

Níveis de Maturidade (e onde nós estamos)

Last updated by | Gabriel Pehls | 18 de fev. de 2024 at 13:47 BRT

Definição

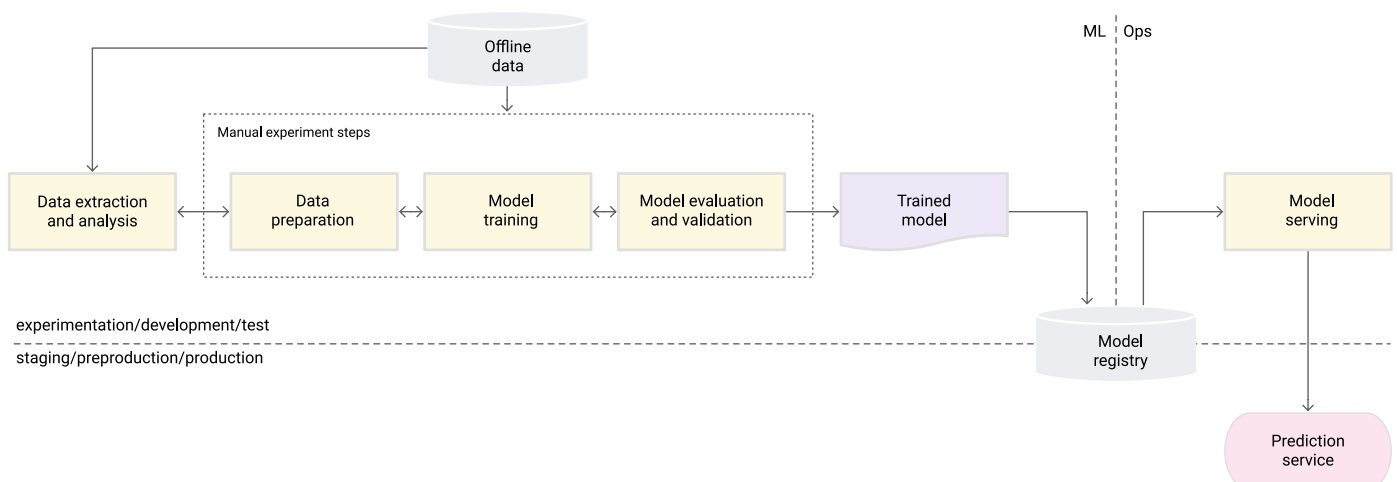
Segundo o [Google](#), existem passos críticos envolvendo projetos de ML:

- **Extração de Dados**, onde realizamos os passos condizentes a Engenharia de Dados, com a seleção e a integração de dados relevantes de várias fontes de dados para a tarefa/projeto de ML;
- **Análise de Dados**, onde dados são preparados para a tarefa de ML, envolvendo limpeza, separação de dados (treino/validação/teste), transformações de dados e engenharia de atributos (feature engineering) ao modelo, que envolve a tarefa de destino, sendo a saída deste passo as divisões de dados no formato preparado;
- **Treinamento do Modelo**, onde diferentes modelos e abordagens são implementados, com os dados preparados, estando os algoritmos implementados sujeitos ao ajuste de hiperparâmetros para atingir o modelo de melhor desempenho possível. A saída, aqui, deve ser o modelo treinado;
- **Avaliação do Modelo**, onde verificamos a qualidade do modelo de ML, tendo um conjunto de métricas definidos para avaliar a qualidade do modelo;
- **Validação do Modelo**, onde confirmamos se o modelo é adequado para implantação, caso o desempenho do modelo seja melhor do que uma determinada linha de base (ou *threshold*);
- **Exibição do Modelo**, ou ainda, a implantação em um ambiente de destino, para exibir previsões. Comumente, a implantação pode ser em Microsserviços, como uma API REST, para exibir previsões on-line ou em resposta ao envio de dados por aplicativos/sistemas, modelos incorporados a uma borda ou dispositivo móvel, ou ainda parte de um sistema de previsão em lote (ou *batch*);
- **Monitoramento do Modelo**, onde o desempenho preditivo do modelo é monitorado, sendo passível de invocar uma nova iteração no processo de ML caso haja diferenças no desempenho ou nos dados de entrada do modelo (*data drift*, *concept drift* ou *model drift*).

