualizada a um ritmo muito mais rápido do que uma universidade tradicional.

Pode acontecer que as competências ensinadas a um estudante se concentrem apenas na obtenção de um emprego. A maioria dos estudantes está concentrada em arranjar emprego. Estão menos concentrados em tornarem-se melhores seres humanos. Há outras saídas para este objetivo.

Reduz o tempo de conclusão

Um curso superior tem de demorar quatro anos a concluir? Pode demorar esse tempo se a maior parte do tempo do curso for dedicado a tarefas não essenciais. Porque é que um curso não pode demorar um ano ou dois anos?

Custo mais baixo

De acordo com a **USNews**, a média das propinas anuais de quatro anos é de 10 116 dólares para uma universidade pública estatal; 22 577 dólares para uma universidade pública não estatal; e 36 801 dólares para uma universidade privada. O custo total da obtenção de um diploma de quatro anos (ajustado pela inflação) tem aumentado sem limites desde 1985 (**Figura G-4**).

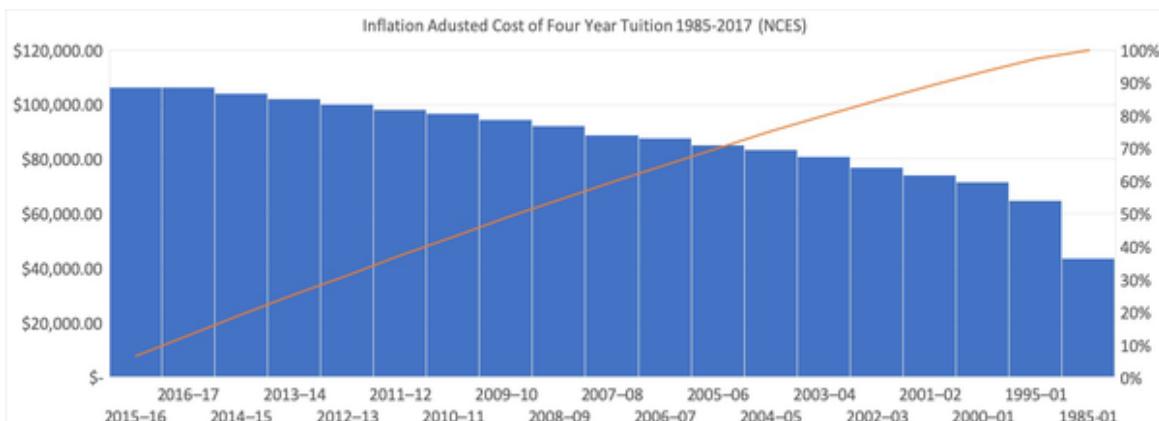


Figura G-4. Inflação das propinas ajustada

Poderá um concorrente oferecer um produto 10 vezes mais barato? Um ponto de partida seria desfazer o que aconteceu de 1985 a 2019. Se o produto não melhorou, mas o custo triplicou, então estás pronto para a rutura.

Assíncrono e remoto primeiro

Muitas empresas de engenharia de software decidiram tornar-se "remote first". Noutros casos, empresas [como o Twitter estão a mudar para uma força de trabalho distribuída](#). Na construção de software, o resultado é um produto digital. Se o trabalho for digital, o ambiente pode ser totalmente assíncrono e remoto. A vantagem de um curso assíncrono e remoto primeiro é a distribuição em escala.

Uma das vantagens de um ambiente "remoto em primeiro lugar" é uma estrutura organizacional centrada nos resultados mais do que na localização. Há enormes interrupções e desperdícios em muitas empresas de software devido a reuniões desnecessárias, ambientes de trabalho ruidosos e longas deslocações. Muitos estudantes irão para ambientes de trabalho "remoto em primeiro lugar", e pode ser uma vantagem significativa para eles aprenderem as competências para serem bem sucedidos nestes ambientes.

Inclusão primeiro versus exclusão primeiro

Muitas universidades declararam publicamente o número de estudantes que se candidataram ao seu programa e o número de estudantes que foram aceites.

Esta abordagem baseada na exclusão em primeiro lugar foi concebida para aumentar a procura. Se o bem vendido for físico, como uma casa de praia em Malibu, então sim, o preço será ajustado com base no mercado. Se o produto vendido for digital e infinitamente escalável, então a exclusão não faz sentido.

No entanto, não há almoços grátis e os programas do tipo "boot camp" não estão isentos de problemas. Em particular, a qualidade do currículo e a qualidade do ensino não devem ser negligenciadas.

Não linear versus série

Antes da tecnologia digital, muitas tarefas eram operações contínuas. Um bom exemplo é a tecnologia de edição televisiva. Nos anos 90, trabalhei como editor na Network+ da ABC. Precisava de cassetes físicas para editar. Em breve, as cassetes de vídeo passaram a ser dados num disco rígido, o que abriu muitas novas técnicas de edição.

Da mesma forma, com a educação, não há razão para um horário obrigatório para aprender algo. O Async abre a possibilidade de muitas novas formas de aprender. A mãe pode estudar à noite; os empregados actuais podem aprender ao fim de semana ou durante as pausas para almoço.

Aprendizagem ao longo da vida: acesso permanente a conteúdos para os antigos alunos com um percurso de atualização contínua de competências

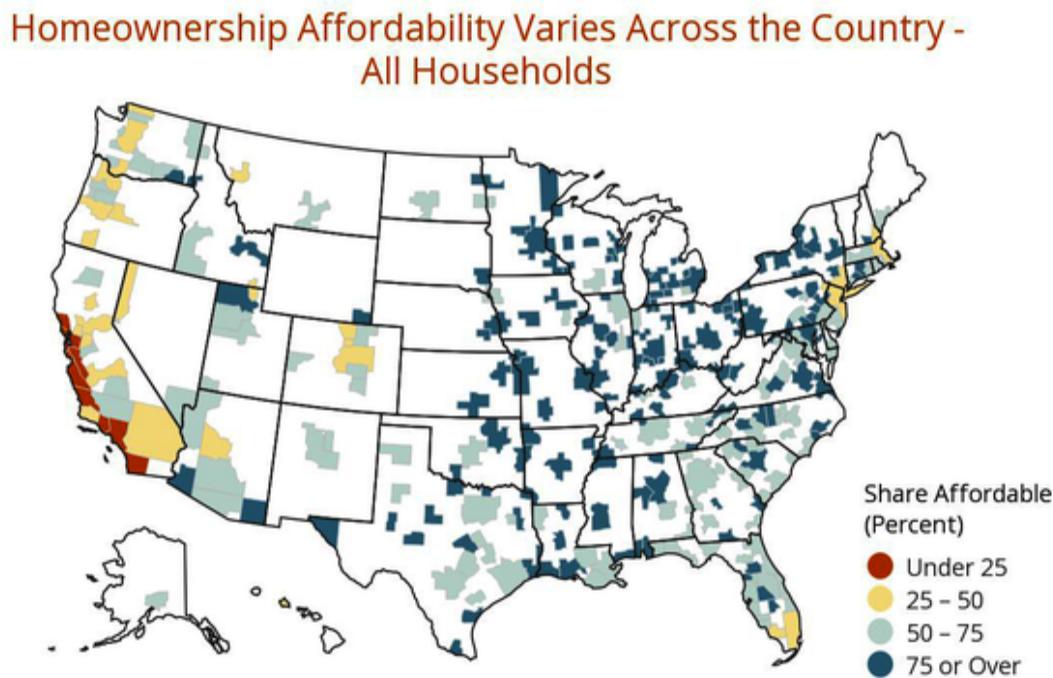
Outra razão pela qual as instituições de ensino devem repensar a opção "remote first" é porque permitiria a criação de cursos oferecidos aos antigos alunos (a custo zero ou a uma taxa SaaS). O SaaS pode servir como um método de proteção contra o ataque da concorrência. Muitos sectores exigem uma atualização constante das competências. Um bom exemplo é a indústria tecnológica.

É seguro dizer que qualquer trabalhador do sector da tecnologia precisa de aprender uma nova competência de seis em seis meses. O atual produto educativo não tem isso em conta. Porque não dar aos antigos alunos a

oportunidade de aprenderem o material e obterem uma certificação sobre esse material? A melhoria dos antigos alunos pode levar a uma marca ainda melhor.

Mercado de trabalho regional que sofrerá perturbações

Como antigo engenheiro de software da Bay Area e proprietário de uma casa, não vejo qualquer vantagem futura em viver na região com a atual estrutura de custos([Figura G-5](#)). O elevado custo de vida nas regiões de hiper Crescimento causa muitos problemas em cascata: sem-abrigo, aumento dos tempos de deslocação, redução drástica da qualidade de vida, etc.



Notes: Median incomes are estimated at the core-based statistical area (CBSA) level. Recently sold homes are defined as homes with owners that moved within the 12 months prior to the survey date. Monthly payments assume a 3.5% downpayment and property taxes of 1.15%, property insurance of 0.35%, and mortgage insurance of 0.85%. Affordable payments are defined as requiring less than 31% of monthly household income. Only CBSAs with at least 30 home sales in the past year are shown.
Source: JCHS tabulations of US Census Bureau, 2017 American Community Survey 1-Year Estimates, and Freddie Mac, PMMS.

Figura G-5. Acessibilidade da habitação nos EUA

Onde há uma crise, há uma oportunidade. Muitas empresas apercebem-se de que, seja qual for a vantagem de uma região de custos extremamente elevados, não vale a pena. Em vez disso, os centros regionais com excelentes infra-estruturas e baixo custo de vida têm uma enorme oportunidade. Algumas das características das regiões que podem atuar

como pólos de crescimento do emprego incluem o acesso a universidades, transportes, baixo preço da habitação e boas políticas governamentais para o crescimento.

Um excelente exemplo de uma região como esta é o Tennessee.

