



MLOps - Transformando Teoria em Prática

Arthur Quintella de Mello Olga - arthurqmo@al.insper.edu.br
Gabriel Lopes Monteiro - gabriellm1@al.insper.edu.br
Guilherme Peres Leite - guilhermep13@al.insper.edu.br
Vinicius Gomes de Lima - viniciusgl1@al.insper.edu.br

Trabalho de Conclusão de Curso

**Relatório
Versão Final
do Projeto Final de Engenharia**

**São Paulo-SP
Junho 2021**

**Arthur Quintella de Mello Olga
Gabriel Lopes Monteiro
Guilherme Peres Leite
Vinicius Gomes de Lima**

MLOps - Transformando Teoria em Prática

Relatório Final do Projeto Final de Engenharia

Relatório apresentado ao curso de Engenharia, como requisito para o Trabalho de Conclusão de Curso.

Professor Orientador: Prof. Fábio José Ayres

Mentores na Empresa: Fabrício Barth, Alan Braz

Coordenador TCC/PFE: Prof. Dr. Luciano Pereira Soares

**São Paulo - SP
Junho 2021**

**Arthur Quintella de Mello Olga
Gabriel Lopes Monteiro
Guilherme Peres Leite
Vinicius Gomes de Lima**

MLOps - Transformando Teoria em Prática

Projeto Final de Engenharia apresentado ao programa Graduação em Engenharia da Computação como requisito parcial para a obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Fabio José Ayres

Banca Examinadora

Fabio José Ayres

Inspere

Igor dos Santos Montagner

Inspere

Raul Ikeda Gomes da Silva

Inspere

Sumário

RESUMO	5
1. INTRODUÇÃO	6
1.1. ESCOPO DO PROJETO	7
1.2. RECURSOS	8
1.3. CRONOGRAMA	8
1.4. MAPEAMENTO DOS STAKEHOLDERS	9
1.5. RISCOS ENVOLVIDOS	10
1.6. QUESTÕES ÉTICAS	10
1.7. REVISÃO DO ESTADO DA ARTE	10
2. METODOLOGIA	13
2.1 PESQUISA DE FERRAMENTAS	14
2.1.1. IBM WATSON	15
2.1.2. FEAST	16
2.1.3. DVC E CML	17
2.1.4. KUBEFLOW	18
2.1.5. AMAZON SAGEMAKER	19
2.2. ANÁLISE DAS INFORMAÇÕES COLETADAS	19
2.2.1. CENÁRIO 1: STARTUPS E EMPRESAS PEQUENAS	19
2.2.2. CENÁRIO 2: EMPRESA MÉDIA OU TIME PEQUENO DE GRANDE CORPORACÃO	20
2.2.3. CENÁRIO 3: PROJETOS DE GRANDE PORTE	22
2.3. PROPOSTA INICIAL DE GUIA	25
3. RESULTADOS	27
3.1. EXEMPLO END-TO-END	27
3.2. CONSTRUÇÃO DO GUIA	31
3.2.1. INTRODUÇÃO AOS PRINCÍPIOS DE MLOPS	32
3.2.2. GUIA DE IMPLEMENTAÇÃO	33
3.2.3. FERRAMENTAL ADICIONAL DO SITE	35
3.3. VALIDAÇÃO DO GUIA	36
4. CONSIDERAÇÕES FINAIS	37
REFERÊNCIAS	39
APÊNDICE A – ESTRUTURA DO GUIA	41
APÊNDICE B – RESPOSTAS AO FORMULÁRIO DE VALIDAÇÃO DO GUIA	43

RESUMO

Este projeto tem como objetivo desenvolver um guia de implementação de MLOps. O termo MLOps refere-se ao conjunto de práticas e ferramentas para colaboração e comunicação entre cientistas de dados e profissionais de operações, além da automatização da transferência de modelos de aprendizado de máquina do ambiente de desenvolvimento para o de produção. Adicionalmente, tem-se por objetivo a construção de um *pipeline* funcional *end-to-end* como prova de conceito, em suporte ao guia de implementação. O guia tem em foco profissionais já familiares com conceitos de *machine learning* e operações e é composto por um tutorial prático que passa por todo o ciclo de desenvolvimento e *deployment* de modelos utilizando práticas de MLOps para aumentar sua automação, qualidade e reprodutibilidade. O ferramental utilizado no tutorial é baseado em produtos de *machine learning* da IBM combinado com outras ferramentas que possam contribuir com a execução de projetos de clientes que procuram melhorar seu ciclo de desenvolvimento através da aplicação da metodologia de MLOps.

Palavras-chave: Machine Learning; MLOps; Guia prático; IBM Watson; CI/CD; *Pipeline* de Dados;

1. INTRODUÇÃO

A IBM é uma das empresas mais tradicionais no setor de tecnologia da informação. Oferece diversas soluções e ferramentas em múltiplas áreas, dentre elas: computação em nuvem, inteligência artificial, segurança e infraestrutura, com foco em parcerias B2B. Dentro da área de inteligência artificial são 36 produtos ofertados, com destaque para a *suite* IBM Watson de ferramentas para aplicações de *machine learning* (aprendizado de máquina), processamento de linguagem natural, reconhecimento de fala e visão computacional. Além de possuir uma plataforma para o desenvolvimento de aplicações de *machine learning*, a *suite* IBM Watson também possui ferramentas que possibilitam colocar aplicações e modelos em produção através do IBM Cloud.

No entanto, ainda existe uma distância muito grande entre a teoria de *machine learning* e a sua aplicação na construção de produtos reais. Poucas empresas adotam uma metodologia clara acerca do desenvolvimento e operacionalização de produtos envolvendo *machine learning*, especialmente em casos nos quais o produto deve ser continuamente aperfeiçoado (e.g. aplicações *online*).

Pode-se comparar o cenário atual de desenvolvimento de produtos envolvendo *machine learning* com o cenário de desenvolvimento de software antes da metodologia DevOps. Em uma equipe de *machine learning*, se os profissionais de ciência dos dados não trabalham de forma conectada com a equipe de operações, todo o processo desde o treinamento até o modelo ser colocado em produção acaba sendo demorado e ineficiente. Quando as várias equipes de desenvolvimento de um produto trabalham de forma isolada, configura-se uma *pipeline* na qual uma equipe completa sua porção do projeto e passa os resultados adiante para outras equipes (*handover*). Esta situação pode gerar grande ineficiência: as diferentes equipes possuem diferentes motivações, e essa discrepância é exacerbada ao trabalharem de forma desconexa, sem acompanhar todo o processo de desenvolvimento do produto. Além disso, a falta de monitoramento eficaz dos dados e dos modelos dificulta correções quando o modelo não performa como esperado.

Lidar com essas questões é de grande interesse tanto dos provedores de produtos e serviços quanto dos clientes. Em 2017, estima-se que o setor de *machine learning* movimentou cerca de 1,4 bilhão de dólares, com projeções de crescimento de 8,8 bilhões em 2022 e 117 bilhões até 2027 [1].

O presente projeto tem como objetivo construir um guia para profissionais que estejam inseridos em um projeto de *machine learning*, para que possam utilizar as ferramentas disponíveis e aplicar as melhores práticas da metodologia de MLOps. Posto que o espectro de escalas de desenvolvimento de produtos de *machine learning* é muito amplo (desde pequenas *startups* até grandes corporações), também é objetivo deste projeto definir claramente um contexto de aplicação e construir o guia de práticas de MLOps para esse contexto.

Para a IBM, este guia pode capacitar seus clientes na utilização de seus produtos, possibilitando resultados ainda mais satisfatórios em seus projetos, assim como atrair novos clientes para a empresa. A IBM forneceu acesso aos principais produtos do IBM Watson, além de mentoria técnica e *know-how* adquirido em projetos passados que tenham afinidade com o projeto presente.

O guia desenvolvido neste projeto permite que uma empresa (pertencente ao contexto definido para o guia) compreender e implementar as práticas de MLOps, explicando de forma clara e didática o funcionamento e aplicação das várias etapas de seu fluxo, como: automação, versionamento, testagem, fluxo contínuo de desenvolvimento e monitoramento dos dados e modelos nos projetos. Desta forma, a aplicação integral ou pontual deste guia pode trazer benefícios dessa prática como diminuir o tempo do ciclo de desenvolvimento, diminuição do custo, maior reprodutibilidade e um produto final mais robusto.

1.1. Escopo do projeto

O escopo do projeto se limitou a:

- Pesquisar as ferramentas utilizadas em um fluxo de MLOps e seu funcionamento
- Definir um contexto de aplicação de MLOps
- Implementar um fluxo de MLOps simples.
- Desenvolver um guia explicando as melhores práticas e ferramentas para cada etapa de MLOps.

Não fez parte do escopo do projeto:

- Ensinar *machine learning*.
- Ensinar *cloud computing*.

Os custos estimados para o desenvolvimento do projeto foram:

- Cerca de R\$125,00 por hora de trabalho de um engenheiro de dados senior (mentor técnico)
- IBM Watson Studio - US\$396/mês (fornecido pela IBM).
- IBM Watson OpenScale - US\$250/modelo/mês (fornecido pela IBM).
- IBM Virtual servers(SP) - US\$0.10/hora (fornecido pela IBM).
- IBM Machine Learning - US\$ 5.00/hora (fornecido pela IBM).

O projeto pressupõe:

- Acesso aos recursos do IBM Watson.
- Acesso a mentoria técnica da IBM.
- Mínimo de 20 horas semanais por aluno.

1.2. Recursos

Para a confecção deste projeto foram necessários, além de recursos individuais do grupo (computadores, acesso à internet, entre outros), acesso à infraestrutura do IBM Watson. Mais especificamente, IBM Watson Studio, IBM Watson Machine Learning, IBM Watson OpenScale, além de recursos adicionais da IBM Cloud. Além disso, foi acordado a disponibilidade mínima de 20 horas semanais por aluno, além de pelo menos uma reunião semanal de auxílio com o professor orientador e o mentor técnico.

1.3. Cronograma

Considerando o escopo do projeto, as tarefas são delineadas no cronograma apresentado na Tabela 1.