Níveis de Maturidade

Novamente, segundo o [Google](#), existem 3 níveis de maturidade em MLOps:

Nível 0: Processo Manual

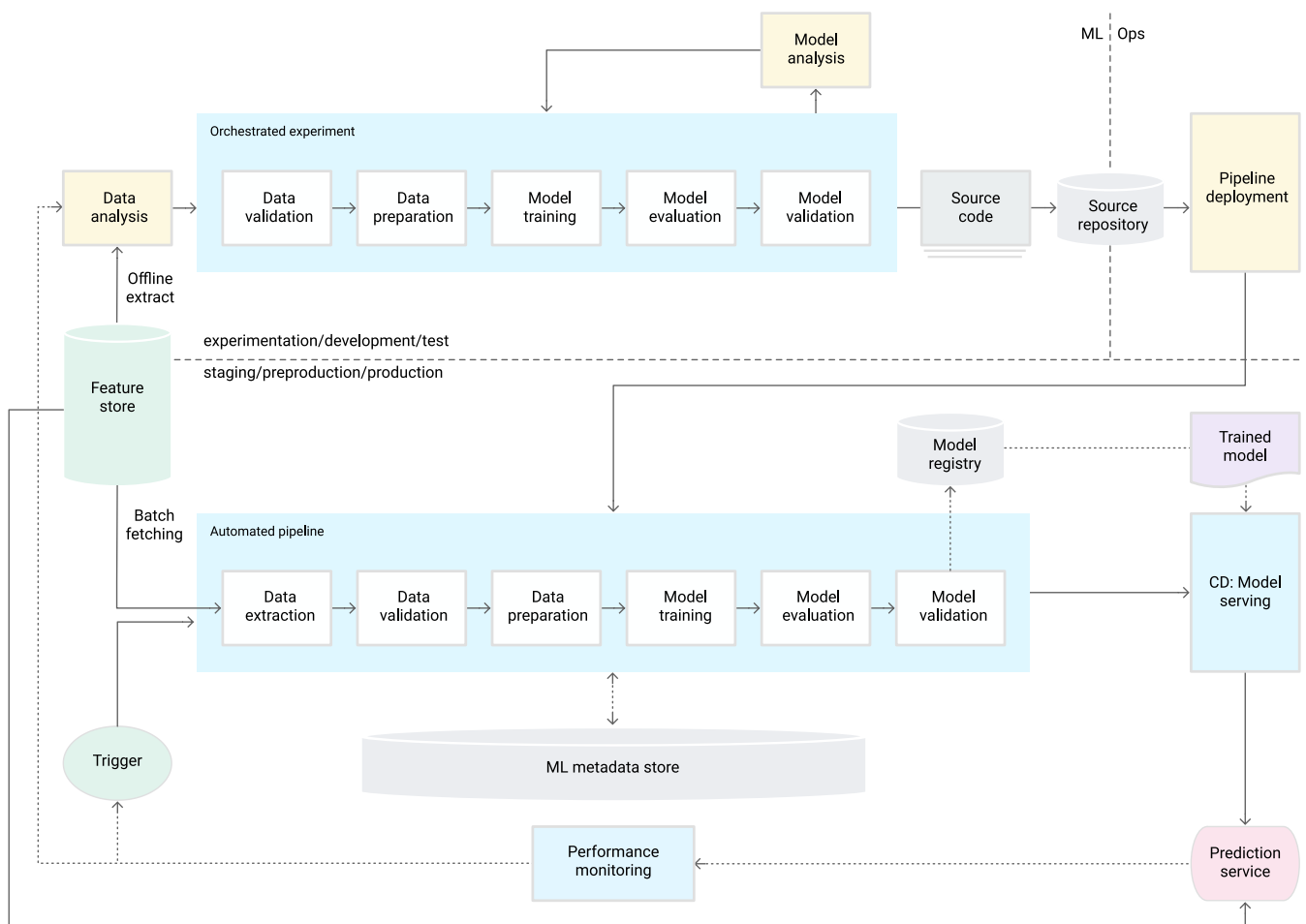


- O Processo é manual, inclusive análise de dados, preparação de dados, treinamento de modelos e validações, sendo impulsionado por código experimental, escrito e executado em notebooks pelo

cientista de dados, orientado por script e interativo, até que um modelo viável seja produzido;