Tem **programas de licenciatura gratuitos** e acesso a muitas universidades de topo, um custo de vida baixo e institutos de pesquisa de topo como o **Oak Ridge National Lab**. Estas regiões podem perturbar drasticamente o status quo, especialmente se adoptarem a educação e a força de trabalho remotas e assíncronas.

Perturbação do processo de contratação

O processo de contratação nos Estados Unidos está pronto para ser interrompido. A **Figura G-6** mostra como seria possível perturbar a contratação eliminando todas as entrevistas e substituindo-as pelo recrutamento automático de indivíduos com certificações adequadas. Este é um problema clássico de programação distribuída que corrige um estrangulamento ao passar as tarefas de um fluxo de trabalho em série e com "bugs" para um fluxo de trabalho totalmente distribuído. As empresas devem "puxar" pelos trabalhadores em vez de puxar continuamente por um recurso preso num trabalho fútil.

Automated technical hiring process Asynchronous and distributed

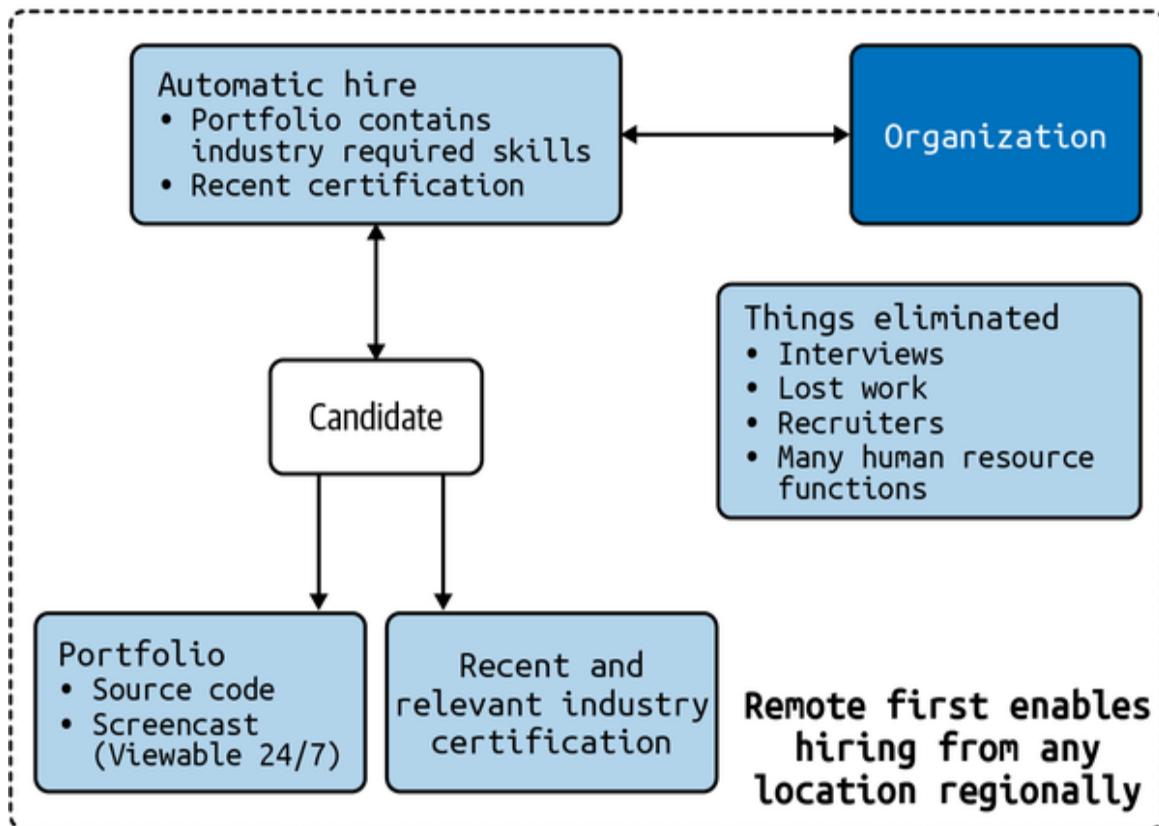


Figura G-6 : Interrompe a contratação

Uma das razões pelas quais a computação em Cloud é uma competência tão importante é o facto de simplificar muitos aspectos da engenharia de software. Uma das questões que simplifica é a construção de soluções. Muitas soluções envolvem frequentemente um ficheiro de formato YAML e um pouco de código Python.

Conclusão

A educação já não é estática. Para seres relevante no MLOps, precisas de ter uma mentalidade de crescimento. Esta mentalidade de crescimento significa que tens de aprender coisas de forma contínua enquanto produzes resultados. A boa notícia é que os motivados têm cada vez mais hipóteses de sucesso se aproveitarem a explosão de oportunidades de formação técnica.

1 Fonte: CALIFÓRNIA, SÃO FRANCISCO 1996 -TAXI MEDALLION SUPPLEMENTAL
LICENSEPLATE - Flickr - woody1778a.jpg.

Apêndice H. Gestão técnica do projeto

Por Noah Gift

Na primeira semana da aula altamente técnica sobre Cloud Computing, falei da gestão técnica de projectos e da importância de fazer o que descrevo neste apêndice. Um aluno levantou a mão na aula e fez a seguinte pergunta: "Esta é uma aula de gestão de projectos? Pensava que esta aula era sobre computação em Cloud?".

Da mesma forma, no mundo real, é fácil pensar que a gestão de projectos não é importante. No entanto, uma pequena quantidade de conhecimentos técnicos de gestão de projectos é fundamental para os MLOps. Os três componentes essenciais da gestão de projectos são os seguintes:

- Tenta planear o teu projeto em função do tempo que tens para o concluir, ou seja, 12 semanas com marcos semanais.
- Faz uma demonstração semanal à tua equipa mostrando os progressos.
- Divide as tarefas em blocos de 4 horas e executa-as semanalmente, acompanhando-as com um sistema simples de acompanhamento de tarefas como o Trello ou o GitHub.

Agora, vamos abordar apenas o básico para garantir o teu sucesso.

Plano do projeto

Não há nada de único nos MLOps relativamente à gestão de projectos. Ainda assim, vale a pena apontar alguns destaques não óbvios da gestão de projectos que ajudam a criar projectos de aprendizagem automática. Em particular, um conceito poderoso na construção de soluções de ML é pensar

em termos de um cronograma de 10 a 12 semanas e tentar mapear os resultados individuais de cada semana. No repositório de código-fonte do livro, há um **exemplo de modelo a ser seguido**.

Uma conclusão é que um plano cria os andaimes necessários para começar a implementar o código de aprendizagem automática na produção, e a demonstração semanal cria responsabilidade.

Na **Figura H-1**, um plano de projeto semana a semana permite definir o âmbito inicial da complexidade de um projeto.

Week	Milestone: High Level Goal	Weekly Demo Task (30 second video)
1	Kickoff	Kickoff (present plan)
2	Continuous Deployment	Continuous Deployment
3	Setup GCP	GCP Skeleton Project
4	Big Query Setup	Big Query Integration
5	Big Query Modeling and Prediction	Big Query Prediction
6	AutoML Setup	AutoML Prediction
7	Creation of Multiple Environments	Staging Environment
8	Integrate API	Computer Vision API
9	Stackdriver Setup	Monitoring
10	Finish MVP	MVP

Figura H-1. Plano do projeto

De seguida, vamos discutir a demonstração semanal.

Demonstração semanal

Em ambos os cenários do mundo real, quando trabalho com uma equipa a implementar código em produção ou em sistemas educativos, a demonstração semanal e o plano de projeto são componentes essenciais que reduzem o risco de um projeto nunca ver a luz do dia. Outro facto escondido sobre a realização de demonstrações é fazer com que a pessoa que faz a demonstração comprehenda o que está a fazer e comunique a informação.

O provérbio romano "Docendo discimus" afirma que "ao ensinar, aprendemos". Este provérbio também está relacionado com a pesquisa académica sobre a aprendizagem e é uma das abordagens de ensino mais eficazes que já vi na indústria e na sala de aula. As demonstrações em vídeo são poderosas tanto para o consumidor como para o criador.

Acompanhamento de tarefas

Podes ver um quadro de acompanhamento básico e público do Trello online. Na [Figura H-2](#), repara numa abordagem simples de apenas três colunas: A fazer, Em curso e Concluído. Cada tarefa demora cerca de quatro horas a ser concluída, uma vez que esta é uma boa unidade de trabalho para dividir as tarefas.

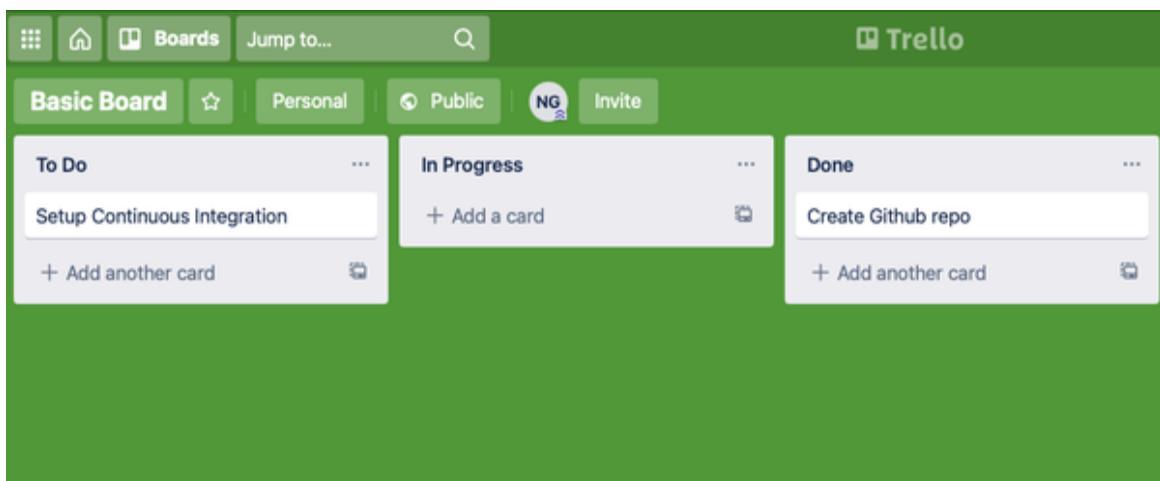


Figura H-2. Acompanhamento de bilhetes com o Trello

Por fim, cada um destes bilhetes é normalmente concluído todas as semanas. Este fluxo de trabalho mostra uma noção do progresso semanal.