Tabela 1 - Cronograma do Projeto

Atividades e Entregas	Março				Abril				Maio				Junho			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Levantamento Bibliográfico	■	■														
Entrega Relatório Preliminar		■														
Ambientação com IBM Watson		■	■													
Elaboração do escopo do tutorial				■	■											
Relatório/Banca Intermediária						■	■									
Desenvolvimento do <i>pipeline</i>							■	■	■							
Documentação do tutorial									■	■	■					
Relatório/Banca Final														■	■	
Apresentação para IBM															■	

Fonte: Os autores (2021).

1.4. Mapeamento dos *stakeholders*

Dado o escopo do projeto, os *stakeholders* estão definidos abaixo na Tabela 2.

Tabela 2 - Stakeholders do Projeto

Stakeholder	Posição	Papel no Projeto	Expectativas
Alan Braz	Engenheiro da IBM	Mentor técnico	Ter um guia MLOps capaz de capacitar potenciais clientes e interessados
Clientes da IBM	Usuários	Usuários	Aplicar a metodologia MLOps em seus projetos
Fabio Ayres	Professor	Orientador	Auxiliar no desenvolvimento de um trabalho que seja de agrado da empresa parceira e dentro dos conformes do PFE

Fonte: Os autores (2021).

1.5. Riscos Envolvidos

Existem riscos que poderiam acarretar em problemas ao longo do projeto, de tal forma que o resultado final poderia ter sido comprometido, sendo eles:

- Falta de disponibilidade de recursos da IBM.
- Falta de objetividade da equipe na definição dos tutoriais e material correlato.
- Rápida evolução do ferramental da área pode atrapalhar a construção de um tutorial efetivo em tempo hábil.

1.6. Questões Éticas

É importante levar em consideração a segurança dos dados e criar procedimentos compatíveis com a Lei Geral de Proteção de Dados Pessoais (LGPD). Além disso, é preciso ressaltar a importância do monitoramento de modelos em produção para evitar comportamentos discriminatórios por parte dos modelos preditivos. Esse tipo de comportamento acontece de forma que um grupo arbitrário de pessoas seja privilegiado em detrimento de outros e normalmente é fruto não-intencional de como os dados são coletados, selecionados e usados para treinar os modelos.

1.7. Revisão do Estado da Arte

Os objetivos que o projeto buscou atingir com a adoção da cultura MLOps são diretamente inspirados pela cultura já difundida de DevOps, onde a aplicação da metodologia em projetos de software promove o encurtamento do ciclo de desenvolvimento, aumento da velocidade de implementação e também um versionamento mais confiável de software. Em uma pesquisa realizada pelo GitLab [2] 25% das empresas na plataforma consideram estar no estado ideal de DevOps e outras 37% consideram que irão atingir esse estado entre 1 a 3 anos. No entanto, dentro do contexto de ciência dos dados e *machine learning*, a simples replicação de DevOps não garante esses benefícios em função das particularidades da área. Dessa forma, o mercado ainda está em busca de práticas e ferramentas que permitam que as empresas com projetos em *machine learning* tenham resultados equivalentes àqueles que o DevOps trouxe para o desenvolvimento tradicional de *software*.

A primeira diferença entre um projeto de desenvolvimento de software para um projeto de *machine learning* passa pelas habilidades presentes no time. Agora, além de engenheiros de software, existe também o papel do cientista de dados. Segundo um relatório da Dice [3], a demanda por cientistas de dados nos Estados Unidos em 2020 teve um aumento de 50% nos setores de saúde, telecomunicação, mídia/entretenimento e financeiro. Esses profissionais são capazes de realizar análises exploratórias de dados assim como desenvolver modelos e testá-los, mas ainda necessitam do auxílio de engenheiros de software para criarem serviços e colocarem seus modelos em produção. O ambiente de desenvolvimento precisa permitir que todos os profissionais dentro do time sejam capazes de exercer suas tarefas dentro de seus conhecimentos e habilidades e também que consigam se comunicar de forma efetiva.

Na etapa de desenvolvimento, um projeto de *machine learning* possui uma via experimental muito forte. Independente se um modelo está sendo criado ou alterado, existe a necessidade de testar diferentes dados, algoritmos e configurar diferentes parâmetros. Do ponto de vista de software, as práticas de versionamento são as mesmas aplicadas em DevOps, entretanto versionar dados, modelos e também um experimento completo que inclua todo o *pipeline* que gerou determinado resultado possui uma complexidade mais elevada.

Outra diferença de aplicação encontrada entre DevOps e MLOps está no processo de integração contínua e entrega contínua (CI/CD). Práticas como aplicação de testes unitários em códigos ainda estão presentes, mas agora o monitoramento dos dados ganha um papel relevante, afinal é a integridade desses dados que garante o treinamento e também um funcionamento adequado do modelo. Além dos testes de dados e modelos, tarefas como colocar um novo modelo em produção ou alterar as features utilizadas para predição necessitam de agilidade, principalmente por estarem no ponto de intersecção do projeto entre os profissionais de ciência dos dados e engenheiros de software.

Um projeto de *machine learning* em produção também necessita de um acompanhamento especial. Em um desenvolvimento de software tradicional garantir que seu produto esteja disponível e funcional para os clientes que utilizam pode ser suficiente. Já em um projeto de *machine learning* é necessário que seu modelo continue dentro das suas métricas definidas na construção do negócio, portanto esse acompanhamento é vital para o projeto. O conceito de *drift* relata justamente como as propriedades estatísticas utilizadas por um modelo de predição mudam ao longo do tempo por razões desconhecidas [4].

Dado o crescimento de projetos que envolvem *machine learning*, existe uma iniciativa atual de orientar esse mercado com novas ferramentas e boas práticas. Todas as necessidades

de análise de dados, controle de versão, serviços de teste e build, serviços de *deployment*, versionamento de dados e modelos, *feature store*, *metadata store* e orquestração de *pipeline* [5] podem ser supridas por uma gama de ferramentas. Iniciativas como o panorama de inteligência artificial da Linux Foundation [6] e o repositório *Awesome Production Machine Learning* no Github do Institute of Ethical AI [7] buscam mapear essas ferramentas disponíveis a fim de auxiliar na escolha apropriada para um projeto.

A implementação dessas ferramentas para construção de um projeto de *machine learning* passa pela leitura de documentações técnicas. Ferramentas completas e robustas como Amazon Sagemaker, Azure AI e Google AI possuem uma documentação unificada e extensa de todas suas soluções oferecidas com exemplos práticos e didáticos [8,9,10], além de integração com projetos *open source* [11].

A *suite* de ferramentas IBM Watson também possui uma documentação completa e extensa [12], mas como as ferramentas disponíveis não cobrem todas as necessidades de um escopo de projeto de MLOps, a combinação de ferramentas passa pela leitura de diferentes documentações descentralizadas.

Levantadas as diferenças entre uma cultura já bem estabelecida como DevOps com uma cultura recente como MLOps e também quais são as referências atuais para o tema, avalia-se que para esse projeto é necessário a realização de uma pesquisa de quais ferramentas e métodos podem complementar o *suite* IBM Watson para aplicar MLOps em empresas que queiram desenvolver projetos eficientes de *machine learning*, assim como um guia prático para auxiliar em uma possível implementação.

2. METODOLOGIA

O foco deste trabalho foi a elaboração de um guia prático de como implementar MLOps. Dessa forma, o desenvolvimento foi baseado em 3 etapas recorrentes como mostrado na Figura 1:

Figura 1 - Diagrama do Ciclo de Desenvolvimento do Projeto.



Fonte: Os autores (2021).

Esse fluxo foi utilizado por permitir:

- Entender, dentro do ciclo de desenvolvimento de um projeto de *machine learning*, quais necessidades eram expostas para a implementação de MLOps.
- Pesquisar ferramentas e métodos que fossem capazes de suprir a necessidade definida na etapa anterior.
- Implementar e entender em qual contexto de projeto ela se aplica, definindo quais personas estariam envolvidas.

A aplicação dessas etapas no desenvolvimento foi utilizada junto a metodologia Scrum, justamente por se encaixar na prática de *sprints* semanais. Às sextas-feiras foram realizadas reuniões com o professor orientador e o mentor da empresa em que era discutido avanços que o grupo havia conseguido em relação a alguma das etapas do fluxo da Figura 1. Ao fim da reunião era decidido se o grupo poderia avançar para a próxima etapa ou analisar se havia necessidade de se voltar uma etapa.

2.1. Pesquisa de Ferramentas

A primeira atividade desenvolvida neste projeto foi uma pesquisa preliminar sobre MLOps: conceitos, práticas e ferramentas. Determinou-se com essa pesquisa que a cultura MLOps abrange essencialmente as seguintes atividades e componentes:

- **Versionamento de Dados e Modelos:** Capacidade de versionar dados e modelos, referindo-se a salvar novas cópias de *datasets* e modelos baseados em novas alterações nos mesmo, garantindo melhor reprodutibilidade do sistema *machine learning*.
- **Feature Store:** Componente de *pipeline* de dados que tem o intuito de transformar dados não formatados de diferentes fontes em *datasets* de features customizadas.
- **Serviço de Deployment:** Ferramentas capazes de gerenciar *deploys* de modelos de *machine learning* em produção.
- **Integração Contínua e Entrega Contínua (CI/CD):** Combinação de práticas que visam tornar o processo de integração de código mais eficiente, utilizando *builds* e testes automatizados, e práticas que garantem entrega contínua de um sistema, com a capacidade de realizar mudanças e *upgrades* de forma rápida e automática.
- **Automatização de Pipelines:** Capacidade de automatizar séries de processos de captura e manipulação de dados e séries de processos referentes ao modelo e seu treinamento.
- **Monitoramento de Modelos:** Processo de monitoramento e análise de performance de modelos em produção em tempo real.

Em seguida foi debatido com o mentor da empresa quais necessidades de MLOps as ferramentas já existentes da IBM eram capazes de suprir e quais ainda não. A ferramenta IBM Watson se mostrou uma alternativa viável como serviço de *deployment* e monitoramento de modelos em produção. No entanto, foi exposto que alternativas para versionamento de dados, *feature store*, automatização de *pipelines* e aplicação de integração contínua (CI/CD) poderiam complementar o uso do IBM Watson a fim de criar um ecossistema mais próximo ao que se espera em MLOps. Dessa forma, a pesquisa realizada foi estruturada nas necessidades complementares ao IBM Watson, porém sem excluir ferramentas de empresas

concorrentes, uma vez que também é de interesse da IBM entender o que de disponível no mercado está sendo utilizado. Na Tabela 3 é possível observar as necessidades traçadas e quais ferramentas as solucionam:

Tabela 3 - Características das Ferramentas

	Versionamento de dados e modelos	Feature store	Automatização de <i>pipelines</i>	Integração Contínua	Deployment e Monitoramento
IBM Watson	Parcial	Não	Parcial	Parcial	Sim
Feast	Não	Sim	Não	Não	Não
DVC	Sim	Não	Sim	Não	Não
CML	Não	Não	Não	Sim	Não
Kubeflow	Sim	Não	Sim	Não	Sim
Amazon Sagemaker	Sim	Sim	Sim	Sim	Sim

Fonte: Os autores (2021).

