Projeto Orientado a Computação 1
Estratégias de MLOps para o Flautim

Maria Luiza Leão Silva[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brazil

Resumo. *Este documento detalha as etapas e resultados obtidos durante o Projeto Orientado em Computação I (POC1), cujo foco foi o desenvolvimento de um pipeline de Machine Learning Operations (MLOps) para a plataforma Flautim. As soluções propostas foram concebidas com o intuito de integrar CI/CD, versionamento de dados e modelos, treinamento para modelos centralizados ou distribuídos e estratégias de automação para futuras integrações com ferramentas como Argo CD e CML. Este trabalho também explora os desafios e as soluções relacionados ao uso de MLOps em cenários de aprendizado federado, destacando os ganhos em escalabilidade, reprodutibilidade e eficiência operacional.*

1. Introdução

Nos últimos anos, o Aprendizado de Máquina (ML) tem revolucionado a forma como analisamos dados e tomamos decisões, tornando-se uma ferramenta essencial em vários setores. No entanto, a complexidade envolvida na escalabilidade, manutenção e reprodutibilidade de modelos impede que muitas organizações maximizem os benefícios dessa tecnologia. Nesse contexto, MLOps (Machine Learning Operations) surge como um paradigma fundamental, integrando os princípios do DevOps ao ciclo de vida do aprendizado de máquina.

A plataforma Flautim, foco deste trabalho, é uma solução de prototipagem e experimentação de modelos de aprendizado de máquina com ênfase no aprendizado federado (FL). Essa abordagem é especialmente relevante para cenários onde privacidade, descentralização e escalabilidade são cruciais, como em sistemas automotivos. O POC1 foi concebido para adaptar e desenvolver um pipeline de MLOps voltado para o contexto da plataforma Isso é, integração de processos do MLOps no ambiente de prototipação e experimentação de modelos oferecido pela plataforma.

Este relatório é estruturado para apresentar os fundamentos teóricos, a metodologia empregada, os resultados obtidos e os desafios enfrentados, culminando em uma discussão sobre os impactos e perspectivas futuras do projeto.

2. Referencial Teórico

2.1. Machine Learning Operations (MLOps)

O fluxo de MLOps organiza o ciclo de vida de aprendizado de máquina em etapas distintas e interconectadas: preparação dos dados, desenvolvimento, treinamento, validação, integração, deployment e monitoramento. Na etapa de preparação dos dados, o foco está no pré-processamento, versionamento e rastreamento de datasets. No desenvolvimento,

temos a extração de features e o desenvolvimento do modelo. O treinamento e a validação são realizados de forma sistemática, garantindo que os modelos sejam ajustados e avaliados com eficiência. As etapas de integração e deployment garantem que os modelos sejam preparados para ambientes de produção, enquanto o monitoramento contínuo assegura que os modelos implantados mantenham sua precisão e eficácia ao longo do tempo.

2.2. Aprendizado Federado (FL)

Permite o treinamento descentralizado sem compartilhar dados brutos. Este paradigma enfrenta desafios específicos no monitoramento de modelos, na sincronização de dispositivos e na integração de MLOps. O FL se torna essencial em áreas como saúde, onde a privacidade dos dados é prioritária.

2.3. Plataforma Flautim

Um sistema para experimentação de modelos de aprendizado federado. Integra Kubernetes, o framework Flower e GPUs para suportar o treinamento distribuído. A arquitetura modular permite uma escalabilidade eficiente e facilita a adaptação a diferentes cenários de experimentação.

3. Escolha das Tecnologias

A escolha das tecnologias foi guiada pelos objetivos do projeto e pelas demandas específicas do MLOps. Cada ferramenta foi selecionada considerando sua funcionalidade principal e compatibilidade com a infraestrutura da plataforma Flautim:

- Git: Escolhido para controle de versões devido à sua confiabilidade, popularidade e integração com plataformas como GitHub e Gitlab. Ele oferece rastreamento detalhado de mudanças e suporte à colaboração em equipe. Ademais permite a utilização do Github Actions para integração contínua.
- DVC: Selecionado para gerenciar dados, modelos e experimentos em projetos de aprendizado de máquina. Ele integra o controle de versão tradicional (como Git) com capacidades específicas para manipulação de grandes volumes de dados e pipelines.
- CML: Selecionado para automação de fluxos de trabalho e integração contínua, permitindo a execução de experimentos diretamente nos repositórios Git e a geração de relatórios automáticos, promovendo transparência e eficiência.
- Argo CD: Escolhido para auxiliar em futuras estratégias de deployment automatizado, oferecendo integração nativa com Kubernetes e suporte a GitOps, o que alinha bem com os repositórios utilizados no projeto.

