MLOps: pipelines de entrega contínua e automação no aprendizado de máquina

Last reviewed 2024-08-28 UTC

Este documento discute técnicas para implementar e automatizar a integração contínua (CI), a entrega contínua (CD) e o treinamento contínuo (CT) para sistemas de aprendizado de máquina (ML). Este documento se aplica principalmente a sistemas de IA preditiva.

A ciência de dados e o ML estão se tornando recursos essenciais para resolver problemas complexos do mundo real, transformando setores e agregando valor em todos os domínios. Atualmente, os ingredientes para aplicar o ML eficaz disponíveis para você:

- Conjuntos de dados grandes
- Recursos de computação sob demanda baratos
- Aceleradores especializados para ML em várias plataformas de nuvem
- Avanços rápidos em diferentes campos de pesquisa de ML (como visão computacional, processamento de linguagem natural, IA generativa e sistemas de IA de recomendações).

Portanto, muitas empresas estão investindo em suas equipes de ciência de dados e recursos de ML para desenvolver modelos preditivos que podem agregar valor aos negócios.

Este documento é destinado aos cientistas de dados e engenheiros de ML que querem aplicar princípios de DevOps a sistemas de ML (MLOps). O *MLOps* é uma cultura e uma prática de engenharia de ML que visa unificar o desenvolvimento de sistemas de ML (Dev) e a operação de sistemas de ML (Ops). A prática de MLOps significa que você defende a automação e o monitoramento de todos os passos da construção do sistema de ML, inclusive integração, teste, lançamento, implantação e gerenciamento de infraestrutura.

Os cientistas de dados podem implementar e treinar um modelo de ML com desempenho preditivo em um conjunto de dados de validação off-line, com os dados de treinamento relevantes para o caso de uso. No entanto, o verdadeiro desafio não é criar um modelo de ML, mas criar um sistema de ML integrado e operá-lo continuamente na produção. Com a longa história dos serviços de ML de produção no Google, descobrimos que pode haver muitas armadilhas na operação de sistemas baseados em ML na produção. Algumas dessas armadilhas são resumidas em Aprendizado de máquina: o cartão de crédito com juros altos da dívida técnica.

Conforme mostrado no diagrama a seguir, apenas uma pequena fração de um sistema de ML real é composta pelo código de ML. Os elementos envolventes necessários são grandes e complexos.

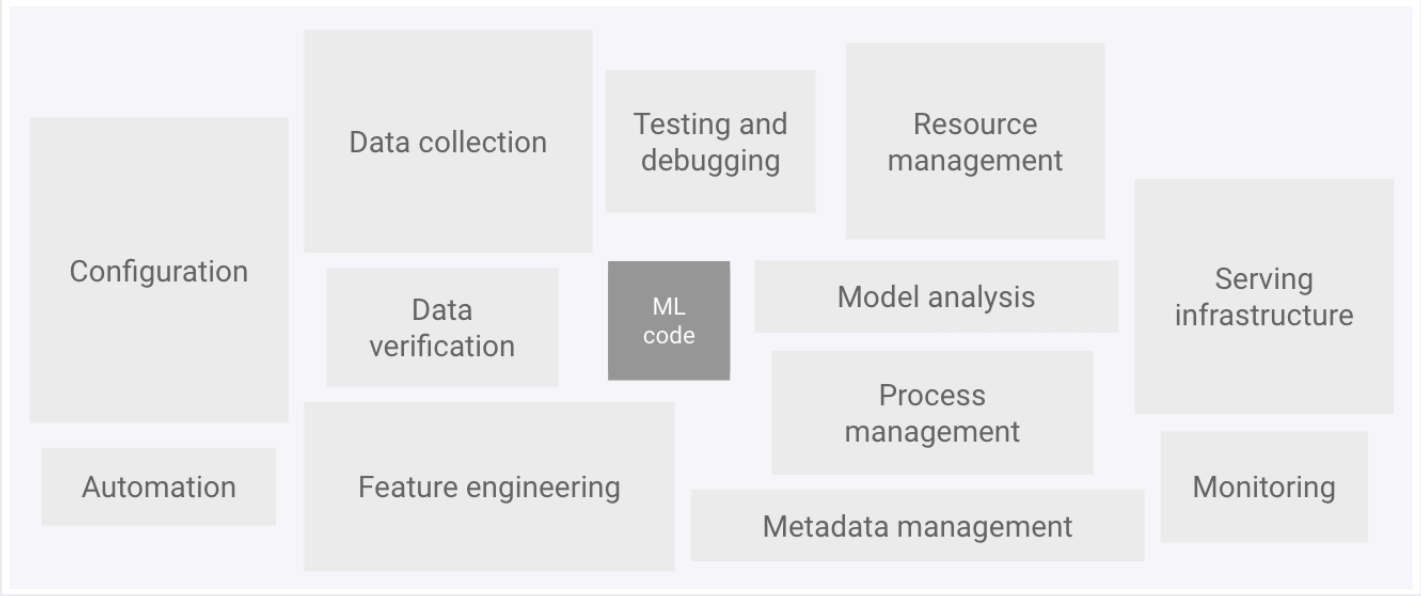

Figura 1. Elementos para sistemas de ML. Adaptado de Débito técnico oculto em sistemas de aprendizado de máquina.

O diagrama anterior mostra os seguintes componentes do sistema:

- Configuração
- Automação
- Coleta de dados
- Verificação de dados
- Testes e depuração
- Gerenciamento de recursos
- Análise de modelos
- Gerenciamento de processos e metadados
- Infraestrutura de exibição
- Monitoramento

Para desenvolver e operar sistemas complexos como esses, aplique princípios de DevOps a sistemas de ML (MLOps). Este documento aborda conceitos a serem considerados ao configurar um ambiente MLOps para suas práticas de ciência de dados, como CI, CD e TC em ML.