Após separar as ferramentas capazes de suprir as necessidades traçadas com o mentor da empresa, foi realizada uma pesquisa mais detalhada sobre como cada uma das ferramentas listadas funcionam.

2.1.1. IBM Watson

A IBM oferece diversos serviços e ferramentas para o desenvolvimento de modelos preditivos e de inteligência artificial que possam melhorar processos, interações e ações. Dentre eles, o Watson ML permite que modelos sejam salvos na nuvem, onde podem ser versionados e acessados por meio de APIs, assim permitindo que predições sejam feitas sem a necessidade de se criar e configurar uma instância para hospedar o micro-serviço.

O AutoAI é uma implementação da IBM de AutoML, tecnologia que estende a automação da construção de modelos para todo o ciclo de vida de um modelo de aprendizado de máquina. Aplicando automação inteligente à tarefa de construir modelos preditivos,

preparando dados para treinamento, identificando o melhor tipo de modelo para os dados fornecidos e, em seguida, escolhendo os recursos, ou colunas de dados, que melhor suportam o problema que o modelo está resolvendo. Por fim, a ferramenta testa uma variedade de opções de ajustes e otimização de hiperparâmetros para alcançar o melhor resultado à medida que gera e classifica os pipelines candidatos.

Com as empresas dependendo cada vez mais de modelos de Inteligência Artificial para acelerar a tomada de decisões e agilizar as operações, a adoção bem-sucedida dessa tecnologia depende da confiança e da transparência em como os modelos são construídos, implantados e gerenciados. Para isso, o OpenScale possibilita o monitoramento e gerência de *fairness*, explicabilidade e *drift* do modelo, além de visualização e rastreamento de modelos em produção. Portanto, a ferramenta valida, testa e monitora modelos para assegurar sua melhor performance, além de mitigar riscos regulatórios, de reputação e operacionais.

2.1.2. Feast

As *features* são os dados de entrada para um modelo preditivo de *machine learning*. Conforme os tipos de modelos que uma empresa emprega em produção, muitos deles necessitam receber dados constantemente, como serviços de detecção de fraude. Esses serviços precisam receber dados atualizados *online*, como o local de uma nova transação de um usuário, e combiná-los com dados já armazenados em *data lakes* ou *data warehouses*, como residência deste usuário. Combinados esses dados, pode-se criar features complexas e úteis para o modelo, como uma feature indicando se o local da transação é o local de residência do usuário. As *feature stores* são componentes que facilitam a integração desses dados e a transformação deles em *features*, proporcionando automatização no processamento de dados, versionamento de *features* com linha do tempo e maior facilidade de geração de novas *features*, além de possibilitar que elas sejam facilmente compartilhadas por diferentes times, isso é especialmente importante em empresas grandes que possuem diversos times que lidam com grandes quantidades de dados.

O Feast é uma ferramenta *open-source* poderosa que cumpre esse papel. Sendo originalmente desenvolvida em parceria com o Google Cloud Platform, é atualmente mantida pela Tecton, uma das maiores empresas de *feature store* empresarial. Essa ferramenta é uma das mais robustas do mercado e com a maior comunidade, contando com um sistema de integração em Kubernetes que possa ser utilizado com IBM Kubernetes Service (IKS),

Amazon EKS e Google GKE, além da instalação normal por Kubernetes com o gerenciador de pacotes Helm. Esse sistema conta com a plataforma de *streaming* Kafka para recebimento de dados em tempo real, banco de dados Redis, que proporciona armazenamento de features online em memória de forma rápida, banco de dados em PostgreSQL para armazenamento das features offline, além dos sistemas Feast-Core e Feast-Serve, que gerenciam a aplicação e se comunicam por linha de comando ou por *SDKs* (*Software Development Kits*) em linguagens como Python, Java e Go.

Com o Feast, sistemas de predição em tempo real ficam muito mais robustos, porém uma desvantagem da ferramenta é seu alto custo de infraestrutura. Ela necessita de Kubernetes e uma quantidade razoável de espaço para o PostgreSQL e memória para o Redis, e muitos pequenos negócios, como startups, teriam que arcar com altos custos para instâncias que suportem o Feast, de forma que esse capital muitas vezes não está disponível. Por esse motivo, essa ferramenta é recomendada para empresas que realmente necessitam de dados *online*, pois existem esses altos custos atrelados a essas altas capacidades.

2.1.3. DVC e CML

A utilização de ferramentas de versionamento de código é vital na indústria de desenvolvimento de software. A importância destas ferramentas estão na possibilidade de replicar uma mesma base de código para que diversas pessoas possam trabalhar ao mesmo tempo em um projeto. Além disso, o versionamento dessas bases permite trabalhar em seções diferentes de forma organizada e sem comprometer a integridade do código em produção.

Por mais que essas ferramentas resolvem diversos problemas no desenvolvimento de software tradicional, ainda existem questões problemáticas no ambiente de *machine learning*. O versionamento de código ainda é importante, mas ao gerar um modelo novo é necessário entender quais dados foram usados para treinar, qual foi o processamento realizado nestes dados, armazenar as features e também como esse modelo foi testado.

Não é possível tratar desses problemas com ferramentas tradicionais de versionamento, como por exemplo o Git. As ferramentas de versionamento de código não são capazes de versionar grandes quantidades de dados. Foi pensando nisso que outras ferramentas começaram a surgir, com objetivo de adaptar soluções já existentes para esse atual cenário.

O Data-Version-Control (DVC) é uma ferramenta de comando de terminal

open-source que procura reproduzir a funcionalidade do Git em fluxos de *machine learning*. A usabilidade não é apenas parecida com o Git, ela é usada de forma conjunta. Os comandos são usados de maneira intercalada, de forma que o Git é responsável por guardar referências para onde os dados estarão armazenados. Já o DVC é responsável por criar referências e armazenar os dados em um repositório local ou remoto como na IBM Cloud Object Storage (COS), Amazon Web Service (AWS) S3, Azure entre outros serviços em nuvem.

Com o DVC é possível armazenar dados, modelos, *pipelines* de pré-processamento, *pipelines* de treinamento e teste em um repositório como uma única fonte de verdade. Toda a equipe pode trabalhar com dados e os códigos, realizando modificações e submetendo suas versões ao repositório principal.

Por fim, com o auxílio da ferramenta Continuous Machine Learning (CML), é possível levar todos os *pipelines* criados no DVC para um cenário de integração contínua. Ao criar arquivos do tipo *yaml* em um serviço como o Github Actions, a aplicação do CML permite executar na nuvem uma série de testes e *pipelines* para garantir que o repositório esteja sempre funcionando e dentro dos padrões definidos. Ao criar *branches* para alterar alguma etapa do projeto, além dos testes, é possível gerar um relatório automatizado para quem for revisar um *pull request*.

Com DVC e CML, um projeto é capaz de garantir versionamento de dados e *pipelines* de forma muito parecida como é feito em um ambiente de desenvolvimento de *software*, além de ser possível automatizar e garantir integração contínua dos processos a um baixo custo, já que se tratam de ferramentas *open-source*.

2.1.4. Kubeflow

O Kubeflow é uma plataforma *open-source* feita para tornar o fluxo de desenvolvimento de *machine learning* mais simples e coordenado em um cluster de Kubernetes. Este oferece ferramentas para *deploy*, escalar e gerenciar sistemas, além de ter integrado um servidor JupyterHub, que permite que vários desenvolvedores contribuam para o projeto. Na parte de gerenciamento, a plataforma permite um monitoramento detalhado de todo o projeto.

Essa plataforma é relativamente nova, sua versão 1.0 foi lançada no início de 2020 para o público, porém ganhou bastante espaço na comunidade por facilitar tanto o desenvolvimento e gerenciamento quanto o *deploy* e monitoramento de um modelo em um

ambiente de Kubernetes. Em contraponto, a configuração inicial e o custo de um cluster podem ser uma barreira de uso para muitos times.

2.1.5. Amazon Sagemaker

O Sagemaker é um serviço oferecido pela AWS que engloba praticamente todo o ciclo de MLOps, incluindo desde *feature store*, jupyter notebooks, ajustes de parâmetros automáticos, *deploy* e monitoramento. Essa plataforma não foi explorada a fundo devido ao fato do trabalho ser desenvolvido com mentoria da IBM e a sua utilização completa também torna mais difícil uma futura migração para outro provedor de nuvem, prática conhecida como *vendor lock-in*.

2.2. Análise das informações coletadas

Após se obter um conhecimento mais aprofundado de cada uma das aplicações citadas, se mostrou necessário a elaboração de contextos e personas em projetos de *machine learning* para analisar em qual cenário cada ferramenta pode contribuir mais. Atualmente, no mercado existem diversos perfis de empresas e grupos que buscam aplicar *machine learning*, de forma que existem diferentes necessidades e características. Para a elaboração de um guia aplicável, é de suma importância entender quem está procurando o guia e qual a necessidade que esta equipe passa. Além disso, as ferramentas pesquisadas também possuem pontos de sobreposição, logo o melhor entendimento do contexto de aplicação permite optar em qual objetivo cada ferramenta se encaixa melhor. Por fim, foram elaborados três contextos.

2.2.1. Cenário 1: Startups e Empresas Pequenas

Nesse primeiro cenário, foi definido que a equipe é formada por poucos engenheiros e pesquisadores, de forma que a maior parte do desenvolvimento é manual e as automações estão raramente presentes. Esse cenário é comum a startups e pequenas empresas que estão utilizando *machine learning* em seus serviços, assim como equipes de pesquisadores ou desenvolvedores amadores, como acadêmicos ou projetos *open source* pequenos. Essas equipes não necessitam de um processo complicado de automação, ao passo que não tem capital financeiro para investir em infraestruturas caras e por estarem mais focadas em provar um conceito do que ter um serviço em produção.