4. Solução proposta

O fluxo de MLOps organiza o ciclo de vida do aprendizado de máquina em etapas distintas e interconectadas: preparação dos dados, desenvolvimento, treinamento, validação, integração, deployment e monitoramento. Na solução proposta, as etapas de preparação dos dados, desenvolvimento, treinamento e validação foram implementadas utilizando o DVC em conjunto com o Git. Por meio de pipelines automatizados, uma funcionalidade do DVC definida em arquivos dvc.yaml, essas etapas são configuradas para garantir a reprodutibilidade. Além disso, arquivos de metadados são gerados automaticamente,

facilitando o rastreamento de alterações ao longo do processo. A integração contínua, realizada com o GitHub Actions e o CML, automatiza testes e validações, assegurando a qualidade do pipeline. Já para o deployment e o monitoramento, foi proposto o uso do Argo CD, uma solução capaz de sincronizar o estado desejado, definido no Git, com o ambiente de produção. Essa abordagem também permite um rollback rápido em caso de falhas, garantindo maior segurança e eficiência na entrega de modelos.

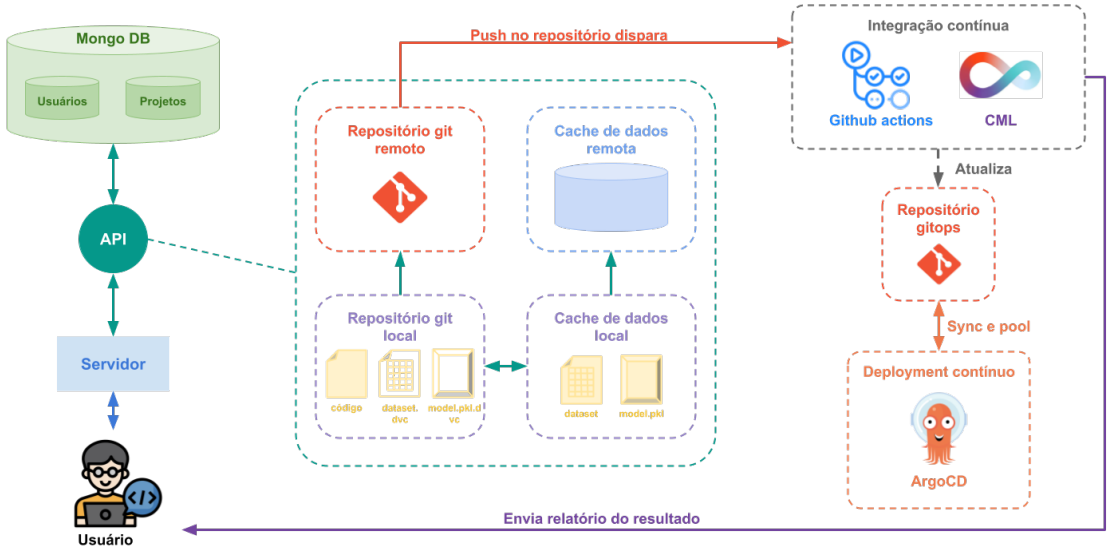

Figura 1. Fluxo MLOps proposto.

O diagrama da solução proposta pode ser observada na Fig.1 e seus componentes serão explicados com mais profundidade a seguir.

4.1. Servidor

A arquitetura do servidor no contexto da solução proposta desempenha o papel de comunicação entre o usuário e as operações relacionadas ao pipeline de MLOps. Ao receber uma requisição em determinada rota, o servidor chama a função da API referente a mesma. Ademais, ele é responsável pela comunicação com o banco de dados (MongoDB) que guarda as relações entre os usuários da plataforma e os seus projetos. Dessa forma, quando um usuário solicita a criação de um novo projeto, ele adiciona mais um projeto ao banco de dados e o associa ao usuário que fez a solicitação. Isso é fundamental para o funcionamento do fluxo pois permite recuperar o caminho para o diretório do sistema em que está o projeto. Isso pois os projetos são armazenados no sistema utilizando o caminho [id do usuário]/[id do projeto].