Os seguintes tópicos são discutidos:

- DevOps versus MLOps
- Passos para desenvolver modelos de ML
- Níveis de maturidade dos MLOps
- MLOps para IA generativa

DevOps versus MLOps

O DevOps é uma prática comum no desenvolvimento e na operação de sistemas de software em larga escala. Essa prática oferece benefícios como encurtar os ciclos de desenvolvimento, aumentar a velocidade de implantação e as versões confiáveis. Para alcançar esses benefícios, dois conceitos no desenvolvimento do sistema de software são introduzidos:

- Integração contínua (CI)
- Entrega contínua (CD)

Um sistema de ML é um sistema de software. Portanto, práticas semelhantes se aplicam para garantir que você crie e opere sistemas de ML de maneira confiável em escala.

No entanto, os sistemas de ML diferem de outros sistemas de software das seguintes maneiras:

- Habilidades da equipe: em um projeto de ML, a equipe geralmente inclui cientistas de dados ou pesquisadores de ML, que se concentram em análise exploratória de dados, desenvolvimento de modelos e experimentação. Esses membros podem não ser engenheiros de software experientes capaz de criar serviços de classe de produção.
- Desenvolvimento: ML é essencialmente experimental. Teste diferentes recursos, algoritmos, técnicas de modelagem e configurações de parâmetros para encontrar o que funciona melhor para o problema o mais rápido possível. O desafio é acompanhar o que funcionou e o que não funcionou, além de manter a reprodutibilidade enquanto maximiza a reutilização do código.
- Teste: testar um sistema de ML exige mais do que testar outros sistemas de software. Além dos testes típicos de unidade e integração, é preciso validar dados, avaliar a qualidade de modelo treinado e validar o modelo.
- Implantação: em sistemas de ML, a implantação não é tão simples quanto em um modelo de ML treinado off-line como um serviço de previsão. Os sistemas de ML podem exigir que você implante um pipeline de várias etapas para treinar e implantar modelos automaticamente. Esse pipeline aumenta a complexidade e exige que você automatize os passos feitos manualmente antes da implantação por cientistas de dados para treinar e validar novos modelos.
- Produção: os modelos de ML podem ter desempenho reduzido não apenas devido à codificação abaixo do ideal, mas também à evolução contínua dos perfis de dados. Em outras palavras, os modelos podem diminuir com mais frequência do que os sistemas de software tradicionais, e você precisa considerar essa degradação. Portanto, é necessário rastrear as estatísticas resumidas dos dados e monitorar o desempenho on-line do modelo para enviar notificações ou reverter quando os valores diferem das expectativas.

O ML e outros sistemas de software são semelhantes na integração contínua de controle de origem, teste de unidade, teste de integração e entrega contínua do módulo de software ou do pacote. No entanto, no ML, existem algumas diferenças notáveis:

- A CI não se trata mais de apenas testar e validar código e componentes, mas também testar e validar dados, esquemas de dados e modelos.
- O CD não é mais sobre um único pacote de software ou serviço, mas um sistema (um pipeline de treinamento de ML) que deve implantar automaticamente outro serviço (serviço de predição de modelo).
- O TC é uma nova propriedade, exclusiva para sistemas de ML, que se preocupa em treinar e exibir automaticamente os modelos.

A seção a seguir discute os passos típicos de treinamento e avaliação de um modelo de ML para servir como um serviço de previsão.

Passos da ciência de dados para ML

Em qualquer projeto de ML, depois de definir o caso de uso de negócios e estabelecer os critérios de sucesso, o processo de entrega de um modelo de ML à produção envolve os passos a seguir. Esses passos podem ser concluídos manualmente ou por um pipeline automático.

1. Extração de dados: selecione e integre os dados relevantes de várias fontes de dados para a tarefa de ML.
2. Análise de dados: realize uma análise de dados exploratória (EDA) para entender os dados disponíveis para a criação do modelo de ML. Esse processo leva ao seguinte:
    - Noções básicas sobre o esquema de dados e as características esperadas pelo modelo.
    - Identificar a preparação de dados e a engenharia de atributos necessárias para o modelo.
3. Preparação de dados: os dados são preparados para a tarefa de ML. Essa preparação envolve a limpeza de dados, em que você os divide em conjuntos de treinamento, validação e teste. Também é possível aplicar transformações de dados e engenharia de atributos ao modelo que resolve a tarefa de destino. A saída deste passo são as *divisões de dados* no formato preparado.
4. Treinamento de modelo: o cientista de dados implementa diferentes algoritmos com os dados preparados para treinar vários modelos de ML. Além disso, os algoritmos implementados ficam sujeitos ao ajuste de hiperparâmetros para atingir o modelo de ML de melhor desempenho. A saída deste passo é um modelo treinado.
5. Avaliação do modelo: o modelo é avaliado em um conjunto de testes de validação para avaliar a qualidade do modelo. A saída deste passo é um conjunto de métricas para avaliar a qualidade do modelo.
6. Validação do modelo: confirma-se que o modelo é adequado para implantação, o desempenho preditivo é melhor do que uma determinada linha de base.
7. Exibição do modelo: o modelo validado é implantado em um ambiente de destino para exibir previsões. Essa implantação pode ser uma das seguintes:
    - Microsserviços com uma API REST para exibir previsões on-line.
    - Um modelo incorporado a uma borda ou dispositivo móvel.
    - Parte de um sistema de previsão em lote.
8. Monitoramento do modelo: o desempenho preditivo do modelo é monitorado para invocar uma nova iteração no processo de ML.

O nível de automação desses passos define a *maturidade* do processo de ML, que reflete a velocidade de treinamento de novos modelos com base em novos dados ou treinamento de novas implementações. As seções a seguir descrevem três níveis de MLOps, começando pelo nível mais comum, que não envolve automação, até a automatização de pipelines de ML e CI/CD.