Nota: podes usar qualquer sistema de acompanhamento baseado em quadros para fazer a mesma coisa. A principal lição é manter a simplicidade! Os sistemas de gestão de projectos simples mantêm-se, mas os complexos não.

Índice

A

ACI (Azure Container Instances), [Termos chave](#)
AKS (Serviço de Kubernetes do Azure), [Termos-chave](#)
alertas, definidos, [termos-chave](#)
Amazon ECR, definido, [Termos-chave](#)
Amazon EKS, definido, [Termos-chave](#)
Amazon Mechanical Turk, [Etiquetagem de dados da Mechanical Turk](#)
antipatterns, [Desafios Críticos em MLOps - Concentra-tena Precisão da Previsão Versus o Quadro Geral](#)
Serviços API, autenticação, Autenticação [de serviços API](#)
App Engine(ver Google App Engine)
App Runner, [AWS CaaS](#), [AWS App Runner](#) [Flask microsserviço](#)-[AWS App Runner](#) [Flask microsserviço](#)
app.py, [Livro de receitas do MLOps na AWS](#)
Apple
ML principal(ver ML principal)
Cria oML, [o AutoML da Apple](#): Criar ML-[Apple's AutoML](#): Criar ML ecossistema de desenvolvimento móvel, [Ecossistema da Apple](#)-Ferramentas de ML essenciais da Apple
[As melhores práticas combinadas](#) de ASICS, DevOps e MLOps?

autenticação

Serviços API, Autenticação de serviços API

autenticação de microsserviços em funções de nuvem, Autenticação em funções de nuvem-Autenticação em funções de nuvem

Azure, Autenticação-Autenticação deserviços API

com base em chaves, Autenticação de serviços API

responsável pelo serviço, responsável pelo serviço -responsável pelo serviço

cheques automatizados, cheques automatizados

Lei do automatizador, Fundamentos e blocos de construção da computação em Cloud, AutoML

AutoML, Fundamentos e blocos de construção da computação em Cloud, AutoML eKaizenML-Conclusão

Ecossistema da Apple e, Ecossistema daApple-Ferramentas principais de ML da Apple

AWS, AWS AutoML-AWSAutoML

AutoML do Azure, AutoML do Azure - AutoML do Azure

noções básicas, AutoML -Armazenamento de recursos

Cria ML, AutoML da Apple: Cria ML-Apple's AutoML: Cria ML ciência dos dados versus, Revolução Industrial dos MLOps

Armazenamento de recursos, Armazenamento de recursos - Armazenamento de recursos

FLAML, FLAML-FLAML

O AutoML da Google e a visão computacional de ponta, [Principais ferramentas de ML da Apple](#)-[As principais ferramentas de ML da Apple](#)

Kaizen versus KaizenML, [Kaizen Versus KaizenML](#)-[Kaizen Versus KaizenML](#)

Ludwig, [Ludwig](#)

Revolução industrial dos MLOps, [Revolução industrial dos MLOps](#) - [Revolução industrial dos MLOps](#)

Explicabilidade do modelo, [Explicabilidade do modelo](#) - [Explicabilidade do modelo](#)

soluções de código aberto, [Soluções AutoML de código aberto](#)-[FLAML](#)

escalonamento automático, definido, [Termos-chave](#)

AWS, [MLOps para AWS](#)-[Conclusão](#)

AutoML, [AWS AutoML](#)-[AWS AutoML](#)

Receitas do AWS Lambda, [Receitas do AWS Lambda](#)-[AWS Lambda](#)-[SAM Containerized Deploy](#)

noções básicas, [Introdução ao AWS-MLOps](#) no AWS

práticas recomendadas para inicializar os recursos do MLOps, [Conclusão](#)

CaaS, [AWS CaaS](#)-[AWS CaaS](#)

conselhos de carreira com o evangelista de ML da AWS Julien Simon, [Aplicar a aprendizagem automática da AWS ao mundo real](#)-[Aplicar a aprendizagem automática da AWS ao mundo real](#)

opções de certificação, [Certificações AWS](#) -[Implementação e operações de aprendizagem automática \(MLOps\)](#)

Certificação de especialista em aprendizagem automática, **Ascensão do engenheiro de aprendizagem automática e MLOps**

Ferramentas CLI para o projeto MLOps Cookbook, **FerramentasCLI-Ferramentas CLI**

visão por computador, **Visão porcomputador-Visão por computador**

Ferramentas de DataOps, **DataOps e engenharia de dados**

Microsserviço Flask,Microsserviço Flask-AWSApp Runner
Microsserviço Flask

Introdução a, **Introdução aos serviçosAWS-Visão computacional**

Sites estáticos do Hugo S3, **Usando sites estáticos do Hugo S3-Usandosites estáticos do Hugo S3**

Livro de receitas do MLOps em, **Livro de receitas do MLOps nomicrosserviço Flask do AWS-AWSApp Runner**

MLOps em, MLOps em AWS-MLOpsem AWS

Solução AWS Comprehend sem código/baixo código, **Utiliza a solução AWS Comprehend "Sem código/baixo código".**

aplicações ML do mundo real, **Aplicar a aprendizagem automática da AWS ao mundo real-Aplicara aprendizagem automática da AWS ao mundo real**

Livro de receitas sem servidor, **Livro de receitas sem servidor -Livro de receitas sem servidor**

Estudo de caso da Sports Social Network, **Aplicar o AWS Machine Learning ao mundo real - Aplicar oAWS Machine Learning ao mundo real**

AWS App Runner, **AWS CaaS, AWS App Runner Flask microsserviço-AWSApp Runner Flask microsserviço**

Certificação AWS Certified Machine Learning - Especialidade, Certificação AWS Certified Machine Learning - Especialidade -Implementação e Operações de MachineLearning (MLOps)

engenharia de dados, Engenharia dedados-Engenharia de dados
análise exploratória de dados, Análise Exploratória de Dados (EDA)-
Análise Exploratóriade Dados (EDA)
MLOps, Implementação e operações de aprendizagem automática
(MLOps)

AWS Cloud Practitioner, perguntas de teste de certificação para, AWS Cloud Practitioner e AWS Solutions Architect-AWSCloud Practitioner e AWS Solutions Architect

AWS Cloud9, Ambientes de desenvolvimento Cloud Shell, Utilização de sites S3 estáticos do Hugo, Termos-chave

AWS Cloudshell, Ambientes de desenvolvimento Cloud Shell -Ambientes de desenvolvimento Cloud Shell

AWS CloudWatch, observabilidade para MLOps em Cloud

AWS Comprehend, Utiliza a solução "No Code/Low Code" da AWS Comprehend

AWS DeepLens, Visão por computador -Visão por computador

AWS DeepRacer, conceitos-chave de aprendizagem automática

AWS Lambda, Curso rápido de Python

definido, Termos-chave

implantando com SAM, Receitas do AWS Lambda-AWSLambda-SAM Containerized Deploy

receitas, AWS Lambda Recipes-AWSLambda-SAM Containerized Deploy

Implantação em contêiner do SAM, [Implantação em contêiner do AWS Lambda-SAM](#) - [Implantação em contêiner do AWSLambda-SAM](#)

SAM Local, [AWS Lambda-SAM Local](#)

aplicações sem servidor, [Livro de receitas sem servidor-Livro de receitas sem servidor](#)

AWS S3

Sítios Web estáticos do Hugo S3, [Utilizar sítios Web estáticos do Hugo S3-Utilizarsítios Web estáticos do Hugo S3](#)

SageMaker e, [Monitorização da deriva com o AWS SageMaker-Monitorização daderiva com o AWS SageMaker](#)

AWS SageMaker(ver SageMaker)

Modelo de aplicação sem servidor da AWS(ver SAM)

Perguntas de teste de certificação do AWS Solutions Architect, [AWS Cloud Practitioner e AWS Solutions Architect-AWSCloud Practitioner e AWS Solutions Architect](#)

Azure, [MLOps paraAzure-Conclusão](#)

Application Insights, [Application Insights](#)

autenticar serviços de API, [Autenticar serviços de API](#)

autenticação, [Autenticação-Autenticação deServiços de API](#)

Azure CLI e Python SDK, [Azure CLI e Python SDK-AzureCLI e Python SDK](#)

Designer de aprendizagem automática do Azure, [Designer de aprendizagem automática do Azure-Designer de aprendizagem automática do Azure](#)

instâncias de computação, [Instâncias de computação -Instâncias de computação](#)

depurar localmente, [Depurar localmente-Depurarlocalmente](#)

implantação, [Implantação-Versão de conjuntos de dados](#)

implantando um modelo, [Implantando um modelo - Implantandoum modelo](#)

implantando modelos em um cluster de computação, [Implantando modelos em um cluster de computação - Implantandoum modelo](#)

Implementar o ONNX no, [Implementar o ONNX no Azure - Implementaro ONNX no Azure](#)

Certificação Microsoft: Certificação de cientista de dados do Azure, [Ascensão do engenheiro de aprendizagem automática e MLOps](#)