4.2. API para DVC e Git

No fluxo de MLOps proposto, a API desempenha um papel central ao facilitar e automatizar as operações relacionadas ao DVC e Git, tornando o gerenciamento de projetos de aprendizado de máquina simples e eficiente. Essa API foi projetada especificamente para atender às necessidades do ambiente de experimentação e prototipação da plataforma Flautim, permitindo aos usuários criar, configurar e monitorar pipelines de forma intuitiva e reprodutível.

4.2.1. Criação e Configuração de Projetos

Quando um usuário do Flautim solicita a criação de um novo projeto, a API é acionada para inicializar a estrutura necessária. Isso inclui a configuração do ambiente do projeto, a inicialização do Git e do DVC no diretório designado, bem como a criação de uma base sólida para rastreamento de dados e código. A integração automatizada desses componentes reduz a complexidade inicial para o usuário e garante que o projeto esteja pronto para uso imediato.

4.2.2. Pipelines Automatizados e Flexíveis

A API fornece ferramentas para que os usuários definam e configurem pipelines personalizados, que estruturam e automatizam o fluxo de trabalho de aprendizado de máquina. Cada pipeline consiste em estágios interconectados, como pré-processamento de dados, extração de características, treinamento e validação de modelos. Esses estágios são descritos em arquivos dvc.yaml e podem ser configurados diretamente por meio de endpoints oferecidos pela API.

Além disso, a API suporta a inclusão de novas etapas no pipeline, conectando automaticamente os dados, o código e os recursos necessários para a execução. Isso permite que os usuários ajustem e expandam facilmente seus fluxos de trabalho conforme necessário, mantendo a reprodutibilidade e o rastreamento detalhado de cada alteração.

4.2.3. Experimentação Orientada por Pipelines

No contexto experimental e de prototipação da plataforma Flautim, os pipelines desempenham um papel crucial. A API permite aos usuários executar experimentos diretamente nos pipelines definidos, possibilitando a variação de hiperparâmetros, ajustes no código ou nos dados, e a comparação sistemática dos resultados.

Esses experimentos são gerenciados rastreados pela DVC, garantindo que cada execução seja documentada e vinculada ao seu conjunto de condições. Essa abordagem fornece transparência e facilita a identificação de configurações mais eficazes, tornando o ambiente ideal para inovação rápida e validação de hipóteses.

4.2.4. Gerenciamento de Dados e Versionamento

Um dos principais recursos da API é o rastreamento e versionamento de dados. Por meio de endpoints específicos, os usuários podem adicionar conjuntos de dados ao projeto e vinculá-los aos diferentes estágios do pipeline. Isso assegura que todas as mudanças sejam capturadas, permitindo a reprodutibilidade total dos experimentos.

4.2.5. Integração com Sistemas de CI/CD e Deployment

Após validar os experimentos, a API auxilia na promoção dos modelos para produção. Ela integra os pipelines com sistemas de CI/CD, como o GitHub Actions e o Argo CD, permi-

tindo que os modelos sejam implantados de forma contínua e confiável. Essa integração acontece, principalmente, pelo use do Git pela API.

4.2.6. Benefícios no Contexto do Flautim

Para os usuários do Flautim, essa API proporciona uma experiência fluida e eficiente, permitindo:

- **Automação do Fluxo de Trabalho:** Com pipelines totalmente integrados e configuráveis.
- **Facilidade na Experimentação:** Execução e rastreamento simplificados de experimentos, otimizando o tempo de prototipação.
- **Escalabilidade e Reprodutibilidade:** Garantia de que cada estágio e experimento possa ser replicado em diferentes contextos.

Em suma, a API vai além de simplificar o gerenciamento de projetos de aprendizado de máquina. Ela oferece uma infraestrutura robusta e adaptável, permitindo que cientistas de dados e desenvolvedores foquem na inovação e na criação de modelos mais eficazes, aproveitando ao máximo os recursos da plataforma.

4.3. Cache de Dados Local

O cache de dados local é gerenciado pelo DVC e serve como uma área intermediária para armazenar dados utilizados no pipeline, como conjuntos de treinamento, resultados intermediários e modelos treinados. Ele é mantido no disco local e vinculado ao repositório Git por meio de metadados.

Funções principais:

- Evitar redundância, garantindo que apenas uma cópia de cada versão dos dados seja armazenada localmente.
- Melhorar o desempenho ao evitar transferências frequentes entre o sistema local e o armazenamento remoto.
- Rastrear e versionar mudanças nos dados de forma eficiente.

4.4. Cache de Dados Remoto

O cache de dados remoto é uma solução de armazenamento utilizada para compartilhar e arquivar dados entre diferentes membros da equipe ou ambientes. Ele é sincronizado com o cache local por meio de comandos do DVC, como dvc push e dvc pull.