Nível 0 de MLOps: processo manual

Muitas equipes têm cientistas de dados e pesquisadores de ML capazes de criar modelos de última geração, mas o processo de criação e implantação de modelos de ML é totalmente manual.

Isso é considerado o nível *básico* de maturidade, ou nível 0. O diagrama a seguir mostra o fluxo de trabalho desse processo.

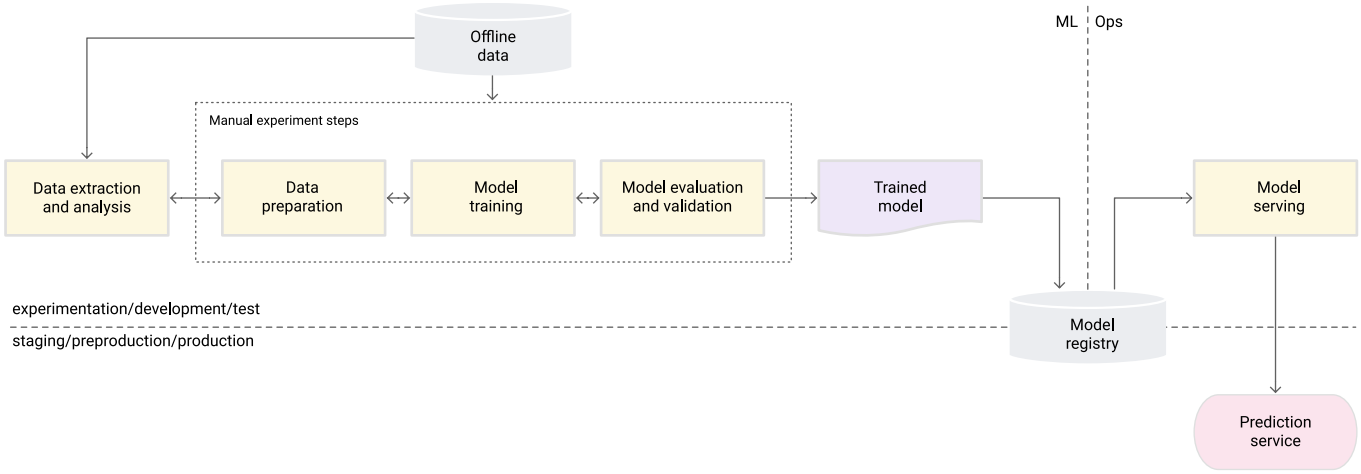

Figura 2. Passos manuais de ML para exibir o modelo como um serviço de previsão.

Características

A lista a seguir destaca as características do processo de MLOps nível 0, conforme mostrado na Figura 2:

- Processo manual, orientado por script e interativo: todos os passos são manuais, inclusive análise de dados, preparação de dados, treinamento de modelo e validação. Ele exige a execução manual de cada passo e a transição manual de um passo para o outro. Esse processo geralmente é impulsionado por código experimental escrito e executado de maneira interativa em notebooks por cientistas de dados, até que um modelo viável seja produzido.
- Desconexão entre ML e operações: o processo separa cientistas de dados que criam o modelo e engenheiros que exibem o modelo como um serviço de previsão. Os cientistas de dados fornecem um modelo treinado como um artefato para a equipe de engenharia implantar na infraestrutura de API. Essa transferência pode incluir colocar o modelo treinado em um local de armazenamento, verificar o objeto de modelo em um repositório de código ou fazer upload dele para um registro de modelos. Em seguida, os engenheiros que implantam o modelo precisam disponibilizar os recursos necessários na produção para veiculação de baixa latência, o que pode levar à *distorção entre treinamento e disponibilização*.
- Iterações de versão não frequentes: o processo pressupõe que sua equipe de ciência de dados gerencia alguns modelos que não mudam com frequência, seja alterando a implementação ou treinando novamente o modelo com novos dados. Uma nova versão de modelo é implantada apenas algumas vezes por ano.
- Sem CI: como algumas alterações de implementação são presumidas, o CI é ignorado. Geralmente, testar o código faz parte da execução dos blocos de notas ou do script. Os scripts e blocos de notas que implementam os passos da experiência são controlados pela origem e produzem artefatos como modelos treinados, métricas de avaliação e visualizações.
- Sem CD: como não há implantações de versão de modelo frequentes, o CD não é considerado.
- Implantação refere-se ao serviço de previsão: o processo é voltado apenas à implantação do modelo treinado como um serviço de previsão (por exemplo, um microsserviço com uma API REST), em vez de implantar todo o sistema de ML.

- Falta de monitoramento de desempenho ativo: o processo não rastreia nem registra as previsões e ações do modelo, que são necessárias para detectar a degradação do desempenho do modelo e outros desvios comportamentais do modelo.

A equipe de engenharia pode ter sua própria configuração complexa para configuração, teste e implantação de API, incluindo segurança, regressão e teste de carga e canário. Além disso, a implantação de produção de uma nova versão de um modelo de ML geralmente passa por testes A/B ou experimentos on-line antes de o modelo ser promovido para exibir todo o tráfego de solicitação de previsão.

Desafios

Os MLOps de nível 0 são comuns em muitas empresas que estão começando a aplicar o ML aos casos de uso. Esse processo manual orientado por cientistas de dados pode ser suficiente quando os modelos raramente são alterados ou treinados. Na prática, os modelos costumam falhar quando são implantados no mundo real. Os modelos não se adaptam às mudanças na dinâmica do ambiente ou às alterações nos dados que descrevem o ambiente. Para obter mais informações, consulte Por que os modelos de aprendizado de máquina apresentam falhas e gravações na produção.