Ciclo de vida do ML, [Ciclo de vida do ML](#)

Pipelines de ML, [Pipelines de ML do Azure -Designer de Aprendizagem Automática do Azure](#)

MLOps para, [MLOps paraAzure-Conclusão](#)

publicar pipelines, [publicar pipelines](#)

registar modelos, [Registrar modelos-Registarmodelos](#)

recuperar registos, [Recuperar registos](#)

service principal, [Service Principal - ServicePrincipal](#)

resolução de problemas de implementação, [Resolução de problemas de implementação - depuraçãolocal](#)

versionamento de conjuntos de dados, [Versionamento de conjuntos de dados](#)

Certificação Azure AI Engineer Associate, Cientista de Dados do Azure e Engenheiro de IA

Certificação de Cientista de Dados Associado do Azure, Cientista de Dados do Azure e Engenheiro de IA

AutoMLdoAzure, AutoML do Azure - AutoMLdo Azure

Instâncias de contentores do Azure (ACI), termos-chave

Serviço de Kubernetes do Azure (AKS), termos-chave

ML do Azure, monitorizar a deriva com, Monitorizar a deriva com o ML do Azure-Monitorizar aderiva com o ML do Azure

Azure Percept, Azure Percept

Azure Pipelines, Criar Pipeline MLOps a partir do zero-Criar Pipeline MLOps a partir do zero

B

Baseball_Predictions_Export_Model.ipynb, Livro de receitas do MLOps no AWS

conjunto de dados de base, conjunto de dados de destino versus, Monitorização da deriva com o AWS SageMaker

Shell Bash, Shell Bash e Comandos - Escrever Script

configuração, Configuração

ficheiros e navegação, Ficheiros e navegação

entrada/saída, Entrada/Saída

ficheiros de lista, Ficheiros de lista

executar comandos, Executa comandos

escrever um script, Escrever um script

BigQuery(ver Google BigQuery)

montagem de fixação, portabilidade de modelos sem TPU
ferramenta preta (Python), **Termos chave**

Blake, William, **Arranja um emprego: Não invadas o castelo, entra pela porta das traseiras**

implantação blue-green, **implantação controlada de modelos**

Bogen, Joseph, **Origem das citações dos capítulos, Introdução ao MLOps, Fundamentos do MLOps, MLOps para Contentores e Dispositivos de Borda, Entrega Contínua para Modelos de Aprendizagem Automática, AutoML e KaizenML, Monitorização e Registo, MLOps para AWS, MLOps para Azure, MLOps para GCP, Interoperabilidade da Aprendizagem Automática, Criação de Ferramentas de Linha de Comando e Microsserviços MLOps, Engenharia de Aprendizagem Automática e Estudos de Caso MLOps**

servidor de compilação, definido, **Termos-chave**

C

CaaS(ver contentor como um serviço)

implantação **decanários, implantação controlada de modelos**

planeamento de carreira

estratégia de receitas da pera (PPEAR), **estratégia de receitas da pera - regrados 25%**

pensar como um capitalista de risco, **Pensa como um capitalista de risco para a tua carreira**

Estudos de casos, **Estudos de casos de engenharia de aprendizagem automática e MLOps -Conclusão**

Desafios críticos nos MLOps, Desafios críticos nos MLOps - Concentra-tena precisão da previsão versus o panorama geral

consequências éticas/não intencionais, Ethical and Unintended Consequences

Entrevista com Francesca Lazzeri, Concentra-te na precisão da previsão e no panorama geral

jejumintermitente, Estudo de caso de ciência de dados: Jejum intermitente - Notassobre jejum intermitente, glicemia e alimentação

Projectos MLOps como rede social desportiva Sqor, Projectos MLOps na rede social desportiva Sqor -Inteligência do atleta(produto de IA)

a técnica perfeita contra o mundo real, A técnica perfeita contra o mundo real - Atécnica perfeita contra o mundo real

Entrevista com Piero Molino, Concentra-te na precisão da previsão versus o panorama geral - Concentra-tena precisão da previsão versus o panorama geral

benefícios improváveis da ignorância na construção de modelos de ML, Benefícios improváveis da ignorância na construção de modelos de aprendizagem automática -Benefícios improváveis da ignorância na construção de modelos de aprendizagem automática

CD(ver entrega contínua (CD))

certificações, Certificações detecnologia-Certificações relacionadas com SQL

AWS Certified Machine Learning - Specialty, AWS Certified Machine Learning - Specialty-MachineLearning Implementation and Operations (MLOps)

AWS Cloud Practitioner, AWS Cloud Practitioner e AWS Solutions Architect - AWSCloud Practitioner e AWS Solutions Architect

Arquiteto de soluções da AWS, praticante da AWS Cloud arquiteto de soluções da AWS -praticante da AWSCloud e arquiteto de soluções da AWS

Cientista de dados/engenheiro de IA do Azure, Cientista de dados e engenheiro de IA do Azure

Certificaçõesrelacionadas com SQL, certificações relacionadas com SQL

desafios nos MLOps

consequências éticas/imprevistas, Consequências éticas e imprevistas

concentra-te na precisão da previsão versus o panorama geral,
concentra-te na precisão da previsão versus o panorama geral

falta de excelência operacional, Falta de excelência operacional

Charpentier, Emmanuelle, AutoML

CI/CD (integração contínua/entrega contínua)

implementação de, DevOps e MLOps

paralelos com a recuperação de lesões desportivas, Entrega Contínua para Modelos de Aprendizagem Automática

Pipeline CI/CD, verificações de controlo de qualidade para, Introdução ao Cloud Computing

CircleCI, definido, Termos-chave

Ferramentas CLI(ver ferramentas da interface da linha de comandos)

cli.py, livro de receitas do MLOps na AWS

computação em Cloud

fundações/construções, Cloud Computing Foundations and Building Blocks-CloudComputing Foundations and Building Blocks

Começar a utilizar, [Começar a utilizar a computação em nuvem - Começar a utilizar a computação em nuvem](#)

aprendizagem automática e, [Rise of the Machine Learning Engineer e MLOps](#)

MLOps em nuvem, observabilidade para, [Observabilidade para MLOps em nuvem](#)

aplicações nativas da cloud, definido, [Termos chave pipelines Cloud](#)

entrega contínua, [Usando Cloud Pipelines - Melhoria contínua](#)

lançamento controlado de modelos, [Lançamento controlado de modelos](#)

técnicas de ensaio para a implantação de modelos, [Técnicas de ensaio para a implantação de modelos - Melhoria contínua](#)

Cloud Run, [visão geral do Google Cloud Platform](#)

ambientes de desenvolvimento em cloud shell, [Ambientes de desenvolvimento em cloud shell - Ambientes de desenvolvimento em cloud shell](#)

Cloud9(ver AWS Cloud9)

Cloudshell, [Ambientes de desenvolvimento Cloud Shell - Ambientes de desenvolvimento Cloud Shell](#)

agrupamento

configurando, [Configurando um Cluster-Configurando um Cluster](#)

implantação de modelos em um cluster de computação, [Implantação de modelos em um cluster de computação - Implantação de um modelo](#)

inferência versus computação, [Configurar um cluster](#)

ferramentas de interface de linha de comando (CLI), **Construir MLOps**
Ferramentas de linha de comando em microserviços-Conclusão

noções básicas, **Ferramentas de linha de comando-Modularizar uma ferramenta de linha de comando**

criar uma CLI baseada na nuvem, **Criar uma CLI baseada na nuvem**

criar um linter de conjunto de dados, **Criar um linter de conjunto de dados-Criar um linter de conjunto de dados**

fluxos de trabalho CLI de aprendizagem automática, **Fluxos de trabalho CLI de aprendizagem automática-Conclusão**

Projeto MLOps Cookbook, **Ferramentas CLI-Ferramentas CLI**

modularizing, **Modularizar uma ferramenta de linha de comando-Modularizar uma ferramenta de linha de comando**

Empacotamento Python, Empacotamento Python

ficheiro de requisitos, **O Ficheiro de Requisitos**

linha de comandos, Linux, **Bash e a linha de comandos Linux**

Vulnerabilidades e Exposições Comuns (CVEs), **Melhores Práticas**

vantagem competitiva, **utilizando a solução "No Code/Low Code" AWS Comprehend**

cluster de computação

implantando modelos para, **Implantando modelos em um cluster de computação-Desdobrando um modelo**

cluster de inferência versus, **Configurando um cluster**

Visão geral do Compute Engine, Google Cloud Platform

instâncias de computação, Azure, **Instâncias de computação - instâncias de computação**

visão computacional

AWS, Visão por computador -Visão por computador

AutoML da Google e visão computacional de ponta, ferramentas de ML essenciais da Apple -ferramentas de ML essenciais da Apple

configuração

Bash shell, Configuração

cluster, Configurando um Cluster-Configurando um Cluster

configurar a integração contínua com GitHub Actions, Configurar a integração contínua com GitHub Actions-Configurara integração contínua com GitHub Actions

tempo de execução, Azure ML Pipelines

contentor como um serviço (CaaS)

AWS, AWS CaaS-AWSaaS

Padrão de conceção MLOps, Padrões de conceção MLOps

fluxo de trabalho em contentor, AWS Lambda-SAM Containerized Deploy-AWSLambda-SAM Containerized Deploy

contentores, Contentores - Servirum modelo treinado através de HTTP

melhores práticas, Melhores Práticas-MelhoresPráticas

Constrói uma vez, executa muitos fluxos de trabalho MLOps, Constrói uma vez, executa muitos fluxos de trabalho MLOps

criando, Criando um contêiner-Criando um contêiner

definido, Termos chave

sistemas ML geridos e, Contentores para sistemas ML geridos-Construiruma vez, Executar muitos fluxos de trabalho MLOps

monetizando MLOps, [Containers em Monetizando MLOps](#)
executando, [Executando um Container-Executando um Container](#)
tempo de execução, [Tempo de execução do contentor](#)
servindo um modelo treinado [por HTTP](#), [Servindo um modelo treinado por HTTP-Servindo um modelo treinado por HTTP](#)