Funções principais:

- Atuar como uma fonte central para armazenar grandes volumes de dados, reduzindo o uso de espaço local.
- Facilitar a colaboração, permitindo que diferentes desenvolvedores acessem as mesmas versões de dados a partir de locais diferentes.
- Garantir que os dados estejam disponíveis mesmo se o cache local for perdido ou danificado.

4.5. Repositório git local

O repositório Git local é a cópia local do repositório do projeto, armazenada no computador do desenvolvedor. Ele é utilizado para versionar o código-fonte, as configurações dos pipelines e as definições de estágios (como no arquivo dvc.yaml). No contexto do MLOps, também rastreia os metadados gerados pelo DVC, como arquivos .dvc que representam o controle sobre os dados e modelos.

Funções principais:

- Permitir que o desenvolvedor trabalhe de forma isolada e faça mudanças locais no projeto.
- Garantir o versionamento do código, facilitando o rastreamento de alterações e a reprodutibilidade do pipeline.
- Sincronizar alterações com o repositório Git remoto para compartilhamento com outros colaboradores.

4.6. Repositório git remoto

O repositório Git remoto é um serviço centralizado onde o código e os metadados do projeto são armazenados. Ele serve como o ponto principal de colaboração entre desenvolvedores e integrações de CI/CD.

Funções principais:

- Permitir a sincronização de alterações entre diferentes cópias locais do projeto, garantindo que todos os desenvolvedores trabalhem na mesma base de código.
- Servir como um local de backup para o repositório Git local, protegendo o histórico do projeto contra falhas locais.
- Integrar-se a ferramentas de CI/CD para automatizar a execução de workflows, testes e implantação de modelos.

4.7. Integração contínua

A integração contínua (CI) é um componente essencial para garantir a automação, reprodutibilidade e qualidade no desenvolvimento de projetos de aprendizado de máquina. No contexto da solução proposta, o GitHub Actions, em conjunto com o Continuous Machine Learning (CML), desempenha um papel central na implementação de CI.

4.7.1. Gatilhos para Automação

Os workflows configurados no GitHub Actions são ativados automaticamente por eventos como push, pull request ou alterações específicas nos dados e código. Esses gatilhos garantem que qualquer modificação seja imediatamente testada e integrada ao pipeline, eliminando atrasos e minimizando erros. A API atua em conjunto, gerenciando os estágios e conectando os dados e experimentos ao fluxo, garantindo que todas as etapas estejam preparadas para integração contínua.

4.7.2. Integração com DVC

Dentro dos workflows do GitHub Actions, o CML é usado para executar experimentos definidos pelo DVC. A API simplifica essa integração, configurando automaticamente os pipelines e rastreando os experimentos vinculados aos dados e modelos. Isso possibilita que os resultados dos experimentos sejam versionados e registrados de forma estruturada, assegurando a continuidade e a rastreabilidade de cada modificação feita no pipeline.

4.7.3. Geração de Relatórios

O CML, em colaboração com o GitHub Actions e a API, cria relatórios detalhados em formato Markdown, que são anexados diretamente a pull requests. Esses relatórios incluem:

- Gráficos de desempenho;
- Tabelas com métricas relevantes;
- Insights importantes para revisão colaborativa.

4.7.4. Benefícios no Contexto do Flautim

A integração entre o GitHub Actions e a API potencializa o fluxo de trabalho no Flautim, automatizando tarefas complexas, como:

- Configuração de pipelines diretamente a partir de solicitações feitas pelos usuários;
- Execução de testes em pipelines de forma automatizada, com base nos parâmetros definidos pela API;
- Registro e versionamento dos resultados de experimentos realizados.

Essa colaboração permite que o GitHub Actions monitore continuamente as mudanças realizadas no repositório Git, enquanto a API coordena os processos subjacentes, como a execução de estágios no pipeline e a validação de modelos. Dessa forma, o ciclo de vida do aprendizado de máquina é otimizado, desde a prototipação até a produção.

4.8. Deployment contínuo e monitoramento

O deployment contínuo (CD) é uma prática essencial que automatiza o processo de entrega de modelos para ambientes de produção, garantindo que alterações aprovadas sejam implementadas de forma consistente e confiável. No contexto da solução proposta, o Argo CD foi escolhido como a ferramenta principal para gerenciar esse fluxo, devido à sua compatibilidade com o Kubernetes e sua abordagem baseada no GitOps. Essa integração permite que os modelos validados e aprovados sejam implantados automaticamente, assegurando um pipeline de entrega robusto e eficiente, ao mesmo tempo que facilita o monitoramento contínuo dos deployments.