Para resolver esses desafios e manter a precisão do modelo na produção, é preciso fazer o seguinte:

- Monitore ativamente a qualidade do modelo na produção: o monitoramento permite detectar a degradação do desempenho e a inatividade do modelo. Ele atua como uma indicação para uma nova iteração de experimento e novo treinamento (manual) do modelo em novos dados.
- Treinar com frequência seus modelos de produção: para coletar os padrões em evolução e emergentes, é necessário treinar novamente o modelo com os dados mais recentes. Por exemplo, se o app recomenda produtos de moda usando ML, as recomendações dele precisam se adaptar às últimas tendências e produtos.
- Teste continuamente novas implementações para produzir o modelo: para aproveitar as ideias e os avanços mais recentes em tecnologia, é preciso testar novas implementações, como engenharia de atributos, arquitetura de modelo e hiperparâmetros. Por exemplo, se você usa a visão computacional na detecção facial, os padrões faciais são corrigidos, mas novas técnicas melhores podem melhorar a precisão da detecção.

Para lidar com os desafios desse processo manual, as práticas de MLOps para CI/CD e TC são úteis. Ao implantar um pipeline de treinamento de ML, é possível ativar o TC e configurar um sistema de CI/CD para testar, criar e implantar rapidamente novas implementações do pipeline de ML. Esses recursos serão discutidos em mais detalhes nas próximas seções.

Nível 1 de MLOps: automação de pipeline de ML

O objetivo do nível 1 é realizar o treinamento contínuo do modelo automatizando o pipeline de ML. Isso permite que você alcance a entrega contínua do serviço de predição de modelo. Para automatizar o processo de uso de novos dados para treinar novamente os modelos na produção, é necessário introduzir dados automatizados e passos de validação do modelo, bem como acionadores de pipeline e gerenciamento de metadados.

A figura a seguir é uma representação esquemática de um pipeline de ML automatizado para TC.

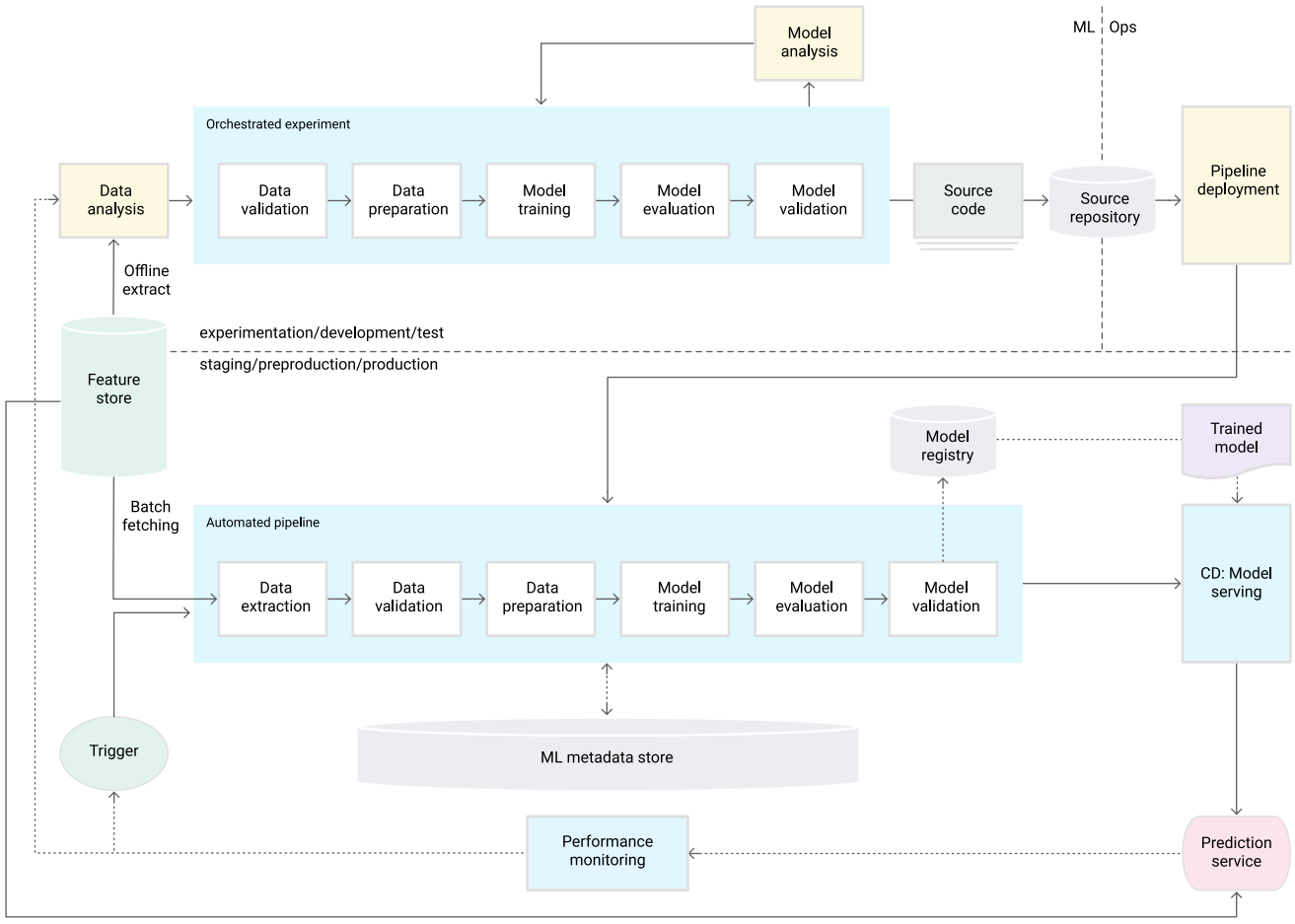

Figura 3. Automação de pipeline de ML para TC.