O mais comum a esses grupos é utilizar plataformas de versionamento de código, normalmente em padrão Git, como Github e Gitlab. Geralmente, as pesquisas são conduzidas em Jupyter Notebooks, por vezes em plataformas como Jupyter Lab ou Google Colab, que são ferramentas que trazem maior integração entre equipes para esses documentos. Os *datasets* de treinamento estão geralmente salvos na máquina, porém outras vezes podem ser adquiridos por um serviço de armazenamento de dados como IBM Cloud Object Store (COS) ou AWS S3, mantido pela empresa com baixo custo de manutenção. Esses *datasets* são raramente versionados e bem organizados. A partir das pesquisas realizadas e do treinamento ser completo, são exportados os modelos em arquivos (com hdf5 ou pickle, por exemplo), que são manualmente utilizados para *deploy* em aplicações. Nesse cenário, desenvolvedores e pesquisadores têm funções mistas, onde muitas vezes pesquisadores terão de arcar com funções de desenvolvimento e infraestrutura, tornando esse meio de desenvolvimento pouco escalável, possível apenas para projetos de pequena escala.

A metodologia de MLOps consegue trazer algumas melhorias para essas equipes, através de ferramentas que melhoram o fluxo de trabalho e preparam o projeto para se tornar mais escalável. Nesse cenário em específico, pode ser interessante para a equipe utilizar ferramentas de versionamento de dados, como o DVC, que possibilitam o versionamento dos *datasets*. Além de tornar mais fácil a inserção de um engenheiro de dados no time, por conta da maior facilidade e organização dos dados, o DVC traz ferramentas de otimização de espaço, que podem diminuir custos de uma IBM COS ou AWS S3. Além disso, o time pode optar por introduzir ferramentas de fácil acesso, como CML para criar um *pipeline* de integração (CI/CD) de *machine learning* de forma simples.

2.2.2. Cenário 2: Empresa Média ou Time pequeno de Grande Corporação

O cenário 2 pode ser considerado uma evolução do cenário 1. Este projeto exige uma equipe maior para o desenvolvimento, e por isso é necessário que os integrantes apliquem boas práticas de MLOps, resultando em um desenvolvimento do projeto mais eficiente e robusto. Aqui se encontram *startups* que já possuem um MVP e precisam evoluir seu desenvolvimento para um produto real, empresas médias que estão desenvolvendo um produto que utiliza *machine learning* e também projetos menores de grandes corporações.

Essas equipes possuem uma disponibilidade maior de capital para investimento em infraestrutura comparado ao cenário anterior.

Para o gerenciamento de dados, a utilização do DVC ainda se justifica pela fácil utilização e capacidade de versionamento de *datasets*. No entanto, pelo tamanho da equipe se justifica a necessidade de criar um ambiente de fácil reprodutibilidade de experimentos e aumentar a automatização dos *pipelines*. Dessa forma, além do DVC, a aplicação do CML é de suma importância.

A partir do momento que a equipe possui diversos profissionais experimentando novas features e novos modelos a todo momento, manter uma organização baseada em Jupyter Notebooks deixa de ser vantajoso e pode dificultar a reprodução de experimentos por outros integrantes. Dessa forma, com o DVC e o CML, o profissional ao gerar um experimento pode salvar todo o *pipeline* utilizado para que outro integrante possa reproduzir da mesma forma.

Além da reprodutibilidade, com a utilização do CML junto à ferramenta Github Actions ou GitLab CI, os profissionais também podem criar e automatizar testes para os dados, modelos e o código. Com a utilização destas ferramentas combinadas é possível poupar tempo e garantir a integridade do projeto.

Após realizar toda a etapa de experimentação do projeto, a equipe precisa colocar o modelo desenvolvido em produção. Nesta etapa, utiliza-se uma integração entre *releases* no Github com o deployment no ambiente do IBM Watson. Se um experimento se mostra suficiente para ser colocado em produção, o integrante da equipe pode fazer um *pull request* com relatório automatizado. Se aprovado, ao gerar um novo *release* no projeto este modelo é exportado para o ambiente de desenvolvimento do IBM Watson e colocado em produção, podendo ser utilizado através de uma API Rest provida pelo sistema.

Por fim, é importante mencionar a importância de acompanhar esse deployment com uma ferramenta como o IBM OpenScale. Essa ferramenta permite acompanhar métricas definidas nas regras de negócio do projeto, assim como a performance do modelo ao longo do tempo. Com isso o time pode garantir a funcionalidade do produto e também analisar a performance a fim de produzir melhorias futuramente.

2.2.3. Cenário 3: Projetos de Grande Porte

O Cenário 3 é composto por empresas de grande porte com fortes equipes de desenvolvedores e pesquisadores. Nessas equipes é esperado que sejam tomados os papéis de engenheiros e cientistas de dados, engenheiros de *machine learning*, especialistas em infraestrutura e outras funções que forem necessárias para o projeto. Aqui se enquadram empresas como bancos, grandes varejistas, redes de supermercado, empresas de cartão de crédito e quaisquer outras empresas grandes que queiram utilizar *machine learning* de forma efetiva para aumentar seus lucros. Essas são equipes que têm apoio financeiro para desenvolver uma *pipeline* robusta e utilizar infraestrutura cara, com o intuito de comportar uma grande quantidade de membros da equipe trabalhando simultaneamente com organização. Para esse cenário o MLOps passa a se tornar obrigatório para conseguir organizar todo o projeto de forma confiável.

Os pesquisadores e cientistas de dados utilizam de dados sensíveis, que devem ficar armazenados em *data lakes* e *data warehouses* consistentes, como a IBM COS, de forma segura. Muitas empresas também podem optar por usar sistemas de dados *in-house* por conta da sensibilidade de dados, utilizando a prática de cloud híbrida, quando se utiliza nuvem pública e privada ao mesmo tempo, para garantir maior segurança e flexibilidade para a empresa. Além disso, esses sistemas geralmente possuem dados vindos constantemente dos usuários, como compras em um sistema de varejo ou transações de um banco, sendo que esses dados são utilizados pelo modelo de *machine learning* em produção em tempo real. Para isso, é interessante que seja usado uma *Feature Store*, tal que consigam organizar os dados de forma efetiva e transformem em *features* para o treinamento dos modelos e para os modelos em produção. O Feast utilizando a arquitetura de IBM Kubernetes Service (IKS) é a solução ideal para esse cenário, sendo um sistema complexo e eficiente, que consegue suportar alto fluxo de dados online com Kafka e prover as *features* para o modelo de forma rápida com um banco de dados Redis. Dessa forma, o Feast se conecta com o Kubeflow, que permite a criação e treinamento de *pipelines* complexas de *machine learning*, tudo de forma integrada com Kubernetes. O *pipeline* do Kubeflow então cria deploys de modelo no IBM Watson, com o sistema de monitoramento OpenScale integrado. Dessa forma, uma empresa de grande porte consegue ter um fluxo de desenvolvimento sólido e capaz de atender as demandas de produto.

Essa arquitetura é cara e de difícil instalação, porém robusta e de fácil gerenciamento após implementada, então é necessário um alto capital para conseguir criar esse sistema.

Kubernetes por padrão é uma tecnologia custosa, e executar ferramentas como o Feast e o Kubeflow necessitam de instâncias poderosas. Apesar disso, essas empresas conseguem arcar com esses custos, de forma que criam um ecossistema de *Machine Learning* que cumpre o MLOps de forma exemplar, conseguindo trazer mais agilidade no desenvolvimento, controle da aplicação, organização da equipe como um todo e rapidez em modificar o modelo em tempo real.

Todos os cenários descritos buscam aplicar a cultura MLOps de forma a otimizar cada projeto em sua própria dimensão. Na Tabela 4 abaixo pode-se analisar um resumo dos cenários e características:

Tabela 4 - Cenários

	Cenário 1	Cenário 2	Cenário 3
Quem são?	<ul style="list-style-type: none"> ● Startups construindo MVP ● Pequenas empresas ● Projetos de pesquisa pequenos 	<ul style="list-style-type: none"> ● Startups construindo produto real ● Médias empresas ● Projetos pequenos em grandes corporações 	<ul style="list-style-type: none"> ● Projetos de larga escala ● Grandes Empresas ● Bancos, grandes varejistas, redes de supermercado e outros
Características	<ul style="list-style-type: none"> ● Pouco investimento em infraestrutura ● Equipe pequena ● Foco em provar conceito 	<ul style="list-style-type: none"> ● Bom investimento em infraestrutura ● Equipe média ● Precisa garantir reprodutibilidade e automação ● Precisa realizar deployment de modelos para produção 	<ul style="list-style-type: none"> ● Alto investimento com infraestrutura, com possível cloud híbrida ● Equipe grande que abrange diversos cargos ● Precisa garantir reprodutibilidade, automação e deploy de forma rápida ● Dados sensíveis em alta quantidade ● Fluxo constante de dados

Fonte: Os autores (2021).

A partir dos cenários definidos, foram escolhidas algumas ferramentas que podem ajudar as equipes, de forma a preencher os mais importantes pontos de MLOps para os casos específicos de cada cenário, como visto na Tabela 5.

Tabela 5 - Relação Cenário com Ferramentas Sugeridas

	Cenário 1	Cenário 2	Cenário 3
Versionamento de dados e modelos	DVC com IBM COS	DVC com IBM COS e IBM Watson ML	IBM Watson Studio e IBM Watson ML
Feature store	Não presente	Não presente	Feast com IBM Kubernetes Service
Automatização de pipelines	DVC Pipelines (Opcional)	DVC Pipelines	Kubeflow
Integração Contínua	Não presente	Github Actions + CML	Kubeflow
Deployment	Deploy manual	IBM Watson ML	IBM Watson ML
Monitoramento	Sem Monitoramento	IBM Watson OpenScale	IBM OpenScale

Fonte: Os autores (2021).