Entrega contínua (CD), [DevOps e MLOps](#), Entrega contínua para
modelos de aprendizagem automática-Conclusão

(ver também CI/CD)

pipelines de nuvem para, [Usando pipelines de nuvem -Melhoria contínua](#)

definido, [Termos-chave](#)

Melhores práticas de DevOps, [DevOps e MLOps](#)

GCP e, [Integração contínua e entrega contínua -Integração contínua e entrega contínua](#)

Infraestrutura como código para entrega contínua de modelos de ML,
Infraestrutura como código para entrega contínua de modelos de ML-
Infraestrutura como código para entrega contínua de modelos de ML

empacotamento de modelos de ML, [Empacotamento de modelos de ML - Empacotamento de modelos de ML](#)

técnicas de ensaio para a implantação de modelos, [Técnicas de ensaio para a implantação de modelos -Melhoria contínua](#)

melhoria contínua, [Implementar DevOps](#), Melhoria contínua

integração contínua (CI), [DevOps e MLOps](#)

(ver também CI/CD)

configurando com GitHub Actions, **Configurando a integração contínua com GitHub Actions**-Configurando a integração contínua com GitHub Actions

definido, **Termos-chave**

Melhores práticas de DevOps, **DevOps e MLOps**

GCP e, **Integração contínua e entrega contínua** -Integração contínua e entrega contínua

convergência, **Otimização**

Projeto Coral, **Coral-Coral**

Core ML, **Ecossistema da Apple**, Ferramentas Core ML da Apple - Ferramentas Core ML da Apple, Core ML da Apple -Core ML da Apple

Cria oML, o **AutoML** da Apple: Criar ML-Apple's AutoML: Criar ML educação centrada **no cliente**, **Foco no cliente**

CVEs (Vulnerabilidades e Exposições Comuns), **Melhores Práticas**

cibersegurança, **Governação de dados e cibersegurança**

D

desvio de dados(ver desvio)

engenharia de dados

AWS Certified Machine Learning - Certificação de especialidade, **Engenharia dedados-Engenharia de dados**

definido, **Termos-chave**

GCP, **DataOps no GCP**: Engenharia de dados aplicada-DataOps no GCP: Engenharia de dados aplicada

Hierarquia das necessidades de ML e, **DataOps e Engenharia de Dados**

governação de dados, governação de dados e cibersegurança, engenharia de dados

lago de dados, DataOps e engenharia de dados, Engenharia de dados ciência dos dados

AutoML versus, Revolução Industrial MLOps

como competência fundamental, Doing Data Science-DoingData Science

armazéns de dados, Feature Stores versus, Feature Stores

DataOps

GCP, DataOps no GCP: Engenharia de dados aplicada-DataOpsno GCP: Engenharia de dados aplicada

Hierarquia das necessidades de ML e, DataOps e Engenharia de Dados

dataset linter, Criar um dataset linter-Criarum dataset linter

conjuntos de dados

linha de base versus objetivo, Monitorização da deriva com o AWS SageMaker

versionamento, Versionamento de conjuntos de dados

Davis, Purnell, Etiquetagem de dados do Mechanical Turk

depuração(ver resolução de problemas)

intuição de Deep Learning, Otimização-Otimização

DeepLens, Visão por computador -Visão por computador

dependências

definir no ficheiro requirements.txt, O Ficheiro de Requisitos

pinning, Melhores práticas

Embalagem Python e, Embalagem Python

implantação

Azure, Implantação-Versão de conjuntos de dados

verde-azul versus canário, Lançamento controlado de modelos

Implantação de um modelo, Implantação de um modelo-
Desdobramento de um modelo

As melhores práticas combinadas de MLOps, DevOps e MLOps?

registro, Registo de modelos-Registode modelos

resolução de problemas de implementação, resolução de problemas de
implementação - depuraçãolocal

estatística descritiva, Estatística Descritiva e Distribuições Normais -
Estatística Descritivae Distribuições Normais, Estatística Descritiva e
Distribuições Normais

padrões de conceção, MLOps Design Patterns

ambientes de desenvolvimento, Ambientes de desenvolvimento CloudShell-
Ambientes de desenvolvimento Cloud Shell

DevOps

melhores práticas, DevOps e MLOps

definido, O que é MLOps?

implementando, Implementando DevOps-Implementando DevOps

MLOps e, DevOps e MLOps-DevOpse MLOps

O ciclo de feedback do MLOps, as melhores práticas combinadas do
DevOps e do MLOps?

dieta, jejum intermitente e, **Estudo de caso de ciência de dados: Jejum intermitente - Notassobre o jejum intermitente, a glicemia e a alimentação**
recuperação de desastres, definido, **Termos-chave**
perturbação(ver perturbação do ensino)

Docker

contentores, **tempo de execução do contentor**
definido, **Termos-chave**

Dockerfile, **Criar um contentor**, **Livro de receitas do MLOps na AWS**
formato do contentor, **termos-chave**

Doudna, Jennifer, **AutoML**

deriva

monitorização com o AWS SageMaker, **Monitorização da deriva com o AWS SageMaker**-Monitorização daderiva com o AWS SageMaker
monitorização com o Azure ML, **Monitorizar a deriva com o Azure ML**-**Monitorizar aderiva com o Azure ML**

E

EDA (análise exploratória de dados), **Estatística descritiva e distribuições normais**, **Análise exploratória de dados (EDA) -Análise exploratória de dados (EDA)**

Dispositivos periféricos, **Dispositivos periféricos - Transportaçãopara modelos sem TPU**

Ecossistema da Apple, **Ecossistema da Apple -Principais ferramentas de ML da Apple**

ASICS e, **DevOps e MLOps combinam as melhores práticas?**

Perceção do Azure, Perceção do Azure

Coral, Coral-Coral

Formato de modelo ORT e, Integração de borda -Integração de borda portabilidade de modelos não-TPU, Portabilidade de modelos não-TPU-Portabilidade de modelos não-TPU

TFHub, TFHub

perturbação do ensino, Perturbação do ensino - Perturbação do processo de contratação

estado atual do ensino superior que vai ser perturbado, Estado atual do ensino superior que vai ser perturbado-Estado atual do ensino superior que vai ser perturbado

10X better education, 10X Better Education-Disrupção do processo de contratação

recursos educativos, Additional Educational Resources-Conclusão (Recursos educativos adicionais-Conclusão)

perguntas adicionais sobre o pensamento crítico dos MLOps, Perguntas adicionais sobre o pensamento crítico dos MLOps - Perguntas adicionais sobre o pensamento crítico dos MLOps

materiais educativos adicionais dos MLOps, Materiais educativos adicionais dos MLOps

perturbação do ensino, Perturbação do ensino - Perturbação do processo de contratação

Aplicação Flask Elastic Beanstalk, MLOps no AWS-MLOps no AWS

ELI5, Explicabilidade do modelo, Explicabilidade do modelo

engenharia, ciência versus, Rise of the Machine Learning Engineer e MLOps

Explicabilidade, AutoML do Azure, Explicabilidade do modelo -
Explicabilidade do modelo

análise exploratória de dados (AED), Estatística descritiva e distribuições normais, Análise exploratória de dados (AED) -Análise exploratória de dados (AED)

F

FaaS(ver função como um serviço)

Faber, Issac, Concentra-te na exatidão da previsão em relação ao panorama geral

Fargate, AWS CaaS-AWSaaS

jejum, intermitente, Estudo de caso de ciência de dados: Jejum intermitente - Notassobre o jejum intermitente, a glicemia e a alimentação

Lojas de recursos, Lojas de recursos -Lojas de recursos

o ciclo de feedback, as melhores práticas combinadas de DevOps e MLOps?

FLAML, FLAML-FLAML

Microsserviço Flask,Microsserviço Flask-AWSApp Runner Microsserviço Flask

construir automaticamente o contentor através do GitHub Actions, construir automaticamente o contentor através do GitHub Actions e enviar para o GitHub Container Registry

AWS App Runner e, AWS App Runner Flask microservice-AWSApp Runner Flask microservice

constrói um exemplo, AWS CaaS-AWSaaS

contentorizado, Microsserviço Flask contentorizado

comida, jejum intermitente e, **Estudo de caso de ciência de dados: Jejum intermitente - Notassobre o jejum intermitente, a glicemia e a alimentação**

Competências básicas para MLOps, Fundamentos de MLOps -Conclusão

Shell e comandos do Bash, Shell e comandos do Bash - Escrevendoum script

Constróium pipeline de MLOps a partir do zero, Constróium pipelinede M LOpsa partir do zero

fundamentos/blocos de construção da computação em nuvem, Fundamentos eblocosde construção da computação em nuvem - Fundamentos e blocos de construção da computação em nuvem

ambientes de desenvolvimento em cloud shell, Ambientes de desenvolvimento em cloud shell - Ambientesde desenvolvimento em cloudshell

Fazendo ciéncia de dados, Fazendo ciéncia de dados - Fazendociéncia de dados

Começar a utilizar a computação em nuvem, Começar a utilizar a computação em nuvem - Começara utilizar a computação em nuvem

Linha de comando do Linux, Bash and the Linux Command Line

conceitos-chave de aprendizagem automática, Conceitos-chave de aprendizagemautomática-Conceitos-chave de aprendizagem automática

curso rápido de matemática para programadores, Curso rápido de matemática paraprogramadores-Otimização

Python (curso intensivo), Python Crash Course-PythonCrash Course

Python (tutorial), Tutorial de Python Minimalista

Fox, Justin, AutoML

função como um serviço (FaaS)