Características

A lista a seguir destaca as características da configuração de MLOps nível 1, conforme mostrado na Figura 3:

- Experimento rápido: os passos do experimento de ML são orquestrados. A transição entre os passos é automatizada, o que leva a uma iteração rápida dos experimentos e à melhor preparação para mover todo o pipeline para a produção.
- TC do modelo em produção: o modelo é treinado automaticamente na produção usando dados recentes com base em acionadores de pipeline ativos, que são discutidos na próxima seção.
- Simetria experimental-operacional: a implementação do pipeline usada no ambiente de desenvolvimento ou experiência é usada no ambiente de pré-produção e produção, que é um aspecto fundamental da prática de MLOps para unificar DevOps.
- Código modularizado para componentes e pipelines: para criar pipelines de ML, os componentes precisam ser reutilizáveis, compostos e potencialmente compartilháveis em pipelines de ML. Portanto, embora o código de EDA ainda possa estar em notebooks, o código-fonte dos componentes precisa ser modulado. Além disso, os componentes devem ser colocados em contêineres para:
  - Desassociar o ambiente de execução do tempo de execução do código personalizado.
  - Tornar o código reproduzível entre os ambientes de desenvolvimento e produção.
  - Isole cada componente no pipeline. Os componentes podem ter sua própria versão do ambiente de execução e ter diferentes linguagens e bibliotecas.

- Entrega contínua de modelos: um pipeline de ML em produção fornece continuamente serviços de previsão para novos modelos treinados com novos dados. O passo de implantação do modelo, que exibe o modelo treinado e validado como um serviço de predição on-line, é automatizado.
- Implantação do pipeline: no nível 0, você implanta um modelo treinado como um serviço de previsão na produção. Para o nível 1, você implanta um pipeline de treinamento inteiro, que é executado de maneira automática e recorrente para veicular o modelo treinado como o serviço de previsão.

Componentes adicionais

Esta seção discute os componentes necessários à arquitetura para ativar o treinamento contínuo de ML.

Validação de modelos e dados

Quando você implanta o pipeline de ML na produção, um ou mais gatilhos discutidos na seção Gatilhos de pipeline de ML executam automaticamente o pipeline. O pipeline espera que dados novos e ativos produzam uma nova versão de modelo treinada com os novos dados (conforme mostrado na Figura 3). Portanto, os passos automatizados de *validação de dados* e *validação de modelo* são necessários no pipeline de produção para garantir o seguinte comportamento esperado:

- Validação de dados: este passo é necessário antes do treinamento do modelo para decidir se você deve treinar novamente o modelo ou interromper a execução do pipeline. Essa decisão será tomada automaticamente se o seguinte for identificado pelo pipeline.
    - Desvios do esquema de dados: são considerados anomalias nos dados de entrada. Portanto, os dados de entrada que não estão em conformidade com o esquema esperado são recebidos pelas etapas do pipeline downstream, incluindo as etapas de processamento de dados e treinamento de modelo. Nesse caso, interrompa o pipeline para que a equipe de ciência de dados possa investigar. A equipe pode lançar uma correção ou uma atualização para o pipeline para lidar com essas alterações no esquema. Os desvios de esquema incluem receber recursos inesperados, não receber todos os recursos esperados ou receber recursos com valores inesperados.
    - Desvios de valores de dados: esses desvios são alterações significativas nas propriedades estatísticas dos dados, o que significa que os padrões de dados estão mudando, e você precisa acionar um novo treinamento do modelo para capturar essas alterações.
- Validação do modelo: esse passo ocorre depois que você treina o modelo com base nos novos dados. Avalie e valide o modelo antes de promovê-lo para produção. Esse passo de *validação de modelo off-line* consiste no seguinte:
    - Produzir valores de métricas de avaliação usando o modelo treinado em um conjunto de dados de teste para avaliar a qualidade preditiva do modelo.
    - Comparar os valores de métrica de avaliação produzidos pelo modelo recém-treinado com o modelo atual. Por exemplo, modelo de produção, modelo de linha de base ou outros modelos de requisitos de negócios. Certificar-se de que o novo modelo produz um desempenho melhor do que o modelo atual antes de promovê-lo para produção.
    - Verificar se o desempenho do modelo é consistente em vários segmentos dos dados. Por exemplo, seu modelo de desligamento de clientes recém-treinado pode produzir uma precisão preditiva melhor do que o modelo anterior, mas os valores de precisão por região do cliente podem ter uma grande variação.

- Certificar-se de testar o modelo para implantação, incluindo compatibilidade de infraestrutura e consistência com a API do serviço de previsão.

Além da validação do modelo off-line, um modelo recém-implantado passa por uma *validação de modelo on-line*, em uma implantação canário ou em uma configuração de teste A/B, antes de exibir a previsão para o tráfego on-line.

Armazenamento de recursos

Um componente adicional opcional para automação de pipeline de ML nível 1 é um armazenamento de recursos. Um armazenamento de recursos é um repositório centralizado em que você padroniza a definição, o armazenamento e o acesso dos recursos para treinamento e exibição. Um armazenamento de recursos precisa fornecer uma API para exibição em lote de alta capacidade e exibição em tempo real de baixa latência para os valores de recursos e para oferecer suporte a cargas de trabalho de treinamento e exibição.