Através da análise das ferramentas e cenários, foi possível tirar algumas conclusões. As empresas que possuem projetos de larga escala já possuem profissionais especializados em criar uma infraestrutura organizada e automatizada para o projeto. Muitas dessas empresas acabam até desenvolvendo sua própria ferramenta adaptada para o seu projeto. Dito isso, os cenários 1 e 2 possuem uma maior indefinição ainda em como executar o projeto utilizando MLOps. O conjunto de ferramentas abordado nesses cenários combina novas tecnologias open-source como o DVC e CML, com tecnologias já conhecidas no mercado com o IBM Watson. Essa combinação permite que projetos de *machine learning* possam aplicar a cultura MLOps de forma completa sem precisar de uma grande infraestrutura e investimentos que só uma grande corporação conseguiria executar.

Por isso se entende que o ferramental dos cenários 1 e 2 foi mais vantajoso para a criação de um guia de MLOps para clientes da IBM. Com ele é possível aplicar em pequenos e médios projetos as melhores práticas que envolvem MLOps.

2.3. Proposta Inicial de Guia

Com o intuito de disponibilizar esses ensinamentos e montar um guia que possa ser útil a diversas equipes, além de um exemplo *end-to-end* que utilize as boas práticas de MLOps com as ferramentas sugeridas pelo grupo, foram pesquisados os seguintes modelos de ferramenta de aprendizado:

- **Livro Guia:** Um livro físico ou digital(*ebook*) que compreende o guia e detalha o exemplo *end-to-end*.
- **Curso em Vídeo:** Série de vídeos que relatam o uso das ferramentas e a montagem do exemplo *end-to-end*, disponível em uma plataforma como YouTube, de forma aberta.
- **Documentação em Website:** Site aberto que introduz a teoria e relata o uso das ferramentas de forma modular, dividida em diferentes páginas, além da construção do exemplo *end-to-end*. Código-fonte aberto no Github e site em um domínio customizado.

A partir desses modelos, o grupo recolheu dados que permitiram a escolha de uma ferramenta que possa ser consultada de forma fácil, modular, sendo possível usufruir do guia de forma não-linear, e pelo maior número de pessoas de forma gratuita.

A comparação de características recolhidas para cada modelo são exibidas na Tabela 6 abaixo.

Tabela 6 - Formatos do Guia

Modelo	Permite Modificação	Suporta mídias adjacentes (vídeos, códigos, links, etc.)	Fácil consulta a diferentes partes do guia	Fácil de ser achado por motores de busca (Google)
Livro Guia	Sim, mas necessita reedição	Não, apenas imagens e links apenas em versão <i>ebook</i>	Sim, mas apenas em versão <i>ebook</i>	Não, não é indexado no Google de forma fácil
Curso em Vídeo	Sim, mas o vídeo é apagado, perdendo visualizações do guia	Não, apenas imagens e vídeos	Não, necessita procurar dentro de cada vídeo por conteúdos muito específicos	Sim, se estiver no YouTube ou semelhante
Documentação em <i>Website</i>	Sim, facilmente alterando o site	Sim, suporta diversos tipos de mídia	Sim, permite busca no <i>Website</i>	Sim, indexado por motores de busca

Fonte: Os autores (2021)

A partir dos dados coletados, foi escolhido o modelo de Documentação em *Website* por sua flexibilidade para ser modificado, seu suporte para diferentes mídias, como vídeos e links, permitir busca dentro do guia, e ser possível de ser indexado pelo Google, podendo atrair mais interessados.

A ferramenta escolhida para montar este *Website* foi o Material for MkDocs, uma plataforma popular *open-source* para criação de documentações baseado em MkDocs com tema *Material*, usado por diversas empresas como Google, AWS e Microsoft e ferramentas *open-source* como FastAPI, Kubernetes e Pydantic. Além disso, foi escolhido o Github para armazenar o projeto *end-to-end*, sendo a plataforma Git mais popular atualmente.

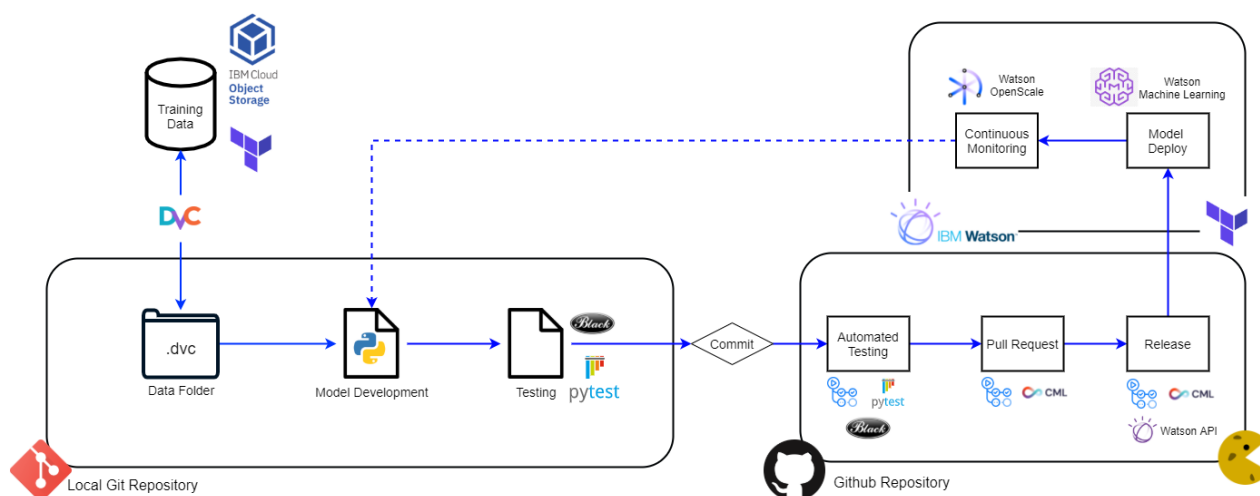
3. RESULTADOS

3.1. Exemplo *end-to-end*

O exemplo *end-to-end* foi desenvolvido visando demonstrar as aplicabilidades de MLOps em um projeto de escala referente ao Cenário 2, definido como projetos de Empresas Médias ou Time pequeno de Grande Corporação na seção 2.3.2. Dessa maneira, foram utilizadas as ferramentas recomendadas para esse cenário, sendo elas as seguintes:

- **IBM COS:** *Data warehouse* para os *datasets* de treinamento.
- **DVC:** Versionamento dos *datasets* guardados na *data warehouse* e permitir fácil acesso para desenvolvedores.
- **CML:** Criação de *pipelines* de *machine learning* e visualização de dados (*DataViz*) de desempenho de modelos.
- **Github:** Gerenciamento de versão de código, além de prover ferramentas como Github Actions e Github Pages.
- **Github Actions:** Executar testes de integração a cada *push* no repositório, para executar o CML e para integrar *releases* com o IBM Watson ML para *deploy*.
- **IBM Watson ML:** Armazenamento e *deploy* de modelos de *machine learning*.
- **IBM Openscale:** Monitorar e avaliar os modelos do IBM Watson em tempo-real.

Figura 2 - Arquitetura da solução.



Fonte: Os autores (2021).

Foi utilizado o *dataset Rain in Australia: Predict next-day rain in Australia* [13], contendo condições meteorológicas na Austrália para criação de um modelo preditivo de chance de chuva.

A estrutura de pastas e arquivos base do projeto é criada a partir de um *template* Cookiecutter, criado pelo grupo. Os arquivos base criados servem para as configurações iniciais do projeto, como o arquivo de requerimentos de pacotes Python, o arquivo de *metadata*, contendo informações como nome do projeto, autor e versão, o arquivo de documentação mostrando como inicializar o projeto, arquivos de configuração de Github Actions que serão comentados posteriormente e um arquivo exemplo de testes em Pytest.

O Terraform foi utilizado como *script* de inicialização da infraestrutura, prática conhecida como *Infrastructure-As-Code*, devido a facilidade de desenvolvimento e compreensão, além de suas funcionalidades que facilitam a migração e modificações da infraestrutura. Esse *script* é responsável por iniciar os dois recursos principais para o projeto na IBM Cloud, o Watson Machine Learning (WML) e o IBM COS.

O projeto conta com uma *test suite* em Python utilizando Pytest, que compreende testes unitários com o intuito de conferir o bom funcionamento das funções do código. Para complementar, é utilizado a ferramenta Pre-Commit, que configura Git Hooks, *scripts* customizados que podem ser acionados por ações Git como *push* e *commit*, de forma, que toda vez que o desenvolvedor for realizar *commit*, seja executado um *check* de Pytest e um do *linter* Black de código Python, de forma que se os arquivos não passarem nos testes ou estiverem mal formatados o *commit* não será executado. Essa técnica diminui a quantidade de código no repositório que carrega erros ou imprecisões.

Em seguida é feito o pré-processamento dos dados, ignorando algumas *features* que foram julgadas desnecessárias, excluindo dados vazios e fazendo *One-Hot-Encoding* de dados categóricos. O *dataset*, tanto antes quanto depois de ser pré-processado, é adicionado ao repositório do DVC do projeto.

O modelo escolhido foi um classificador *Random Forest* com um *Standard Scaler*. Tanto o código referente ao treinamento do modelo quanto para a validação dele a partir de métricas como *accuracy*, matriz de confusão e curva ROC foram declarados em funções de forma que possam ser acessados facilmente por outros *scripts*. Isso é importante pois, a partir dessa estrutura, é possível treinar, avaliar e fazer deploy dos modelos com *scripts* padronizados, garantindo a reprodutibilidade.

Foi criado um *pipeline* do DVC que detalha cada etapa do experimento, desde o pré-processamento, passando pelo treinamento e finalmente avaliação, cada etapa tem suas

dependências e saídas, dessa forma, ao executar a *pipeline*, as etapas só serão rodadas se alguma mudança for feita que altere a dependência delas. A *pipeline* é ilustrada na Figura 3.