O Argo CD funciona em conjunto com ferramentas como GitHub Actions e CML, criando um ecossistema integrado que automatiza desde a validação de modelos até sua implantação em ambientes reais. Essa arquitetura permite que os modelos estejam sempre prontos para uso, com menos necessidade de intervenções manuais, tornando o processo mais ágil e seguro.

4.8.1. Monitoramento Contínuo de Repositórios Git

O Argo CD realiza o monitoramento contínuo dos repositórios Git configurados, observando alterações em arquivos de configuração, pipelines e outros componentes críticos. Ele garante que o estado do ambiente de produção reflita sempre o estado desejado definido no repositório Git. Essa abordagem reduz discrepâncias e melhora a confiabilidade ao eliminar inconsistências entre os estados declarados e reais da infraestrutura.

4.8.2. Sincronização Automática

As mudanças aprovadas no repositório Git, como atualizações em configurações de infraestrutura, novos modelos treinados ou ajustes nos pipelines, são automaticamente sincronizadas com o ambiente de produção. Esse processo minimiza a necessidade de ações manuais, reduzindo o risco de erros humanos e acelerando a implementação de novas versões.

4.8.3. Rollback Simplificado

O Argo CD mantém um histórico detalhado de todas as alterações aplicadas, permitindo um rollback rápido para versões anteriores em caso de problemas. Essa funcionalidade é particularmente útil para garantir a continuidade operacional e evitar impactos negativos decorrentes de falhas em atualizações ou implantações.

4.8.4. Visualização do Estado do Sistema

A interface gráfica do Argo CD oferece uma visão clara e interativa do estado atual dos aplicativos e recursos implantados. Ela destaca discrepâncias entre o estado desejado (definido no repositório Git) e o estado real no ambiente de produção. Isso permite que os desenvolvedores e operadores identifiquem rapidamente problemas e tomem medidas corretivas de forma proativa.

4.8.5. Benefícios no Contexto do Flautim

No ambiente experimental e de prototipação do Flautim, a integração do Argo CD com o restante da solução é crucial para manter a consistência e a confiabilidade durante o desenvolvimento e implantação de modelos. A capacidade de automatizar todo o processo de deployment e monitoramento oferece diversas vantagens, como:

- Redução do Tempo de Entrega: Alterações são aplicadas rapidamente, permitindo iterações frequentes e maior agilidade no desenvolvimento.
- Maior Segurança e Controle: O monitoramento contínuo e a possibilidade de rollback garantem que erros possam ser rapidamente corrigidos sem comprometer o sistema.
- Facilidade de Escalabilidade: A integração com Kubernetes permite gerenciar recursos de forma eficiente, mesmo em ambientes distribuídos, como os exigidos pelo aprendizado federado.

Em resumo, o deployment contínuo com o Argo CD, em conjunto com GitHub Actions e CML, forma uma base sólida para garantir a entrega eficiente e monitorada de modelos no Flautim, facilitando tanto a experimentação quanto a transição para produção.

5. Protótipo da solução

O protótipo da solução desenvolvida consiste em um conjunto de arquivos Python que integram as principais funcionalidades necessárias para implementar o pipeline de MLOps no contexto do Flautim. Esses arquivos foram projetados para gerenciar operações de versionamento, rastreamento, execução de experimentos e automação, utilizando Git, DVC, e outras ferramentas modernas de MLOps. A seguir, detalham-se os principais componentes do protótipo:

5.1. Configurações

A implementação do protótipo foi realizada fora do ambiente da plataforma Flautim, utilizando o Windows Subsystem for Linux (WSL) em uma máquina pessoal. Esse ambiente exigiu a configuração de dependências específicas para viabilizar o desenvolvimento e a execução do pipeline proposto. As configurações envolveram a instalação de bibliotecas, o uso de cache de dados remota, a criação de um banco de dados MongoDB e a integração com GitHub Actions.

5.1.1. Instalação de Dependências

Foi necessário configurar o ambiente no WSL para suportar ferramentas como DVC, Git, Python (com as bibliotecas exigidas para o desenvolvimento) e os serviços relacionados, como o MongoDB e o GitHub Actions. Essas ferramentas são os componentes centrais para a execução do protótipo e garantem que as operações do pipeline possam ser reproduzidas e monitoradas corretamente.