AWS Lambda, curso rápido de Python
definido, Termos-chave

G

GCP(ver Google Cloud Platform)

Gilovich, Thomas, AutoML

Ações do GitHub

construir automaticamente o contentor através de, Construir
automaticamente o contentor através de GitHub Actions e enviar para
GitHub Container Registry

configurar a integração contínua com, Configurar a integração
contínua com GitHub Actions-Configurara integração contínua com
GitHub Actions

GCP versus, Integração Contínua e Entrega Contínua

Registo de contentores do GitHub, enviar o contentor para, Criar
automaticamente o contentor através de GitHub Actions e enviar para o
Registo de contentores do GitHub

GitHub, limitações de tamanho de ficheiro para, Infraestrutura como
Código para Entrega Contínua de Modelos ML

GKE (Google Kubernetes Engine), Visão geral do Google Cloud Platform,
Termos principais

Google

AutoML e visão computacional de ponta, Ferramentas principais de
ML da Apple-Apple's Core ML Tools

Certificação de Engenheiro Profissional de Aprendizagem Automática, Ascensão do Engenheiro de Aprendizagem Automática e MLOps

Google App Engine, Visão geral do Google Cloud Platform, Operacionalização de modelos de ML

Google BigQuery, Visão geral do Google Cloud Platform, Escolha e design da base de dados nativa da Cloud -Escolha e design da base de dados nativa da Cloud

Google Cloud Build, Integração Contínua e Entrega Contínua

Funções do Google Cloud, DataOps no GCP: Engenharia de dados aplicada - DataOpsno GCP: Engenharia de dados aplicada

Google Cloud Platform (GCP), MLOps paraGCP-Conclusão

vantagens de usar, Visão geral do Google Cloud Platform

engenharia de dados aplicada no, DataOps no GCP: engenharia de dados aplicada-DataOpsno GCP: engenharia de dados aplicada

certificações, GCP

escolha/design de bases de dados nativas da nuvem, Escolha e design de bases de dados nativas da nuvem -Escolha e design de bases de dados nativas da nuvem

integração contínua/entrega contínua, Integração contínua e entrega contínua -Integração contínuae entrega contínua

desvantagens de usar, Visão geral da Google Cloud Platform -Visão geral da GoogleCloud Platform

Kubernetes e, Kubernetes Hello World-KubernetesHello World

componentes principais, Visão geral da Google Cloud Platform

operacionalização de modelos de ML, Operacionalização de modelos de ML-Operacionalização demodelos de ML

visão geral, Visão geral do Google Cloud Platform -Escolha e design de banco de dados nativo da Cloud

Google Cloud Run, Visão geral do Google Cloud Platform

Google Kubernetes Engine (GKE), visão geral do Google Cloud Platform, principais termos

algoritmos gulosos, Otimização-Otimização

H

Haber, Jonathan, A técnica perfeita contra o mundo real

Hardgrove, Jacob, A técnica perfeita contra o mundo real

Hargadon, Andrew, Conclusão

Harris, Tristan, Ética e consequências não intencionais

problemas de saúde, área de trabalho em casa, Saúde e área de trabalho

hierarquia de necessidades, ML, Uma hierarquia de necessidades de MLOps -melhores práticas combinadas de DevOpse MLOps?

configurar a integração contínua com o GitHub Actions, Configurar a integração contínua com o GitHub Actions-Configurara integração contínua com o GitHub Actions

DataOps e engenharia de dados, DataOps e engenharia de dados

implementando DevOps, Implementando DevOps-
ImplementandoDevOps

MLOps no topo de, MLOps-DevOpse MLOps combinaram as melhores práticas?

automação da plataforma, Automação da plataforma

processo de contratação, perturbação do, **Perturbação do processo de contratação**

passatempos, **Arranja um emprego: Não invadas o castelo, entra pela porta das traseiras**

Autoscaler de pod horizontal (HPA), **Kubernetes Olá mundo**

acessibilidade da habitação, **mercado de trabalho regional que será perturbado**

HTTP, servir um modelo treinado através de, **Servir um modelo treinado através de HTTP-Servirum modelo treinado através de HTTP**

htwtmlb.csv, **Livro de receitas do MLOps na AWS**

Sites do Hugo, **Usando sites estáticos do Hugo S3-Usando sites estáticos do Hugo S3**

I

IaC(ver infraestrutura como código)

ignorância, benefícios da, **Benefícios improváveis da ignorância na construção de modelos de aprendizagem automática -Benefícios improváveis da ignorância na construção de modelos de aprendizagem automática**

implementação de MLOps

governação de dados e cibersegurança, **Governação de dados e cibersegurança**

padrões de conceção, **MLOps Design Patterns**

recomendações globais, **Recomendações finais para a implementação dos MLOps**

recomendações para, Recomendações finais para implementar os padrões de conceção MLOps-MLOps

clusters de inferência, Configurar um cluster

infraestrutura como código (IaC)

entrega contínua de modelos de aprendizagem automática e, Infrastructure as Code for Continuous Delivery of ML Models- Infrastructureas Code for Continuous Delivery of ML Models

As melhores práticas de DevOps, DevOps e MLOps

entrada/saída

Bash shell, Entrada/Saída

instruções (palavras-chave do contentor), Criar um contentor

instrumentação (melhores práticas DevOps), DevOps e MLOps

jejumintermitente, Estudo de caso de ciência de dados: Jejum intermitente - Notassobre o jejum intermitente, a glicemia e a alimentação

interoperabilidade, interoperabilidade da aprendizagemautomática- Conclusão

Apple Core ML, Apple Core ML-AppleCore ML

integração dos limites, Integração dos limites -Integração dos limites

importância da, Porque é que a interoperabilidade é fundamental- Porqueé quea interoperabilidade é fundamental

ONNX, ONNX: Open Neural Network Exchange - Implementao ONNX no Azure

Intérprete IPython, definido, Termos-chave

Isaacson, Walter, AutoML

J

James, Lebron, Influencer Rank

mercados de trabalho, regional, Mercado de trabalho regional que será perturbado

JSON, definido, Termos-chave

K

Kaggle, MLOps em AWS

Kaizen, o que é MLOps, Kaizen versus KaizenML-Kaizenversus KaizenML

KaizenML, Kaizen Versus KaizenML-KaizenVersus KaizenML, Armazenamento de recursos-Armazenamentode recursos

gráfico de densidade de kernel, estatísticas descritivas e distribuições normais

termos-chave,Termos-chave-Termos-chave

autenticação baseada em chaves, autenticação de serviços API

Koonin, Steven, Estatística Descritiva e Distribuições Normais, Concentrante na Precisão das Previsões Versus o Quadro Geral

Kubernetes, Kubernetes Hello World-KubernetesHello World

implantação blue-green, implantação controlada de modelos

definido, Termos-chave

Autoscaler de pod horizontal, Kubernetes Hello World

Clusters Kubernetes, Termos chave

Contentores Kubernetes, Termos chave

Pods Kubernetes, [Termos chave](#)

Design centrado em Kubernetes, [padrões de design MLOps](#)

L

Lazzeri, Francesca, [Concentra-te na precisão da previsão e não no panorama geral](#)

aprendizagem ao longo [da vida](#), aprendizagem ao longo da vida: acesso permanente aos conteúdos para os antigos alunos com um percurso de atualização contínua das competências

ciclo de vida, Azure e, [Ciclo de vida ML](#)

linter

contentores e, [Melhores práticas](#)

criar um conjunto de dados linter, [Criar um conjunto de dados linter](#)-
[Criar um conjunto de dados linter](#)

modularizando, [Modularizando uma ferramenta de linha de comando](#)-
[Modularizando uma ferramenta de linha de comando](#)

linting, [Introdução à computação em Cloud](#), [Linting](#)

Linha de comandos do Linux, [Bash](#) e a linha de comandos do Linux

teste de carga, definido, [Termos-chave](#)

Gafanhoto, definido, [Termos-chave](#)

registo, [Introdução ao registo](#)

(ver também monitorização)

noções básicas, [Introdução ao registo](#)

definido, [Termos-chave](#)

modificar níveis de registo, [Modificar níveis de registo](#)

Python, [Registo em Python-Loggingem diferentes aplicações](#)

recuperar registos, [Recuperar registos](#)

Loranger, Rob, [Aplicar a aprendizagem automática da AWS ao mundo real](#)
[Aplicara aprendizagem automática da AWS ao mundo real](#)

Loukides, Mike, [Conclusão](#)

Comando ls, [Lista ficheiros](#)

Ludwig AutoML, [Ludwig](#)

M

aprendizagem automática (em geral)

computação em cloud e, [Ascensão do engenheiro de aprendizagem automática e MLOps](#)

interoperabilidade(ver interoperabilidade)

conceitos-chave, [Conceitos-chave de aprendizagem automática - Conceitos-chavede aprendizagem automática](#)

engenharia de aprendizagem automática, ferramentas e processos utilizados,
[Rise of the Machine Learning Engineer e MLOps](#)

pensamento mágico, [AutoML](#)

Makefile, [Implementação de DevOps, Livro de receitas do MLOps na AWS](#)

definido, [Termos-chave](#)

razões para usar, [Implementação do DevOps](#)

sistemas ML geridos, contentores para, [Contentores para sistemas ML geridos - Constróiuma vez, executa muitos fluxos de trabalho MLOps](#)