Figura 3 - Pipeline do Experimento do DVC

```

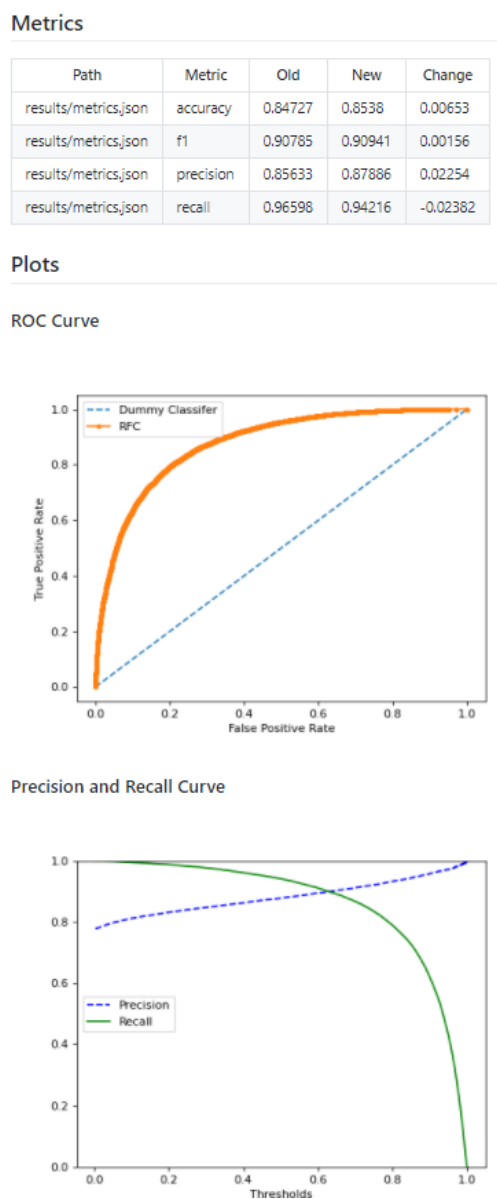
stages:
  std_check:
    cmd: src/scripts/Scripts/std_check.sh ./
  preprocess:
    cmd: python3 ./src/preprocess_data.py ./data/weatherAUS.csv
    deps:
      - ./src/preprocess_data.py
      - data/weatherAUS.csv
    outs:
      - ./data/weatherAUS_processed.csv
      - ./data/features.csv
  train:
    cmd: python3 ./src/train.py ./data/weatherAUS_processed.csv ./src/model.py 200
    deps:
      - ./data/weatherAUS_processed.csv
      - ./src/model.py
      - ./src/train.py
    outs:
      - ./models/model.joblib
  evaluate:
    cmd: python3 ./src/evaluate.py ./data/weatherAUS_processed.csv ./src/model.py ./models/model.joblib
    deps:
      - ./data/weatherAUS_processed.csv
      - ./models/model.joblib
      - ./src/evaluate.py
      - ./src/model.py
    outs:
      - ./results/precision_recall_curve.png
      - ./results/roc_curve.png
    metrics:
      - ./results/metrics.json:
          cache: false

```

Fonte: Os autores (2021).

Além de usar o Git para versionar os códigos, é usado o DVC para versionar os dados, modelos e métricas. Portanto, usando o CML e o GitHub Actions, a cada *commit* feito ao repositório, são executados novamente os testes de Pytest e Black, gerando redundância nos testes e evitando problemas reprodutibilidade, garantindo que não estão ocorrendo erros na execução do *pipeline* para que então seja gerado um relatório, exemplificado pela Figura 4, comparando a nova versão com a anterior. Esta prática acaba por facilitar o entendimento das melhorias feitas a cada iteração do modelo.

Figura 4 - Relatório gerado pelo CML para uma nova versão do modelo



Fonte: Os autores (2021).

Quando o modelo estiver maduro o suficiente, é interessante colocá-lo em produção ou atualizar o modelo que já está, dessa forma toda vez que um *release* é feito no Git, o CML executa um *script* que realiza o *deploy* ou atualização do modelo Watson ML referente aos IDs do arquivo de *metadata*.

Com modelo em produção, é possível acessar as suas predições a partir de requisições ou da API Python enviando as *features* e recebendo a predição e probabilidade para o valor predito na resposta. Também é possível registrar essas transações usando o OpenScale, isso é importante pois pode-se extrair informações interessantes a partir desses dados, como a

probabilidade geral das predições ou as características dos dados dos usuários. A partir do registro das transações também é possível acessar a explicação da decisão do modelo, para que seja possível entender quais *features* tiveram mais peso na decisão e quais valores teriam que ser diferentes para que a predição fosse diferente. O OpenScale está sendo configurado para o monitoramento de certos parâmetros como *drift* e *fairness*, também podendo emitir alarmes caso algum parâmetro atinja uma certa medida.

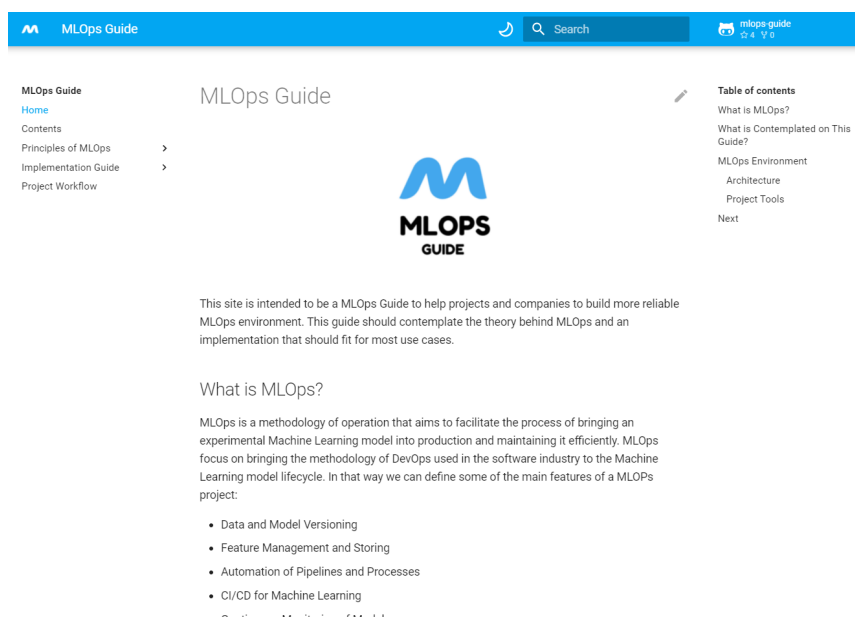
3.2. Construção do Guia

Dada a escolha de plataforma em 2.3. Proposta Inicial de Guia, foi desenvolvido o guia em site utilizando MkDocs em um repositório aberto no Github [14]. Dessa forma, foi construída uma plataforma de *deploy* automatizado utilizando Github Actions e Github Pages, onde a cada *push* na *branch* principal do repositório se tem uma versão nova no domínio.

Devido a abrangência global da IBM, foi decidido que o guia seria escrito na língua inglesa para que o público não fosse restringido somente a pessoas lusófonas e, levando em conta que o material disponível na *internet* relacionado a *machine learning* é majoritariamente em inglês, grande parte dos profissionais que pertencem ao público alvo, mesmo no Brasil, têm conhecimento dessa língua.

O guia é primariamente composto pela Página Inicial, que tem o objetivo de introduzir de forma branda o conceito de *MLOps*, demonstrar o funcionamento do ambiente pronto por meio de um vídeo, um diagrama da arquitetura e uma descrição. A partir da Página Inicial, visualizada na Figura 5, o usuário pode escolher entre seguir para as duas seções principais: Introdução aos Princípios de *MLOps* e Guia de Implementação, com subseções para cada tópico dentro dessas etapas. O guia recomenda que o usuário siga pela parte teórica dos princípios caso não esteja familiarizado com o tema.

Figura 5 - Página inicial do guia



Fonte: Os autores (2021).

3.2.1. Introdução aos Princípios de MLOps

Dentre as duas principais seções do site, a Introdução aos Princípios de MLOps trata dos conceitos teóricos de MLOps, de forma a introduzi-los para o leitor e explicar os benefícios de se empregar as técnicas definidas. Essa seção se divide em cinco subseções:

- **Versioning:** Explica os conceitos de versionamento de dados, modelos e features, além do princípio de reprodutibilidade.
- **Automation:** Concentra os benefícios de se utilizar de automações no projeto, além de explicar quais os principais estágio de um fluxo de *machine learning* que podem ser automatizados.
- **Feature Storing:** Explica o componente conhecido como *Feature Store*, seu funcionamento e vantagens no projeto. Define os conceitos de *offline* e *online store*.
- **CI/CD for Machine Learning:** Explica os conceitos de integração e entrega contínua, junto com como utilizá-los no ambiente de *machine learning*.
- **Continuous Monitoring:** Mostra os diferentes problemas que podem ocorrer com um sistema de *machine learning* em produção e explica conceitos chave de seu monitoramento.

Em cada uma das subseções, também são listadas algumas das principais ferramentas do mercado que são utilizadas para cumprir esse conceito em um ambiente de *MLOps*. Um exemplo de página pode ser visto na Figura 6, mostrando que a página concentra texto, imagens e um menu de navegação por tabela de conteúdos.

Figura 6 - Página referente *Feature Storing*, dentro da seção dos Princípios de *MLOps*

The screenshot shows the 'MLOps Guide' website. The main content area is titled 'Feature Storing' and contains a sub-section 'What is a Feature Store?'. The text explains that Feature Stores are components of data architecture that process data from various sources into features for model training and serving. A diagram below the text shows 'Streaming (new data)', 'Data Lake / Warehouse', and 'Other Misc Data' feeding into a central 'Feature Store' box. This box then outputs to 'Model Training' (with a 'Data Scientist' icon) and 'Model Serving' (with an 'Application in Production' icon). The page also features a 'Table of contents' on the right and a navigation menu on the left.

Fonte: Os autores (2021).