O armazenamento de recursos ajuda os cientistas de dados a fazer o seguinte:

- Descobrir e reutilizar os conjuntos de recursos disponíveis para suas entidades, em vez de recriar os mesmos ou semelhantes.
- Evitar ter recursos semelhantes que tenham definições diferentes, mantendo os recursos e os metadados relacionados.
- Exibir valores de recursos atualizados do armazenamento de recursos.
- Evitar desvios entre treinamento e exibição usando o armazenamento de recursos como fonte de dados para experimentação, treinamento contínuo e exibição on-line. Essa abordagem garante que os recursos usados no treinamento sejam os mesmos usados durante a exibição:
    - Para a experimentação, os cientistas de dados podem coletar um extrato off-line do armazenamento de recursos para realizar os experimentos.
    - Para o treinamento contínuo, o pipeline de treinamento automatizado de ML pode buscar um lote dos valores de recursos atualizados do conjunto de dados usados para a tarefa de treinamento.
    - Para a previsão on-line, o serviço de previsão pode buscar um lote dos valores de recurso relacionados à entidade solicitada, como recursos demográficos do cliente, recursos do produto e recursos de agregação de sessão atuais.
    - Para previsão on-line e recuperação de recursos, o serviço de previsão identifica os recursos relevantes para uma entidade. Por exemplo, se a entidade for um cliente, os recursos relevantes poderão incluir idade, histórico de compras e comportamento de navegação. O serviço agrupa esses valores de atributos e recupera todos os atributos necessários para a entidade de uma só vez, em vez de individualmente. Esse método de recuperação ajuda na eficiência, especialmente quando você precisa gerenciar várias entidades.

Gerenciamento de metadados

As informações sobre cada execução do pipeline de ML são registradas para ajudar na linhagem, na reprodutibilidade e na comparação de dados e artefatos. Ele também ajuda a depurar erros e anomalias. Ao executar o pipeline, o armazenamento de metadados de ML registra os seguintes metadados:

- As versões do pipeline e do componente que foram executadas.

- A data de início e de término, a hora e o tempo que o pipeline levou para concluir cada um dos passos.
- O executor do pipeline.
- Os argumentos do parâmetro que foram passados para o pipeline.
- Os ponteiros para os artefatos produzidos por cada passo do pipeline, como a localização de dados preparados, anomalias de validação, estatísticas calculadas e vocabulário extraído dos recursos categóricos. O rastreamento dessas saídas intermediárias ajuda você a retomar o pipeline a partir do passo mais recente se o pipeline parou devido a um passo com falha, sem precisar executar novamente passos que já foram concluídos.
- Um ponteiro para o modelo treinado anterior se você precisar reverter para uma versão de modelo anterior ou se precisar produzir métricas de avaliação para uma versão de modelo anterior quando o pipeline receber novos dados de teste durante o passo de validação do modelo.
- As métricas de avaliação do modelo produzidas durante a avaliação do modelo para os conjuntos de treinamento e teste. Essas métricas ajudam a comparar o desempenho de um modelo recém-treinado com o desempenho registrado do modelo anterior durante o passo de validação do modelo.

Acionadores de pipeline de ML

É possível automatizar os pipelines de produção de ML para treinar novamente os modelos com novos dados, dependendo do caso de uso:

- Sob demanda: execução manual ad-hoc do pipeline.
- Em uma programação: os dados novos e rotulados estão disponíveis sistematicamente para o sistema de ML diariamente, semanalmente ou mensalmente. A frequência de treinamento também depende da frequência com que os padrões de dados mudam e do custo do treinamento dos modelos.
- Sobre a disponibilidade de novos dados de treinamento: novos dados não estão sistematicamente disponíveis para o sistema de ML e, em vez disso, estão disponíveis ad hoc quando novos dados são coletados e disponibilizados nos bancos de dados de origem.
- Na degradação de desempenho do modelo: o modelo é treinado novamente quando há uma degradação perceptível no desempenho.
- Em alterações significativas nas distribuições de dados (*mudança de conceito*). É difícil avaliar o desempenho completo do modelo on-line, mas você percebe alterações significativas nas distribuições de dados dos recursos usados para realizar a previsão. Essas alterações sugerem que o modelo ficou desatualizado e que precisa ser treinado novamente em dados recentes.

Desafios

Supondo que novas implementações do pipeline não sejam implantadas com frequência e que você esteja gerenciando apenas alguns pipelines, normalmente o pipeline e seus componentes são testados manualmente. Além disso, novas implementações de pipeline são implantadas manualmente. O código-fonte testado do pipeline também é enviado para a equipe de TI implantar no ambiente de destino. Essa configuração é adequada ao implantar novos modelos com base em novos dados, e não em novas ideias de ML.

No entanto, é preciso testar novas ideias de ML e implantar rapidamente novas implementações dos componentes de ML. Ao gerenciar muitos pipelines de ML na produção, é preciso uma

configuração de CI/CD para automatizar a criação, o teste e a implantação de pipelines de ML.

Nível 2 de MLOps: automação de pipeline de CI/CD

Para uma atualização rápida e confiável dos pipelines em produção, é preciso ter um sistema de CI/CD robusto automatizado. Um sistema de CI/CD automatizado permite que os cientistas de dados explorem rapidamente novas ideias sobre engenharia de atributos, arquitetura de modelos e hiperparâmetros. É possível implementar essas ideias e criar, testar e implantar automaticamente os novos componentes do pipeline no ambiente de destino.

O diagrama a seguir mostra a implementação do pipeline de ML usando CI/CD, que tem as características da configuração de pipelines de ML automatizados mais as rotinas automatizadas de CI/CD.