- Há uma clara desconexão entre ML e Ops, onde os modelos são fornecidos como um artefato a serem implementados em uma infraestrutura, seja a partir de um repositório de código ou através de uma ferramenta de registro de modelos. Assim que o modelo é fornecido, outra equipe disponibiliza recursos necessários para sua entrada em produção, buscando uma veiculação com baixa latência e recursos mínimos, podendo levar a uma distorção entre treinamento e disponibilização;
- As iterações de versão não são frequentes, pois o processo pressupõe que o time de ciência de dados os gerenciam, seja alterando a implementação ou treinando novamente os modelos, o que não necessariamente se torna verdadeiro.
- Sem CI, pois algumas alterações são presumidas. Geralmente, testar o código faz parte da execução de notebooks ou scripts, que implementam os passos da experimentação, produzindo artefatos como modelos treinados, métricas de avaliação e visualizações;
- Sem CD, pois não há implantações de versões de modelo frequentes;
- A implantação se refere apenas ao serviço de previsão, como um microserviço ou API REST, no lugar de implantar todo o sistema de ML;
- Falta de Monitoramento de desempenho ativo, tornando a detecção da degradação dos modelos e mudanças comportamentais dos dados/modelos custosa.

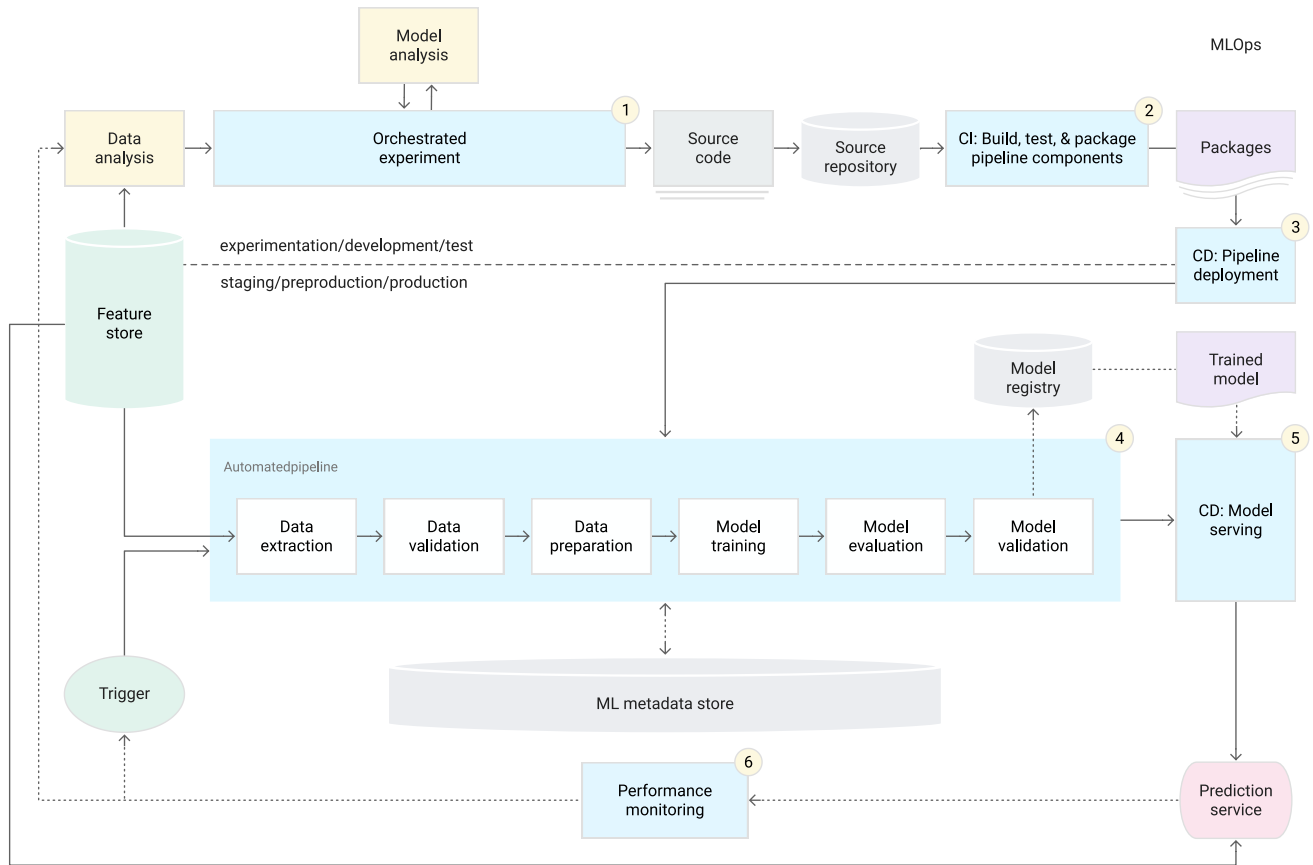
Nível 1: Automação de pipelines de ML



- Experimentação rápida, pois os passos do experimento de ML são orquestrados, a transição entre os passos é automatizada, levando a uma iteração rápida dos experimentos e uma melhor preparação para mover todo o pipeline para a produção;
- Treinamento Contínuo (TC) do modelo em produção, usando dados recentes com base em acionadores de pipeline ativos;