3.2.2. Guia de Implementação

Essa seção do guia concentra o projeto de forma mais prática e menos teórica, diferente da seção anterior. No Guia de Implementação, o site explica para um usuário como criar um ambiente MLOps do zero, utilizando do estilo tutorial e do exemplo end-to-end. Ao final dessa seção é também ensinado a utilização do fluxo criado por esse ambiente de projeto. As subseções deste guia são:

- **Introduction:** Uma introdução técnica ao projeto, contando com diagramas, lista de ferramentas e estruturação das pastas. Nessa etapa são dados os primeiros passos do projeto utilizando o Cookiecutter para criação da estrutura.
- **Environment:** Nessa etapa é mostrado como criar e utilizar o ambiente IBM Watson, utilizando Terraform e a *API* do Watson para criação da arquitetura e *setup* inicial do ambiente em nuvem.
- **Versioning:** Etapa que explica como utilizar o DVC e o IBM COS para criação de um *bucket* para versionamento de dados e modelos. Além disso, define como criar as *pipelines* de automação do projeto utilizando DVC Pipelines.
- **Deployment:** Explicação das vantagens do IBM Watson e como utilizar a *API* do Watson ML para fazer *deployment* do modelo através de Python. Além disso, mostra funções de atualização e alteração do modelo no Watson ML.
- **CI/CD:** Nesta etapa é mostrado como criar testes com Pytest e Black, além de ser criado um fluxo de testes com redundância, de forma local utilizando Pre-Commit e de forma em nuvem utilizando Github Actions. A partir das Actions é feito também a criação de relatórios customizáveis pelo CML no Github e o *deployment* automático no Watson ML a cada *release* do repositório, criando assim um *pipeline* completo de integração contínua.
- **Monitoring:** Essa etapa concentra um guia de utilização do IBM OpenScale para monitoramento contínuo, a partir da interface gráfica do site, assim como a sua utilização a partir da *API* do Watson em Python.
- **Project Workflow:** Demonstração do funcionamento do projeto pronto, partindo da clonagem do repositório configurado, alterações no modelo, *commit* e *push* no repositório, *pull request*, *deployment* feito a partir do release no repositório e uso e monitoramento do modelo no IBM Watson. Essa etapa pode ser utilizada por equipes para instruir novos participantes ou membros que não participaram do processo de criação do ambiente.

As subseções deste guia podem conter mais de uma página, conforme necessidade da etapa, como é mostrado na Figura 7.

Figura 7 - Página referente as *pipelines* do DVC, dentro da seção de Implementação

The screenshot shows the 'Working with Pipelines' page on the MLOps Guide website. The page has a blue header with the 'MLOps Guide' logo, a search bar, and a GitHub repository link. A left sidebar contains a navigation menu with categories like 'MLOps Guide', 'Contents', 'Principles of MLOps', 'Implementation Guide', 'Introduction', 'Environment', 'Versioning', 'What is DVC?', 'Data Versioning', 'Working with Pipelines', 'Deployment with Watson Machine Learning', 'CI/CD for Machine Learning', 'Monitoring with IBM OpenScale', and 'Project Workflow'. The main content area is titled 'Working with Pipelines' and includes a table of contents on the right. The text explains that after learning about versioning, it's time to build an experiment pipeline. It mentions using a dataset for training a classification model and lists three stages: preprocessing data, training the model, and evaluating the model. A warning box states that the page uses a template repository for explanations and encourages users to create their own pipelines. The 'Creating pipelines' section describes the three components of a pipeline: inputs, outputs, and command, with an example for the preprocessing stage.

MLOps Guide Search mlops-guide

MLOps Guide

Home

Contents

Principles of MLOps >

Implementation Guide >

Introduction >

Environment >

Versioning >

What is DVC?

Data Versioning

Working with Pipelines

Deployment with Watson Machine Learning

CI/CD for Machine Learning >

Monitoring with IBM OpenScale

Project Workflow

Working with Pipelines

After learning how to version data or models using DVC it's time to build your experiment pipeline.

Let's assume that we are using the last section dataset as a data source for training a classification model. Let's also consider that we have three stages in this experiment:

- Preprocessing your data(extract features...)
- Train the model
- Evaluate the model

Here you should have in hands our scripts: [preprocess.py](#), [train.py](#), [evaluate.py](#) and [model.py](#).

Warning

Just as in the last section, we will use the help of the [template repository](#) to explain and build DVC's pipelines. Feel free to use your scripts and create specific pipelines for your project needs.

Creating pipelines

DVC builds a pipeline based on three components: Inputs, Outputs, and Command. So for the preprocessing stage, this would look like this:

- Inputs: weatherAUS.csv.csv and preprocess.py script
- Outputs: weatherAUS_processed.csv
- Command: python preprocess.py weatherAUS.csv

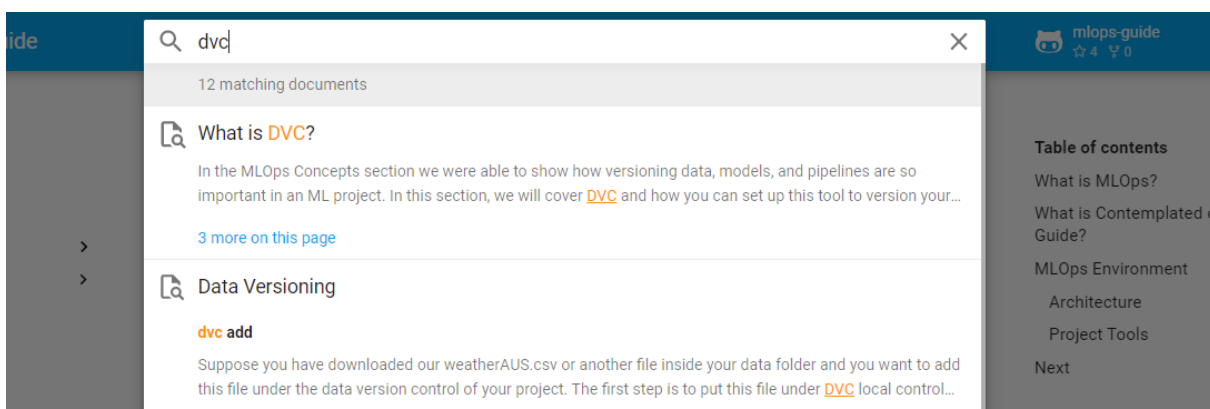
So to create this stage of preprocessing, we use `dvc run`:

Fonte: Os autores (2021).

3.2.3. Ferramental Adicional do Site

Além da página inicial e das duas principais seções, o site também conta com algumas funções e páginas adicionais para facilitar a leitura e a utilização. A página *Contents* serve como um glossário para o site, que complementa e facilita a navegação do site, uma vez que apresenta links para todas as páginas e uma breve descrição sobre o assunto que estas abordam. Além disso, o site também apresenta uma ferramenta de busca, que permite ao usuário procurar palavras chave em meio a todos os textos das páginas, de forma que links serão exibidos de forma inteligente por ordem de relevância. Um exemplo da ferramenta de busca é exibido na Figura 8.

Figura 8 - Ferramenta de Busca



Fonte: Os autores (2021).

3.3. Validação do Guia

Para validar o guia desenvolvido, foi feito um formulário de avaliação usando a ferramenta Google Forms. Apesar de não ter sido possível encontrar pessoas para realizar todos os passos do guia devido a extensão do material, cinco voluntários leram atentamente todas as páginas do *website* e responderam às seguintes perguntas:

- De maneira geral, quão claro você sentiu que foi o guia?
- Como você sente que a parte teórica lidou com os conteúdos complexos?
- Como você sente que a parte de implementação lidou com o tutorial?
- Possui alguma crítica ou sugestão?

O retorno foi positivo, todos os voluntários responderam “Muito claro” para a primeira pergunta e “Todos os conceitos estão bem explicados” para a segunda. Na terceira pergunta, todos os voluntários responderam que as ferramentas estavam bem explicadas, porém três deles indicaram que o tutorial está completo enquanto dois disseram que o tutorial poderia estar melhor.

As principais sugestões recebidas foram relacionadas a correção de erros gramáticos no texto e formatações inconsistentes em algumas partes. Outro retorno importante foi em relação aos tamanhos de algumas imagens, que dificultavam seu entendimento. Essas críticas e outras que foram consideradas cabíveis foram solucionadas.

4. CONSIDERAÇÕES FINAIS

A elaboração desse projeto buscou introduzir os principais conceitos da cultura MLOps e desenvolver um guia com aplicação prática para profissionais envolvidos no setor de *machine learning*, assim como capacitar atuais ou futuros clientes da IBM a aplicarem essas práticas em projetos que utilizam as ferramentas da empresa. A falta de guias que buscam expor a parte prática de MLOps além da teoria e a dificuldade de escolha de ferramentas adequadas devida a extensa disponibilidade de opções no mercado são as principais questões que o projeto propôs resolver.

O objetivo geral do projeto foi atingido com o desenvolvimento do guia de implementação de MLOps. O guia contempla uma seção teórica capaz de introduzir os principais conceitos como versionamento, automação, *feature store*, integração contínua e monitoramento. Também é contemplado um tutorial de como aplicar na prática esses conceitos utilizando uma *stack* de ferramentas selecionadas para o cenário definido no projeto. Este tutorial é sustentado por um exemplo *end-to-end* desenvolvido junto ao guia que acaba por auxiliar na compreensão de cada etapa, permitindo que o leitor siga cada passo aplicando a orientação no seu próprio projeto ou siga para uma implementação já validada do exemplo *end-to-end*. No final, o guia apresenta uma seção que instrui como trabalhar no projeto recém desenvolvido e aplicar as boas práticas de MLOps.

Além do objetivo geral, o projeto também foi capaz de trazer o contexto de MLOps para os clientes que utilizam o *suite* do IBM Watson em seus projetos de *machine learning*. O exemplo *end-to-end* se utiliza do IBM COS como ferramenta de armazenamento, o IBM Watson como ferramenta para colocar os modelos em produção e o IBM Openscale para monitorar os modelos. Desta maneira, o cliente da IBM que utilizar esta *suite* terá no guia desenvolvido uma fonte de ferramentas complementares e boas práticas para o seu projeto.

A metodologia do projeto envolveu pesquisar as principais ferramentas presentes no contexto de *machine learning* e também traçar os diferentes cenários encontrados no mercado. Como parte das ferramentas utilizadas eram fornecidas pela empresa parceira do projeto, a pesquisa analisou ferramentas que pudessem agregar a *suite* do IBM Watson baseado nos conceitos chaves de MLOps. Com a *stack* definida, foi realizada uma coleta de dados com objetivo de traçar o formato no qual o guia seria disponibilizado.

Por fim, o projeto produziu uma documentação extensa que engloba a teoria de MLOps e também a implementação dessa teoria em prática. O guia completo está disponível

no formato *website* e também é aberto para receber contribuições.

É importante ressaltar que por limitações relacionadas ao tempo de desenvolvimento, o projeto não conseguiu validar a utilização do guia de uma maneira robusta. Realizar uma validação com um número maior de profissionais do setor poderia contribuir com melhores *feedbacks* e complementar seções em que o guia pode não ter cumprido seu objetivo didático.