5.1.2. Cache de dados remoto

O DVC permite o uso de diferentes sistemas de armazenamento para o cache de dados, incluindo diretórios locais, sistemas NAS e serviços externos. Para o protótipo, foi configurado um diretório local na raiz do sistema (/tmp/dvcstore) como cache de dados remota. Essa configuração foi essencial para armazenar e rastrear os dados utilizados nos experimentos, permitindo que as alterações fossem registradas e acessíveis durante o desenvolvimento.

A escolha de um diretório local simplifica o processo e demonstra a capacidade do DVC de gerenciar dados em diferentes tipos de armazenamento, enquanto oferece um fluxo de trabalho escalável para cenários mais complexos.

5.1.3. MongoDB

O banco de dados MongoDB foi configurado utilizando o serviço MongoDB Atlas, que oferece um cluster na nuvem para gerenciamento eficiente e seguro de dados. Essa

configuração foi utilizada para armazenar informações relacionadas aos usuários, projetos, experimentos e resultados. O MongoDB Atlas permite escalabilidade e fornece ferramentas de monitoramento integradas, garantindo que os dados do protótipo fossem gerenciados de forma centralizada e acessível.

5.1.4. Github

O GitHub foi configurado como o repositório remoto principal para o armazenamento e gerenciamento dos projetos. Ele desempenha um papel essencial no protótipo, permitindo que os projetos criados localmente sejam sincronizados com o repositório remoto, o que facilita o versionamento e o rastreamento de mudanças no código e nos pipelines. A integração com o GitHub garante que os dados e o código estejam sempre disponíveis para colaboração e revisão, mesmo fora do ambiente local.

Além de atuar como repositório remoto, o GitHub Actions foi utilizado para automatizar tarefas relacionadas à integração contínua. Por meio de workflows configurados no GitHub Actions, o código, os pipelines e os experimentos podem ser validados e executados automaticamente sempre que uma modificação é feita no repositório.

Um self-hosted runner foi configurado para executar os workflows do GitHub Actions diretamente na máquina local utilizada no protótipo. Esse runner permite que o Github Actions consiga acessar o conteúdo que o DVC armezana no cache de dados remoto, que nesse caso é um diretorio local.

5.2. API

A API foi implementada utilizando a biblioteca FastAPI, conhecida por sua performance e facilidade de uso, juntamente com o Uvicorn, um servidor ASGI de alto desempenho. Esses componentes garantem uma API eficiente, escalável e de rápida resposta, ideal para as necessidades do protótipo. O foco do desenvolvimento foi a integração de funcionalidades essenciais para o gerenciamento de projetos de aprendizado de máquina centralizado, utilizando ferramentas como Git, DVC e Python.

5.2.1. Funcionalidades Implementadas

Uma das funcionalidades centrais da API é a criação de novos projetos, permitindo inicializar um ambiente de trabalho completamente configurado. Isso é feito com a execução de `git init` para versionar o código, `dvc init` para habilitar o rastreamento de dados e a realização de um `git commit` inicial contendo a configuração básica. Essa abordagem garante que os projetos sejam versionados desde o início, fornecendo uma base consistente para o desenvolvimento.

Outra funcionalidade importante é a adição de novos dados e arquivos ao repositório. A API facilita esse processo por meio de comandos como `dvc get_url`, que permite importar arquivos de fontes externas, e `dvc add`, que rastreia os dados localmente. Além disso, alterações são versionadas no Git utilizando `git add` e `git commit`, garantindo que todas as mudanças sejam documentadas e integradas ao fluxo de trabalho.

A API também possibilita a definição de estágios no pipeline de aprendizado de máquina, configurando etapas como pré-processamento, treinamento e validação. Isso é realizado com o uso de `dvc stage add`, que conecta dados, código e parâmetros necessários para cada etapa. Os arquivos gerados (dvc.yaml) são versionados no Git para assegurar reprodutibilidade e rastreamento.

A reprodução do pipeline é outra funcionalidade essencial oferecida pela API. Com o uso do comando `dvc repro`, os usuários podem executar novamente todo o pipeline ou apenas etapas específicas, garantindo que as mudanças realizadas sejam testadas e documentadas de maneira transparente.

A visualização de métricas também é suportada pela API, permitindo aos usuários acessar informações detalhadas sobre o desempenho do pipeline. Com o comando `dvc metrics show`, os resultados são apresentados em tabelas ou formatos como JSON, enquanto `dvc plots show` gera gráficos para análises visuais. Esses recursos são fundamentais para validar experimentos e interpretar resultados.