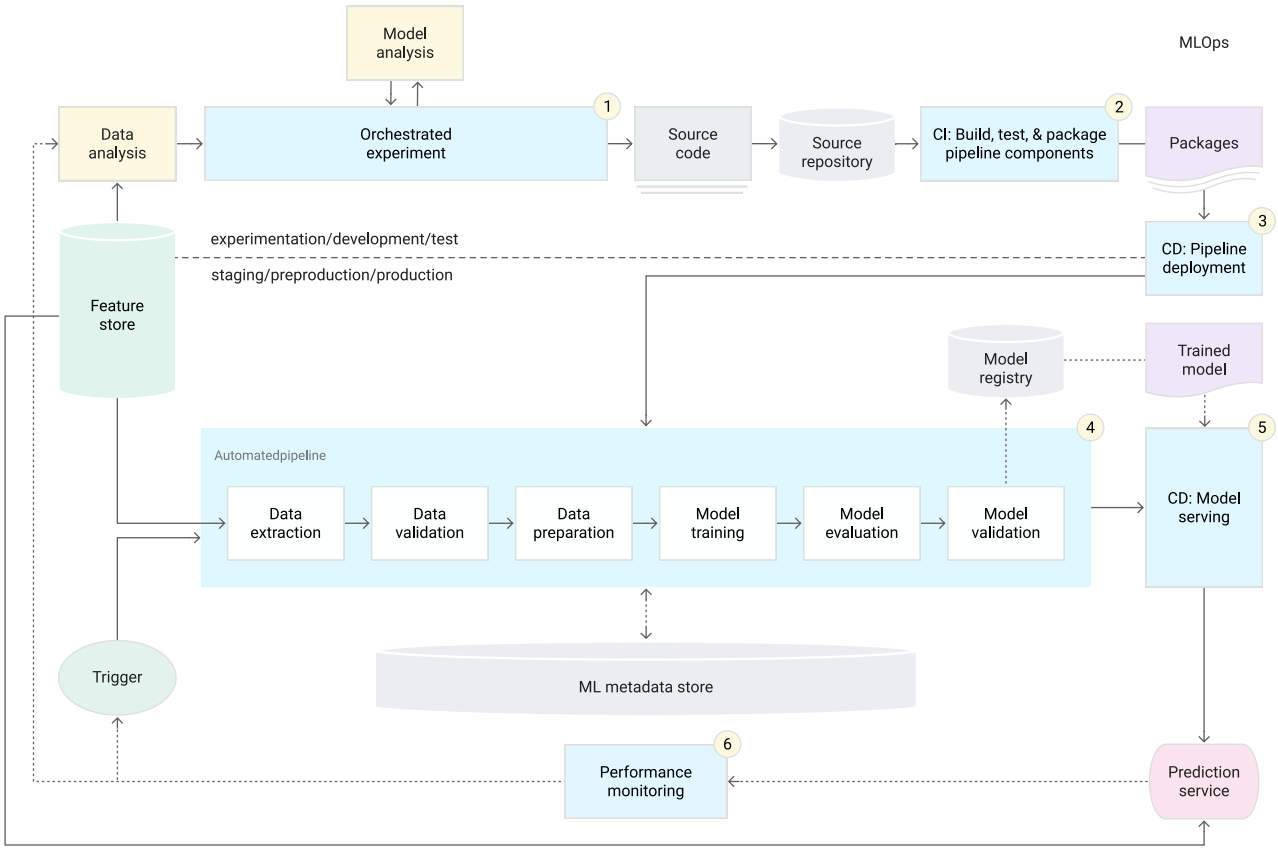

Figura 4. Pipeline de ML automatizado e CI/CD.

Essa configuração de MLOps abrange os seguintes componentes:

- Controle de origem
- Serviços de teste e criação
- Serviços de implantação
- Registro de modelos
- Armazenamento de recursos
- Armazenamento de metadados de ML
- Orquestrador de pipeline de ML

Características

O diagrama a seguir mostra os estágios do pipeline de automação de CI/CD de ML:

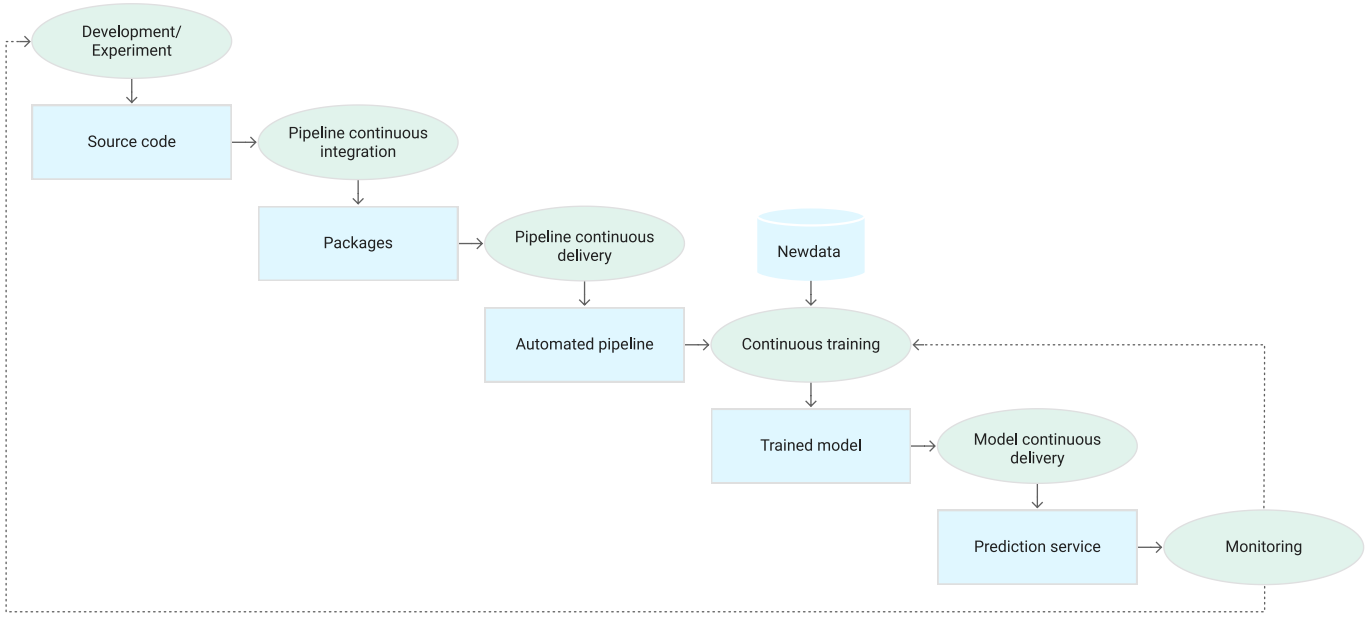

Figura 5. Estágios do pipeline de ML automatizado de CI/CD.