Para trabalhos futuros, a criação de variações no tutorial para os outros cenários traçados tem potencial de enriquecer ainda mais o conteúdo do guia. A utilização de uma *feature store* por exemplo, foi abordada na seção teórica mas como não se aplicava no cenário trabalho acabou não sendo incluída na seção de implementação prática.

REFERÊNCIAS

- [1] BCC Publishing, **Machine Learning: Global Markets to 2022**. Disponível em <<https://www.bccresearch.com/market-research/information-technology/machine-learning-global-markets.html>> Último acesso em 13/04/2021.
- [2] Gitlab, **Mapping the DevSecOps Landscape**. Disponível em <<https://about.gitlab.com/resources/downloads/2020-devsecops-report.pdf>> Último acesso em 13/04/2021.
- [3] Dice, **Dice Q2 Tech Job Report**. Disponível em <<https://techhub.dice.com/Dice-Q2-Tech-Job-Report.html>> Último acesso em 13/04/2021.
- [4] Widmer, G., Kubat, M. **Learning in the Presence of Concept Drift and Hidden Contexts**. *Machine Learning* **23**, 69–101, 1996. Disponível em <<https://doi.org/10.1023/A:1018046501280>> Último acesso em 13/04/2021.
- [5] MLOps.org. **MLOps Infrastructure Stack**. Disponível em <<https://ml-ops.org/content/state-of-mlops>> Último acesso em 14/04/2021.
- [6] Linux Foundation. **LF AI & Data Foundation Interactive Landscape**. Disponível em <<https://landscape.lfai.foundation/>> Último acesso em 14/04/2021.
- [7] Ethical AI. **Awesome Production Machine Learning**. Disponível em <<https://github.com/EthicalML/awesome-production-machine-learning>> Último acesso em 14/04/2021.
- [8] Amazon Sagemaker. Disponível em <<https://docs.aws.amazon.com/sagemaker/>> Último acesso em 14/04/2021.
- [9] Azure Machine Learning. Disponível em <<https://docs.microsoft.com/pt-br/azure/machine-learning/>> Último acesso em 14/04/2021.

[10] **Google AI.** Disponível em <<https://cloud.google.com/ai-platform/docs?hl=pt-br>> Último acesso em 14/04/2021.

[11] Red Hat, **O que é open source?.** Disponível em <<https://www.redhat.com/pt-br/topics/open-source/what-is-open-source>> Último acesso em 14/04/2021.

[12] **Watson Machine Learning.** Disponível em <<https://www.ibm.com/demos/collection/Watson-Machine-Learning/>> Último acesso em 14/04/2021.

[13] YOUNG, J. **Rain in Australia.** Disponível em <<https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>> Último acesso em 13/04/2021.

[14] **Guia MLOps,** Disponível em: <https://mlops-guide.github.io> Último acesso em 08/06/2021.

[15] Google, **MLOps Continuous Delivery and Automation Pipelines.** Disponível em <<https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning?hl=pt-br>> Último acesso em 13/04/2021.

[16] **IBM AutoAI.** Disponível em <<https://www.ibm.com/cloud/watson-studio/autoai>> Último acesso em 13/04/2021.

[17] **Openscale.** Disponível em <<https://www.ibm.com/cloud/watson-openscale>> Último acesso em 13/04/2021.

[18] **DVC.** Disponível em <<https://dvc.org/doc>> Último acesso em 13/04/2021.

[19] **Feast.** Disponível em <<https://feast.dev>> Último acesso em 13/04/2021.

[20] **Kubeflow.** Disponível em <<https://www.kubeflow.org>> Último acesso em 13/04/2021.

Apêndice A – Estrutura do Guia

- **Home:** Introdução ao guia e seu formato bem como um vídeo narrado de demonstração do fluxo completo do pipeline desenvolvido e uma tabela com as ferramentas utilizadas e suas respectivas licenças de uso.
- **MLOps Theory:** Uma breve introdução aos principais Princípios de MLOps: *Data and Model Versioning, Feature Management and Storing, Automation of Pipelines and Processes, CI/CD for Machine Learning* e *Continuous Monitoring of Models*. Acompanhados de ferramentas comumente utilizadas para solucionar cada um desses pontos.
- **Implementations Guide**
 - **Introduction**
 - **Tools and Project Structure:** Introdução a estrutura do projeto e quais ferramentas serão utilizadas.
 - **Starting a New Project with Cookiecutter:** Introdução ao *Cookiecutter* e como criar um novo projeto a partir do *template* disponibilizado.
 - **Environment**
 - **Setting Up the IBM Environment with Terraform:** Usando o *Terraform* para criar o ambiente de desenvolvimento dos serviços da IBM com código.
 - **Managing the deployment space:** Como utilizar as ferramentas da IBM com o *Terraform*.
 - **Versioning**
 - **What is DVC?:** Introdução ao DVC, instalação e como utilizá-lo com armazenamento remoto.
 - **Data Versioning:** Usando o DVC para versionar dados e modelos.
 - **Working with Pipelines:** Criando e reproduzindo *pipelines* para treinamento e avaliação de modelos.
 - **Deployment with Watson Machine Learning:** Usando a API do Watson ML para fazer o deploy de modelos como um serviço web.

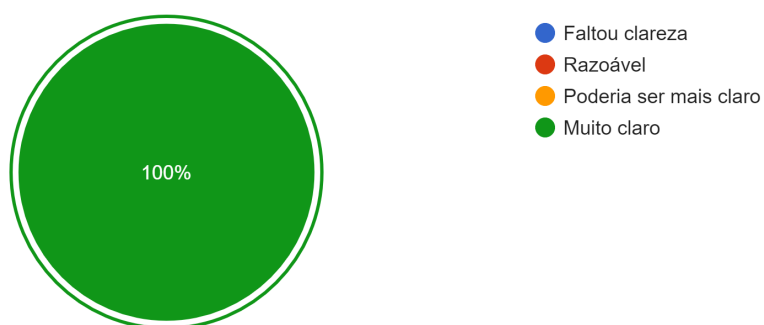
- ***CI/CD for Machine Learning***
 - ***Continuous Integration with CML and Github Actions:*** Introdução ao CML e GitHub Actions e como criar testes e gerar relatórios automaticamente.
 - ***Continuous Delivery with CML and Github Actions:*** Utilizando CML, GitHub Actions e a API do IBM Watson para fazer deployments automáticos a cada novo release .
- ***Monitoring with IBM OpenScale:*** Configurando o ambiente do OpenScale, criando instâncias de monitoramento, avaliando o modelo e explicando predições.
- ***Project Workflow:*** Demonstração de como utilizar o pipeline criado dentro de um ciclo de desenvolvimento, com os comandos e vídeos.

Apêndice B – Respostas ao Formulário de Validação do Guia

Gráfico 1 - Respostas à pergunta sobre a clareza do guia

De maneira geral, quão claro você sentiu que foi o guia?

5 respostas

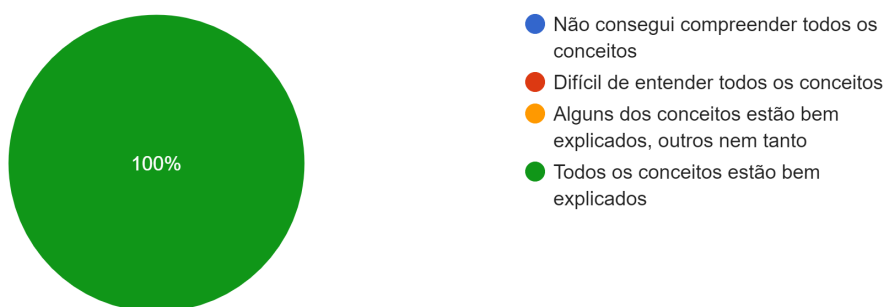


Fonte: Os autores (2021).

Gráfico 2 - Respostas à pergunta sobre a parte teórica

Como você sente que a parte teórica lidou com os conteúdos complexos?

5 respostas

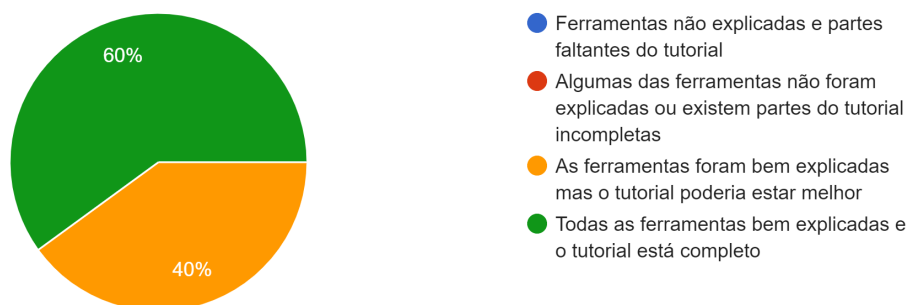


Fonte: Os autores (2021).

Gráfico 3 - Respostas à pergunta sobre a parte de implementação

Como você sente que a parte de implementação lidou com o tutorial?

5 respostas



Fonte: Os autores (2021).

Sugestão 1:

Na Home Architecture, como também (sic) na parte Tools and Project Structure, seria legal ter uma opção para dar zoom na imagem porque não (sic) consegui ler direito.

O Folder Structure também acho que poderia ter sido representado mais bonito ou melhor a imagem

Sugestão 2:

- Alguns erros de formatação.

- Fluxo da página inicial um pouco confuso, por onde começar e qual a próxima etapa.

- Algumas seções com blocos de código que não ficaram muito claros e outras onde o código é desnecessário, levando em consideração a documentação presente para determinada solução. Ex: Script Terraform.

- Algumas seções com muito texto desnecessário acabaram deixando a leitura cansativa. Ex: Pytests.

Sugestão 3:

Só detectei alguns erros de digitação (ex: Principles of MLOps/Continuous Monitoring/What to monitor?/*subtítulo*) e em algumas das imagens das primeiras (sic) partes, tive que me aproximar da tela para conseguir lê-las, mas nada que impediu a compreensão do conteúdo que estava sendo passado. Outro ponto em relação aos textos é que em diversos momentos eu achei as frases escritas um tanto estranhas. Mesmo que não estivessem gramaticalmente erradas, no contexto que foram utilizadas ficaram um pouco estranhas.

Sugestão 4:

Poderia colocar um esquema na parte de 'Automation' do guia, assim como nas outras seções, pois ajuda muito a compreender o conceito.
Além disso, rever erros de ortografia e gramática!