Por fim, a API oferece suporte para a execução de experimentos com o uso do comando `dvc exp run`. Essa funcionalidade possibilita ajustar hiperparâmetros, explorar diferentes configurações e comparar resultados de forma eficiente. Isso é especialmente valioso no contexto do Flautim, onde o foco está em prototipação e experimentação rápida.

5.2.2. Base para Expansão

O protótipo desenvolvido fornece uma base funcional que integra as etapas fundamentais do MLOps no aprendizado de máquina centralizado. Ele foi projetado para ser facilmente adaptado a futuras melhorias, incluindo a expansão para aprendizado federado, a sincronização de dispositivos e a gestão de dados descentralizados. Além disso, sua arquitetura pode ser integrada com ambientes de produção, automatizando monitoramento contínuo e suportando escalabilidade em sistemas distribuídos.

Mais detalhes técnicos sobre a implementação do protótipo, assim como seu código-fonte, estão disponíveis em `https://github.com/MarialuizaLeao/dvc_git_server`.

5.3. Integração contínua

A implementação inicial da integração contínua foi prototipada com o objetivo de validar o fluxo proposto para automação de tarefas relacionadas ao treinamento de modelos e avaliação de desempenho. Essa validação foi realizada por meio de um workflow simples no GitHub Actions, configurado para ser executado automaticamente sempre que o repositório do projeto recebesse um `push`.

O workflow incluiu o uso do CML (Continuous Machine Learning) para realizar o treinamento do modelo, a geração de métricas de avaliação e a criação de relatórios automáticos. O CML foi utilizado para gerar arquivos em formato Markdown contendo gráficos, tabelas de métricas e insights dos resultados. Essa funcionalidade proporcionou uma revisão colaborativa mais eficiente, permitindo que os resultados fossem analisados diretamente no repositório.

Apesar de demonstrar o funcionamento básico do fluxo, a integração entre as funcionalidades da API e o workflow ainda não foi completamente implementada. Por esse motivo, a construção do workflow, as configurações e a análise dos resultados foram realizadas de forma manual, com o CML complementando o processo de validação.

Mesmo com essas limitações, os experimentos realizados destacaram a viabilidade do fluxo proposto, especialmente no uso do CML para automatizar partes essenciais do pipeline. Essa validação inicial demonstrou que, com maior integração entre a API e os workflows, é possível criar um sistema totalmente automatizado, reduzindo a necessidade de intervenções manuais e aumentando a eficiência no desenvolvimento, validação e implantação de modelos.

5.4. Deployment contínuo

A prototipação do deployment contínuo não foi realizada devido à complexidade e às limitações do ambiente de implementação utilizado. O funcionamento do Argo CD requer a configuração de um cluster Kubernetes para gerenciar a implantação automatizada e a sincronização contínua do estado desejado, definido nos repositórios Git, com o ambiente de produção. No entanto, o ambiente local baseado em WSL utilizado para o protótipo não oferecia suporte adequado para a configuração do Kubernetes necessário, impossibilitando a execução dessa etapa.

Embora a implementação do deployment contínuo não tenha sido possível no protótipo, os fluxos e conceitos definidos permanecem viáveis e prontos para serem aplicados em um ambiente mais robusto, com infraestrutura que suporte as exigências do Kubernetes e do Argo CD.

6. Futuras alterações e melhorias

O protótipo desenvolvido apresenta uma base funcional para o fluxo de MLOps, mas há diversas oportunidades de melhorias e expansões que podem torná-lo mais robusto e adequado para cenários mais complexos, como o aprendizado federado. As principais direções futuras incluem:

6.1. Suporte a Outras Funcionalidades do DVC

O DVC oferece diversas funcionalidades que ainda não foram exploradas no protótipo atual. Melhorias futuras incluem a integração de recursos como a comparação detalhada de experimentos com `dvc exp diff`, o uso de `dvc push` e `dvc pull` para gerenciar repositórios remotos de dados em ambientes distribuídos, e a utilização de dvc lock para rastreamento avançado de pipelines. Esses aprimoramentos aumentariam significativamente a flexibilidade e a capacidade do sistema de lidar com diferentes fluxos de trabalho e cenários de gerenciamento de dados.