O pipeline consiste nos seguintes estágios:

1. Desenvolvimento e experimentação: experimento iterativo de novos algoritmos de ML e nova modelagem em que os passos da experiência são orquestrados. A saída desse estágio é o código-fonte dos passos do pipeline de ML que são enviados para um repositório de origem.
2. Integração contínua do pipeline: criação do código-fonte e execução de vários testes. As saídas desse estágio são componentes de pipeline (pacotes, executáveis e artefatos) que serão implantados em um estágio posterior.
3. Entrega contínua do pipeline: implantação dos artefatos produzidos pelo cenário de CI no ambiente de destino. A saída desse estágio é um pipeline implantado com a nova implementação do modelo.
4. Acionamento automatizado: execução automática do pipeline na produção com base em uma programação ou em resposta a um acionador. A saída desse estágio é um modelo treinado que é enviado ao registro de modelos.
5. Entrega contínua do modelo: exibição do modelo treinado como um serviço de previsão para as previsões. A saída desse estágio é um serviço de previsão de modelo implantado.
6. Monitoramento: coleta de estatísticas sobre o desempenho do modelo com base em dados ativos. A saída desse estágio é um acionador para executar o pipeline ou executar um novo ciclo de experiência.

O passo de análise de dados ainda é um processo manual para cientistas de dados antes que o pipeline inicie uma nova iteração do experimento. O passo da análise do modelo também é um processo manual.

Integração contínua

Nesta configuração, o pipeline e os componentes dele são criados, testados e empacotados quando o novo código é confirmado ou enviado ao repositório de código-fonte. Além de criar

pacotes, imagens de contêiner e executáveis, o processo de CI pode incluir os seguintes testes:

- Teste de unidade da lógica de engenharia de atributos.
- Teste de unidade dos diferentes métodos implementados no modelo. Por exemplo, uma função que aceita uma coluna de dados categóricos e a codifica como um recursos one-hot.
- Testar se o treinamento do modelo converge (ou seja, a perda do modelo diminui por iterações e overfits de alguns registros de amostra).
- Testar se o treinamento do modelo não produz valores NaN devido à divisão por zero ou à manipulação de valores pequenos ou grandes.
- Testar se cada componente no pipeline produz os artefatos esperados.
- Testar a integração entre componentes de pipeline.

Entrega contínua

Nesse nível, o sistema entrega continuamente novas implementações de pipeline ao ambiente de destino que, por sua vez, fornece serviços de previsão do modelo recém-treinado. Para uma entrega contínua e confiável de pipelines e modelos, considere o seguinte:

- Verificar a compatibilidade do modelo com a infraestrutura de destino antes de implantar o modelo. Por exemplo, é preciso verificar se os pacotes exigidos pelo modelo estão instalados no ambiente de veiculação e se os recursos de memória, computação e acelerador necessários estão disponíveis.
- Testar o serviço de previsão chamando a API de serviço com as entradas esperadas e verificar se você recebe a resposta esperada. Esse teste geralmente captura problemas que podem ocorrer ao atualizar a versão do modelo e espera uma entrada diferente.
- Testar o desempenho do serviço de previsão, que envolve o teste de carga do serviço para capturar métricas como consultas por segundo (QPS) e latência do modelo.
- Validar os dados para treinamento ou previsão em lote.
- Verificar se os modelos atendem às metas de desempenho preditivo antes de serem implantados.
- Implantação automatizada em um ambiente de teste, por exemplo, uma implantação acionada por push de código para a ramificação de desenvolvimento.
- Implantação semiautomática em um ambiente de pré-produção, por exemplo, uma implantação acionada pela mesclagem de código para a ramificação principal após os revisores aprovarem as alterações.
- Implantação manual em um ambiente de produção após várias execuções bem-sucedidas do pipeline no ambiente de pré-produção.

Resumindo, implementar o ML em um ambiente de produção não significa apenas implantar o modelo como uma API para previsão. Em vez disso, significa implantar um pipeline de ML que pode automatizar o treinamento e a implantação de novos modelos. A configuração de um sistema de CI/CD permite testar e implantar automaticamente novas implementações de pipeline. Esse sistema permite lidar com alterações rápidas nos dados e no ambiente de negócios. Não é preciso mover imediatamente todos os processos de um nível para outro. É possível implementar gradualmente essas práticas para ajudar a melhorar a automação do desenvolvimento e da produção do sistema de ML.

A seguir

- Saiba mais sobre Arquitetura para MLOps usando o TensorFlow Extended, os pipelines da Vertex AI e o Cloud Build.
- Saiba mais sobre o Guia para profissionais de operações de machine learning (MLOps).
- Assista às Práticas recomendadas de MLOps no Google Cloud (Cloud Next '19) no YouTube.
- Para uma visão geral dos princípios e recomendações de arquitetura específicos para cargas de trabalho de IA e ML no Google Cloud, consulte a perspectiva de IA e ML no framework bem arquitetado.
- Para mais arquiteturas de referência, diagramas e práticas recomendadas, confira a Central de arquitetura do Cloud.

Colaboradores

Autores:

- Jarek Kazmierczak | Arquiteto de soluções
- Khalid Salama | Engenheiro de software, aprendizado de máquina
- Valentin Huerta | Engenheiro de IA

Outro colaborador: Sunil Kumar Jang Bahadur | Engenheiro de clientes