- Simetria experimental-operacional, sendo a implementação do pipeline utilizada no ambiente de dev/experiência utilizada no ambiente de pré-prod e produção, sendo um aspecto fundamental da prática de MLOps para unificar DevOps;
- Código modularizado para componentes e pipelines, devido a necessidade dos componentes serem reutilizáveis, compostos e potencialmente compartilháveis em pipelines de ML. Aqui, portanto, embora o código da EDA possa estar em notebooks, o código-fonte dos componentes precisa ser modulado, além de serem colocados em contêineres para:
 - Desassociar o ambiente de execução do tempo de execução do código personalizado;
 - Tornar o código reproduzível entre ambientes;
 - Isolar cada componente no pipeline, pois cada um deles pode ter sua própria versão do ambiente de execução e ter diferentes linguagens e bibliotecas.
- Entrega contínua de modelos, tendo uma entrega contínua dos serviços de previsão para novos modelos treinados com novos dados, sendo 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;
- Gerenciamento de metadados, pois as informações de cada execução do pipeline deve ser registrado para ajudar na linhagem, reprodutibilidade e na comparação de dados e artefatos. Metadados como versões de componentes e pipeline, data-hora de início e término de cada passo, o executor, parâmetros, e ponteiros para artefatos produzidos por cada passo do pipeline, e modelos anteriores e novos, visando a reversão em um modelo anterior caso necessário, além de métricas de avaliação do modelo para conjuntos de treino e teste, visando a execução de triggers caso necessário;
- Acionadores (triggers) de pipelines, dado que poderemos invocar o treinamento de novos modelos de forma "manual", em uma programação diária/semanal/mensal, de acordo com a disponibilidade de novos dados (caso onde novos dados estão disponíveis apenas em períodos específicos), na degradação de desempenho do modelo (por isso, metadados de avaliação do modelo são importantes), ou ainda em alterações significativas nas distribuições dos dados (concept drift), que sugerem que o modelo pode estar desatualizado.

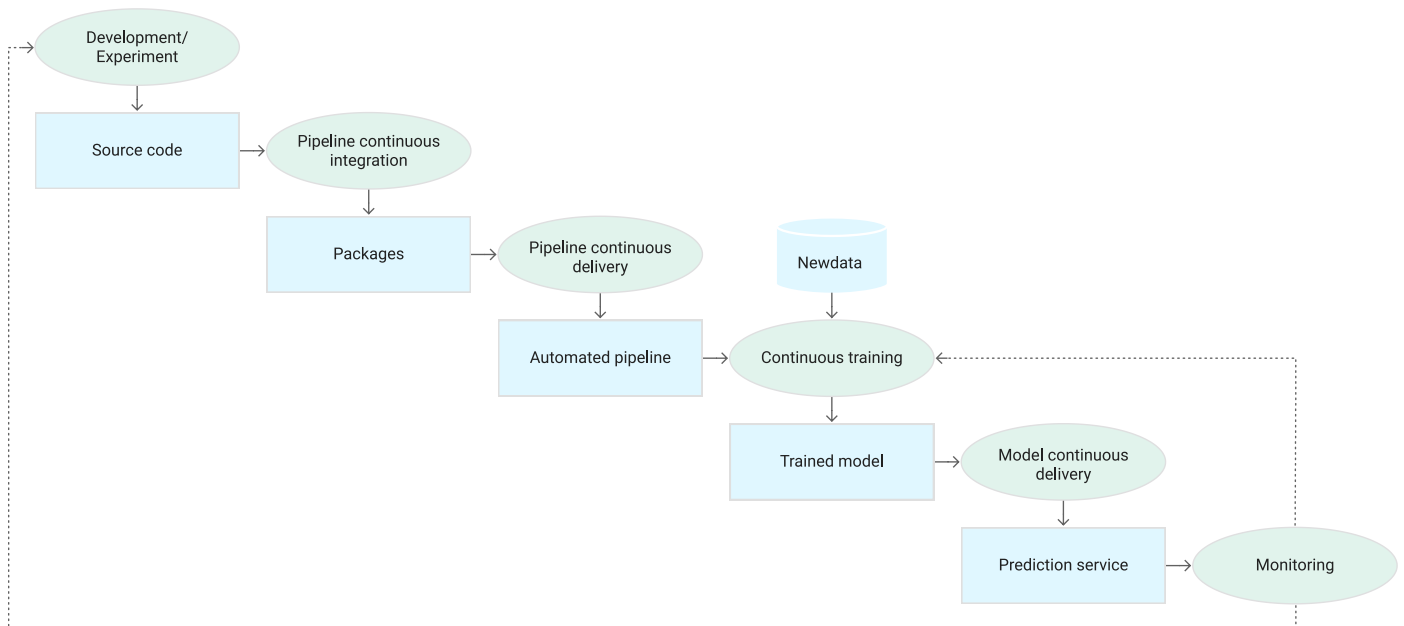
Nível 2: Automação de pipelines de CI/CD