6.2. Melhor Integração entre a API e a Parte de Integração Contínua

A integração entre a API e os workflows de CI/CD pode ser aprimorada para permitir uma automação mais eficiente. Isso inclui a geração dinâmica de workflows do GitHub Actions diretamente pela API, eliminando a necessidade de configurações manuais. Além disso, a API pode ser estendida para gerenciar diretamente os resultados dos experimentos executados pelos workflows, criando um fluxo contínuo e sincronizado entre a definição de pipelines, a execução de experimentos e a análise dos resultados.

6.3. Implementação do Deployment Contínuo

A implementação do deployment contínuo com Argo CD continua sendo uma prioridade para alinhar o fluxo implementado com as práticas de MLOps. Futuramente, a configuração de um cluster Kubernetes será necessária para permitir a integração do Argo CD, possibilitando a automação completa do processo de entrega de modelos em ambientes de produção. Isso incluiria a sincronização contínua do estado desejado e o suporte a rollback automático em caso de falhas, aumentando a confiabilidade do sistema.

6.4. Adaptação para o Fluxo de Aprendizado Federado

Um dos principais objetivos futuros é expandir o protótipo para suportar fluxos de aprendizado federado, alinhando-se aos requisitos da plataforma Flautim. Isso exigirá a adaptação do fluxo de MLOps para incluir a sincronização de modelos em dispositivos distribuídos, o gerenciamento de dados locais e a agregação de parâmetros utilizando frameworks como o Flower. Além disso, a integração com ferramentas de monitoramento será essencial para garantir a consistência e a eficiência dos modelos em um ambiente descentralizado.

7. Conclusão

Este trabalho detalhou as etapas de desenvolvimento de um pipeline de MLOps integrado à plataforma Flautim, com foco em modelos centralizados. O projeto abordou questões fundamentais como a automação do ciclo de vida de aprendizado de máquina, o versionamento de dados e modelos, e a integração de fluxos contínuos de trabalho, alinhando-se às exigências do aprendizado de maquina.

O uso das ferramentas escolhidas, como Git, DVC e CML, foi essencial para garantir a rastreabilidade e reprodutibilidade dos experimentos. A aplicação de conceitos modernos de MLOps permitiu a construção de pipelines automatizados e personalizáveis, facilitando a execução de experimentos em um ambiente controlado e reproduzível.

Os desafios enfrentados ao longo do projeto foram abordados com soluções que balanceiam eficiência e simplicidade, destacando a importância de uma arquitetura modular e de ferramentas que promovem a colaboração entre usuários e o gerenciamento ágil dos experimentos. Por fim, a conclusão deste projeto evidencia a importância de metodologias sistemáticas para o aprendizado de máquina, oferecendo uma base sólida para estudos futuros e aplicações em larga escala.

8. Referências

[1] Han BASIL. Data versioning for cd4ml - part 2, 2021. Accessed: 2024-11-05.

[2] Qi Cheng and Guodong Long. Federated learning operations (flops): Challenges, lifecycle and approaches. In 2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), pages 12–17, 2022.

[3] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, P.K. Gupta, Somya Garg, and Saransh Ahalawat. On continuous integration / continuous delivery for automated deployment of machine learning models using mlops. In 2021 IEEE Fourth International Conference on Artificial Inetelligence and Knowledge Engineering (AIKE), pages 25–28, 2021.

[4] Alice Johnson. Devops pipelines for federated learning: Implementing mlops in decentralized machine learning systems. Journal of Artificial Intelligence Research and Applications, 4(2):77–83, 2024. Accessed: 2024-11-05.

[5] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. IEEE Access, 2023.

[6] Sasu Makinen, Henrik Skogstrom, Eero Laaksonen, and Tommi Mikkonen. Who needs mlops: What data scientists seek to accomplish and how can mlops help? In 2021 IEEE First Workshop on AI Engineering - Software Engineering for AI (WAIN), pages 109–112, 05 2021.

[7] Alexander Malyuk. Flops: Practical federated learning via automated orchestration (on the edge). Master's thesis, Technical University of Munich, 2024. Accessed: 2024-11-05.

[8] Danilo Sato, Arif Wider, and Christoph Windheuser. Continuous delivery for machine learning, 2019. Accessed: 2024-11-05.

[9] TensorOpera. Introducing fedml octopus: scaling federated learning into production with simplified mlops, 2023. Accessed: 2024-11-05.

[10] David Valdez, Ardalan Mehraram, Adrien Debray, Christoforos Trakas, Maciej Piotrowski, and Johannes Lootens. Applying a mlops approach to federated learning using ml flow with nv flare: A healthcare use case, 2023. Accessed: 2024-11-05