A hierarquia das necessidades de Maslow, **Uma hierarquia das necessidades de MLOps**

matemática para programadores

curso intensivo, **Curso intensivo de matemática paraprogramadores-Otimização**

estatística descritiva/distribuições normais, **Estatística descritiva e distribuições normais -Estatística descritivae distribuições normais**

otimização, **Otimização-Otimização**

Mechanical Turk, **Etiquetagem de dados da Mechanical Turk**

métricas

definido, **Termos-chave**

para a monitorização de modelos, **Noções básicas de monitorização de modelos**

microsserviços, **Microsserviços - Construindo uma CLI baseada em Cloud**

autenticando em funções de nuvem, **Autenticando em funções de nuvem - Autenticando em funções de nuvem**

construindo uma CLI baseada na nuvem, **Construindo uma CLI baseada na nuvem**

criando uma função sem servidor, **Criando uma função sem servidor - Criando uma função sem servidor**

definido, **Contêineres, Termos-chave**

como prática recomendada de DevOps, **DevOps e MLOps**

Microsoft(ver entradas Azure)

migrar (termo), **Termos-chave**

mentalidade de erro, **Arranja um emprego: Não invadas o castelo, entra pela porta das traseiras**

Engenharia ML(ver hierarquia das necessidades, ML)

mlib.py, **Livro de receitas do MLOps na AWS**

MLOps (em geral)

noções básicas, **Introdução aos MLOps-Conclusão**

definido, **O que é MLOps?**

Como é que podes combinar as melhores práticas de implementação, **DevOps e MLOps?**

DevOpse, DevOps e MLOps-DevOpse MLOps

o ciclo de feedback, **as melhores práticas combinadas de DevOps e MLOps?**

competências de base para(ver competências de base)

recomendações globais, **Recomendações finais para a implementação dos MLOps**

hierarquia de necessidades, **Uma hierarquia de necessidades de MLOps -as melhores práticas combinadas de DevOpse MLOps?**

Kaizen versus KaizenML, **Kaizen versus KaizenML-Kaizenversus KaizenML**

recomendações para a implementação, **Recomendações finais para implementar os padrões de conceção MLOps-MLOps**

ascensão dos engenheiros de aprendizagem automática e dos MLOps, **Rise of the Machine Learning Engineer and MLOps**

portfólio técnico para, **Construir um portfólio técnico para MLOps - Conseguiram emprego: Não invadas o castelo, entra pela porta das**

traseiras

no topo da hierarquia de necessidades do ML, as melhores práticas combinadas MLOps-DevOps e MLOps?

Livro de receitas MLOps

Livro de receitas do AWS e do MLOps nomicrosserviço Flask do AWSApp Runner

Ferramentas CLI, Ferramentas CLI-Ferramentas CLI

Flaskmicrosserviço, Flask microsserviço-AWSApp Runner Flask microsserviço

MLOps revolução industrial, MLOps revolução industrial-
MLOpsrevolução industrial

Plataforma MLOps, MLOps Design Patterns

telemóveis(ver dispositivos periféricos)

implantação de um modelo, Implantação de um modelo-Deploying a Model

explicabilidade do modelo, AutoML do Azure, Explicabilidade do modelo-
ModelExplainability

ciclo de vida do modelo, ciclo de vida do ML

Zoo modelo, ONNX Zoo modelo-ONNXZoo modelo

model.joblib, Livro de receitas do MLOps na AWS

Molino, Piero, Focus on Prediction Accuracy Versus the Big Picture-
Focus on Prediction Accuracy Versus the Big Picture

monitorização, DevOps e MLOps, Monitorização e registo- Conclusão

(ver também registo)

noções básicas, Monitorização e registo

noções básicas de monitorização de modelos, Noções básicas de monitorização de modelos-Boasicsof Model Monitoring como melhor prática de DevOps, DevOps e MLOps métricas para, Noções básicas de monitoramento de modelos monitorizara deriva com o AWS SageMaker, Monitorizar a deriva com o AWS SageMaker-Monitorizara deriva com o AWS SageMaker monitorizara deriva com o Azure ML, Monitorizar a deriva com o Azure ML-Monitorizara deriva com o Azure ML observabilidade e, Monitorização e Observabilidade-Monitorização doDrift com o Azure ML observabilidade para MLOps na nuvem, Observabilidade para MLOps na nuvem

Lei de Moore, definida, Termos-chave

N

processamento de linguagem natural (PNL), utilizando a solução "No Code/Low Code" AWS Comprehend Network+ rede doméstica física, Rede doméstica física gestão de energia para redes domésticas, Gestão de energia e redes domésticas para trabalhares remotamente, Network+ NLP (processamento de linguagem natural), utilizando a solução "No Code/Low Code" AWS Comprehend

distribuições normais, Estatística descritiva e distribuições normais - Estatística descritivae distribuições normais

O

O'Reilly, Tim, **Conclusão**

observabilidade

Informações sobre aplicações, **Informações sobre aplicações**

monitorização e, **Monitorização e Observabilidade-Monitorizar aderiva com o Azure ML**

ONNX (Open Neural Network Exchange), **ONNX: Open Neural Network Exchange - Implementao ONNX no Azure**

converter modelos Core ML em, **Apple Core ML-AppleCore ML**

converter PyTorch em, **Converter PyTorch em ONNX-ConvertPyTorch em ONNX**

converter TensorFlow em, **Converter TensorFlow em ONNX-ConverterTensorFlow em ONNX**

criar um verificador ONNX genérico, **Create a Generic ONNX Checker-Criarum verificador ONNX genérico**

implantando no Azure, **Implantar ONNX no Azure - ImplantaONNX no Azure**

integração de borda com ORT, **Integração de borda -Integração de borda**

Zoo de modelos, **Zoo de modelos ONNX -Zoo de modelos ONNX**

empacotamento de modelos de ML, **Empacotamento de modelos de ML-Empacotamentode modelos de ML**

soluções AutoML de fonte aberta, **soluções AutoML de fonte aberta-FLAML**

FLAML, FLAML-FLAML

Ludwig, Ludwig

operacionalização, definido, Termos-chave

otimização, Otimização-Otimização

P

embalagem, Embalagem para modelos ML - Embalagem para modelos ML

Pandas, estatísticas descritivas e distribuições normais

estratégia de receitas da pera (PPEAR), Estratégia de receitas da pera - Regrados 25%

autonomia, Autonomia

potencial exponencial dos projectos, Exponential

rendimento passivo, Passive

regra dos 25%, Regra dos 25%

trabalho como experiência positiva, Positivo

pip, definido, Termos-chave

pipelines, Pipelines de publicação

(ver também pipelines Cloud)

Pipelines de ML do Azure, Pipelines de ML do Azure -Designer de Aprendizagem Automática do Azure

criar um pipeline de MLOps a partir do zero, Criar um pipeline de MLOps a partir do zero-Criar um pipeline de MLOps a partir do zero

publicação, Publicação de Pipelines

automatização da plataforma, automatização da plataforma

portos, definidos, termos-chave

PPEAR(ver estratégia de receitas de peras)

Arquiteto profissional de Cloud (certificação GCP), **GCP**

Engenheiro profissional de aprendizagem automática (certificação GCP),
GCP

gestão de projectos(ver gestão técnica de projectos)

plano de projeto (em gestão técnica de projectos), **Plano de Projeto**

Prometeu, definido, **termos-chave**

pylint, definido, **Termos-chave**

PyPI, definido, **Termos-chave**

pytest, definido, **Termos-chave**

Python

CLI do Azure e Python SDK, **CLI do Azure e Python SDK-AzureCLI e Python SDK**

ferramentas de linha de comandos, **Ferramentas de linha de comandos- Modularizaruma ferramenta de linha de comandos**

curso intensivo, **Cursointensivo de Python-Cursointensivo de Python**

ineficiência energética de, **Python Crash Course**

registar aplicações diferentes, **Registar aplicações diferentes - Registar aplicaçõesdiferentes**

registar, **registar em Python - registardiferentes aplicações**

estrutura do projeto de aprendizagem automática, **Implementar DevOps-ImplementarDevOps**

tutorial minimalista, **Tutorial Python minimalista**

Livro de receitas do MLOps, Livro de receitas do MLOps no AWS-AWSApp Runner Microsserviço Flask

modificar os níveis de registo, Modificar os níveis de registo

andaime de projeto, Implementação de DevOps

ficheiro de requisitos, O Ficheiro de Requisitos

lentidão de, Python Crash Course

testar/linting código, Introdução à computação em Cloud

Funções Python, Tutorial Python minimalista, Livro de receitas sem servidor

Python SDK, Azure CLI e Python SDK-AzureCLI e Python SDK

Ambiente virtual Python, definição, termos-chave

PyTorch, converter em ONNX, Converter PyTorch em ONNX-ConvertePyTorch em ONNX

R

Red Hat, Tempo de execução do contêiner

mercados de trabalho regionais, Mercado de trabalho regional que será perturbado

registro, Azure, Registo de modelos-Registo demodelos

registos, contentores

aprendizagem por reforço, conceitos-chave da aprendizagem automática

Educação "remote first", Async e remote first

trabalho remoto, Trabalho remoto - Localização, Localização, Localização

equipamento para, Network+espaço de trabalho em casaconfiguração de estúdio virtual

problemas de saúde, Saúde e área de trabalho

área de trabalho em casa, Área de trabalho em casa

configuração do estúdio virtual do espaço de trabalho em casa, Configuração do estúdio virtual do espaço de trabalho em casa

localização, Localização, Localização, Localização

rede, Network+

requirements.txt, Livro de receitas do MLOps na AWS, O ficheiro de requisitos

ficheiro requirements.txt, Implementar DevOps, O Ficheiro de Requisitos

Ridley, Matt, MLOps Revolução Industrial, Conclusão

regra dos 25% (rendimento), Regra dos 25%

regra dos 25% (MLOps), Conclusão

executar comandos, Comandos de execução

configuração de tempo de execução, Azure ML Pipelines

S

S3(consulta AWS S3)