Nesse último estágio, temos 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

Ainda, há alguns estágios do pipeline de automação de CI/CD de ML:




- No primeiro estágio, de desenvolvimento e experimentação, há um experimento iterativo de novos algoritmos e modelagens, dado que os passos da experiência são orquestrados. A saída é um código-fonte dos passos do pipeline de ML que serão enviados a um repositório de origem;
- Integração contínua de pipeline, onde são executados vários testes e é criado o código fonte. A saída seriam os componentes do pipeline (como pacotes, executáveis e artefatos) que serão implantados no próximo passo;
- Entrega Contínua do pipeline, implantando os artefatos produzidos no passo anterior no ambiente de destino, tendo como saída um pipeline já com uma nova implementação do modelo;
- Acionamento automatizado, executando automaticamente o pipeline na produção com base em uma programação ou em resposta a uma trigger, tendo como saída um modelo treinado, que é enviado ao registro de modelos;
- Entrega contínua do modelo, onde o modelo treinado é exibido como um serviço de previsão;
- Monitoramento, coletando estatísticas sobre o desempenho do modelo com base em dados ativos, tendo como saída um acionador para executar o pipeline novamente ou um novo ciclo de experiência

E onde nós estamos?

Isolando projetos que utilizaram modelos de Machine Learning como parte do processo, notamos que a utilização do Registro de modelos do Databricks acontece, de forma direta ou indireta, tendo sido utilizado como forma de estudo e de experimentação entre alguns times. Em uma reunião na sexta-feira, dia 30/06/2023, mencionamos a utilização do Model Catalog e Model Serving da ferramenta, em alguns projetos, servindo modelos como uma API REST a partir do Databricks. Identificamos também que, de fato, existe um custo envolvido - e como temos custo, analisaremos a viabilidade de utilizar o mesmo, comparando com outras ferramentas, como uma das primeiras iniciativas de boas práticas de MLOps. Tal ponto será trabalhado na subpágina [Model Serving](#), linkado com o card  14967 [Estudo] Formas de Deploy de modelos de ML | New

Ainda não possuímos uma experimentação orquestrada em todos os times, apesar da utilização dos pipelines do databricks como forma de treinamento e execução dos processos em alguns times, de forma que não estaríamos aptos a ser classificados como um "Nível 2", pelas especificações acima. Para o atingir, portanto, a iniciativa de criar um framework para modelos de ML, é uma forma essencial para estabelecer padrões, permitindo uma reprodutibilidade entre ambientes diferentes de experimentos, e, consequentemente, de projetos efetivamente em produção. Esse ponto permitirá a orquestração dos experimentos, portanto, tendo cada um dos passos de um projeto de ML como diferentes níveis de automatização.

Analisaremos o mesmo na subpágina [Treinamento Contínuo de Modelos](#), linkada com o card  14968 [Ação] Criação do Framework para desenvolvimento de ML | New

Como uma das etapas finais, a avaliação da performance dos modelos poderá servir de trigger para um novo treinamento, ou uma nova iteração das etapas de criação de um novo modelo. Por já termos modelos de Forecast sendo executadas como um processo da área, esse passo é colocado como um dos primeiros, afim de trazermos uma economia ao não termos novos treinamentos, caso algumas condições sigam verdadeiras para modelos já treinados anteriormente. Tal característica está presente apenas no nível de maturidade 2, conforme vimos, mas entendemos que sua importância o coloca na posição atual. Ele será tratado na subpágina [Performance Monitoring](#), já possui algumas codificações prontas, e deve ser tratado no card  14969 [Ação] Padronização de monitoramento de ML | New