SageMaker, Automação da plataforma

monitorização de modelos, Observabilidade para MLOps em Cloud

monetizando MLOps, Contêineres na monetização de MLOps

monitorando a deriva com, Monitorando a deriva com o AWS SageMaker-Monitorando a deriva com o AWS SageMaker

criação de pipeline, [Usando pipelines na Cloud](#)-[Usando pipelines na Cloud](#)

SageMaker AutoPilot, [AWS AutoML](#)-[AWS AutoML](#)

SAM (modelo de aplicação sem servidor da AWS)

Receitas do AWS Lambda, [Receitas do AWS Lambda](#)-[AWS Lambda](#)-[SAM Containerized Deploy](#)

Implantação em contêineres do AWS Lambda-SAM, [Implantação em contêineres do AWS Lambda-SAM](#) - [Implantação em contêineres do AWS Lambda-SAM](#)

SAM Local, [AWS Lambda-SAM Local](#)

ciência, engenharia versus, [Rise of the Machine Learning Engineer](#) e [MLOps](#)

guião, escrever um, [Escrever um guião](#)

auto-hidráulica, [AutoML](#)

serverless (termo), [Termos-chave](#)

Modelo de aplicação sem servidor(ver SAM)

metodologia sem servidor

AWS, [Livro de receitas sem servidor](#) -[Livro de receitas sem servidor](#)

criar uma função sem servidor, [Criar uma função sem servidor](#)-[Criar uma função sem servidor](#)

como padrão de desenho MLOps recomendado, [Padrões de desenho MLOps](#)

responsável pelo serviço, [responsável pelo serviço](#) -[responsável pelo serviço](#)

SHAP, [Explicabilidade do modelo](#), [Explicabilidade do modelo](#)

concha

Shell e comandos do Bash, [Shell e comandos do Bash - Escrevendo um script](#)

ambientes de desenvolvimento em cloud shell, [Ambientes de desenvolvimento em cloud shell - Ambientes de desenvolvimento em cloud shell](#)

definido, [Bash Shell e Comandos](#)

shell script, escrever um, [Escrever um guião](#)

Silver, Nate, [concentra-te na precisão das previsões e não no panorama geral](#)

Simon, Julien, [Aplicar a aprendizagem automática da AWS ao mundo real - Aplicar a aprendizagem automática da AWS ao mundo real](#)

Sinclair, Upton, [AutoML](#)

Design centrado no Spark, [padrões de design MLOps](#)

Certificações relacionadas com SQL, [Certificações relacionadas com SQL](#)

Estudo de caso da rede social desportiva Sqor, [Projectos MLOps na rede social desportiva Sqor - Inteligência do atleta \(produto de IA\)](#)

[Inteligência do atleta \(produto de IA\), Inteligência do atleta \(produto de IA\) - Inteligência do atleta \(produto de IA\)](#)

classificação de influenciadores, [Influencer Rank](#)

Etiquetagem de dados da Mechanical Turk, [Etiquetagem de dados da Mechanical Turk](#)

Fila de espera SQS, definição, [termos-chave](#)

Acesso SSH, [Executar um contentor](#)

estatísticas, descritivas, Estatísticas descritivas e distribuições normais -
Estatísticas descritivas e distribuições normais

aprendizagem automática supervisionada, Conceitos-chave da
aprendizagem automática

swagger, definido, Termos-chave

T

Taleb, Nassim, concentra-te na precisão das previsões e não no panorama
geral

conjunto de dados de destino, conjunto de dados de base versus,
Monitorização da deriva com o AWS SageMaker

acompanhamento de tarefas (na gestão técnica de projectos),
acompanhamento de tarefas

comunicação técnica (melhores práticas DevOps), DevOps e MLOps

portfólio técnico, construindo um, Construir um portfólio técnico para
MLOps - Conseguiram emprego: Não Invadas o Castelo, Entra pela Porta
dos Fundos

exemplo de projeto: aplicação ML nativa da Cloud ou API, Projeto:
Constrói uma aplicação ou API de ML nativa da Cloud

exemplo de projeto: Projeto de contentor Docker e Kubernetes,
Projeto: Projeto de contentor Docker e Kubernetes

exemplo de projeto: solução ML de ponta, Projeto: Cria uma solução
de ML de ponta

exemplo de projeto: pipeline de engenharia de dados de IA sem
servidor, Projeto: Pipeline de engenharia de dados de IA sem servidor

estratégias para conseguir um emprego, Conseguir um emprego: Não
invadas o castelo, entra pela porta dos fundos

gestão técnica de projectos, **Gestão Técnica de Projectos -Seguimento de Tarefas**

como melhor prática de DevOps, **DevOps e MLOps**

plano de projeto, **Plano de Projeto**

acompanhamento de tarefas, **acompanhamento de tarefas**

demonstração semanal, **Demonstração semanal**

certificações tecnológicas(ver certificações)

TensorFlow

converter em ONNX, **Converter TensorFlow em ONNX-
ConverterTensorFlow em ONNX**

TFHub, **TFHub**

TensorFlow Developer Certificate, **GCP**

TensorFlow Playground, **Otimização**

Unidade de processamento TensorFlow(ver TPU)

Sistema educativo "10X melhor", **10X melhor educação - Perturbação do processo de contratação**

Terrell, Dave, **Concentra-te na exatidão da previsão em relação ao panorama geral**

teste

cheques automatizados, **cheques automatizados**

melhoria contínua e, **Melhoria contínua**

linting, **Linting**

implantação de modelos, **Técnicas de teste para implantação de modelos -melhoria contínua**

Código Python, Introdução à computação em Cloud

TFHub (TensorFlow Hub), TFHub

teoria da vantagem competitiva, utilizando a solução "No Code/Low Code" da AWS Comprehend

Thiel, Peter, A situação atual do ensino superior que vai sofrer alterações

autenticação baseada em token, Autenticação de serviços API

TPU (Unidade de processamento TensorFlow)

Projeto Coral e, Coral-Coral

portando sobre modelos não-TPU, Portando sobre modelos não-TPU-
Portando sobre modelos não-TPU

resolução de problemas

Informações sobre aplicações, Informações sobre aplicações

depurar localmente, Depurar localmente-Depurarlocalmente

problemas de implementação, Resolução de problemas de
implementação - depuraçãolocal

recuperar regtos, Recuperar regtos

U

aprendizagem automática não supervisionada, Conceitos-chave da
aprendizagem automática -Conceitos-chave da aprendizagem automática

Acelerador USB, Coral

Utiliza o utilscli.py, livro de receitas do MLOps na AWS

V

versionamento, de conjuntos de dados, [Versionamento de conjuntos de dados](#)

[Visão geral da IA](#) vértice, [Google Cloud Platform](#)

ambiente virtual, Python, [Implementação de DevOps](#)

máquinas virtuais

contentores versus, [Contentores](#)

definido, [Termos-chave](#)

visão(ver visão por computador)

Y

[YAML](#), definido, [Termos-chave](#)

Z

Zhang, Feng, [AutoML](#)

ZSH, [shell](#) e comandos bash, configuração

Sobre os autores

Noah Gift é o fundador da Pragmatic A.I. Labs. Dá palestras no MSDS, na Northwestern, no Duke MIDS Graduate Data Science Program, no Graduate Data Science Program da UC Berkeley, no programa MSBA da UC Davis Graduate School of Management, na UNC Charlotte Data Science Initiative e na Universidade do Tennessee (como parte da Tennessee Digital Jobs Factory). Ensina e concebe cursos de pós-graduação em aprendizagem automática, MLOps, IA e ciência de dados, e presta consultoria sobre aprendizagem automática e arquitetura Cloud para estudantes e professores. Como antigo CTO, colaborador individual e consultor, tem mais de 20 anos de experiência no envio de produtos geradores de receitas em muitas indústrias, incluindo filmes, jogos e SaaS.

Alfredo Deza é um engenheiro de software apaixonado, orador, autor e ex-atleta olímpico com quase duas décadas de experiência em DevOps e engenharia de software. Atualmente, ensina engenharia de aprendizagem automática e dá palestras em todo o mundo sobre desenvolvimento de software, desenvolvimento pessoal e desporto profissional. Alfredo escreveu vários livros sobre DevOps e Python, e continua a partilhar o seu conhecimento sobre infra-estruturas resilientes, testes e práticas de desenvolvimento robustas em cursos, livros e apresentações.

Colofão

O animal que aparece na capa do *Practical MLOps* é um dálmata(*Canis lupus familiaris*). Embora atualmente se encontre em todo o mundo, esta raça de cão pode ser encontrada na atual Croácia, na região histórica da Dalmácia.

O dálmata é um cão musculado de tamanho médio que mede entre 19 e 23 polegadas (ou 48 e 58 cm) e tem uma pelagem branca distinta com manchas pretas. Uma raça altamente cultivada, estes cães são animais de estimação familiares populares, bem como participantes em competições de clubes de canis. Apresentam propensão para problemas de saúde relacionados com a sua criação (incluindo surdez, alergias e cálculos urinários) e têm normalmente uma esperança de vida entre 11 e 13 anos. O Dálmata foi criado como cão de caça, mas depois foi frequentemente usado como cão de carruagem, trotando ao lado das carroagens puxadas por cavalos dos ricos para proteger as carroagens e os seus habitantes, e foram considerados símbolos de estatuto durante o período da Regência em Inglaterra. Também era comum protegerem os cavalos e as carroagens de cervejeiros, caravanistas de ciganos e bombeiros.

Muitos dos animais das capas de O'Reilly estão em vias de extinção; todos eles são importantes para o mundo.

A ilustração da capa é da autoria de Karen Montgomery, baseada numa gravura a preto e branco da *Animate Creation* de Wood. Os tipos de letra da capa são Gilroy Semibold e Guardian Sans. O tipo de letra do texto é Adobe Minion Pro; o tipo de letra do cabeçalho é Adobe Myriad Condensed; e o tipo de letra do código é Ubuntu Mono de Dalton Maag.