Integração contínua para experimentos de aprendizado de máquina Reprodutibilidade: um estudo prático

AM Andrade
Universidade Federal do Ceará
Crateús-CE, Brasil

MB Pereira
Universidade Federal do Ceará
Crateús-CE, Brasil

SES Silveira
Universidade Federal do Ceará
Sobral-CE, Brasil

FIF Linhares
Universidade Federal do Ceará
Sobral-CE, Brasil

AHO Neto
Universidade Federal do Ceará
Sobral-CE, Brasil

RMC Andrade
Universidade Federal do Ceará
Fortaleza-CE, Brasil

IL Araújo
Universidade Federal do Ceará
Fortaleza-CE, Brasil

ABSTRATO

O desenvolvimento de um modelo de Machine Learning (ML) depende de muitas variáveis em seu treinamento. Ambas as variáveis relacionadas à arquitetura do modelo , como pesos iniciais e hiperparâmetros, e variáveis gerais, como conjuntos de dados e versões de estrutura, podem afetar as métricas do modelo e a reprodutibilidade do experimento. Um aplicativo não pode ser confiável se produzir bons resultados apenas em um ambiente específico . Portanto, para evitar problemas de reprodutibilidade, algumas boas práticas precisam ser adotadas. Este trabalho tem como objetivo relatar uma experiência prática no desenvolvimento de uma aplicação de aprendizado de máquina adotando um fluxo de trabalho que garanta a reprodutibilidade dos experimentos e, consequentemente, sua confiabilidade, melhorando a produtividade da equipe.

CONCEITOS CCS

• **Software e sua engenharia** ÿ **Gestão do processo de desenvolvimento de software; • Metodologias de computação** ÿ Aprendizado de máquina .

PALAVRAS-CHAVE

Sistemas de Aprendizado de Máquina, Processo de Desenvolvimento de Software, Engenharia de Software, Engenharia de Software Inteligente, MLOps

Formato de Referência
ACM: AM Andrade, MB Pereira, SHS Silveira, FIF Linhares, AHO Neto, RMC Andrade e IL Araújo. 2021. Integração contínua para reprodutibilidade de experimentos de aprendizado de máquina: um estudo prático. In Proceedings of Brazilian Workshop on Intelligent Software Engineering (ISE' 21). ACM, Nova York, NY, EUA, 4 páginas. https://doi.org/10.5753/ise.2021.17279

1. INTRODUÇÃO

A reprodutibilidade de experimentos computacionais em trabalhos científicos é um desafio fundamental na academia e na indústria. Segundo Beaulieu-Jones e Greene [4], os resultados científicos muitas vezes só podem ser reproduzidos com a ajuda dos autores originais. Muitos fatores impactam a reprodutibilidade do experimento baseado em software, como o ambiente de desenvolvimento, ou seja, especificações de hardware e software e versões da Interface de Programação de Aplicativo (API) empregada .

Os experimentos baseados em ML, sejam da indústria ou da academia, são ainda mais complexos de reproduzir devido a muitas variáveis que afetam as métricas de desempenho dos modelos, como amostras de conjuntos de dados, hiperparâmetros dos modelos e dependência de parâmetros aleatórios usados para inicializar e atualizar modelos.

O trabalho de Amershi et al. [1] relata práticas que as equipes da Microsoft usaram em projetos de ML. Os autores afirmam que a introdução de capacidades de ML em software desenvolvido com um processo ágil vem com três complicações principais: o gerenciamento de dados; o desenvolvimento de modelos requer habilidades raras; a modularização é prejudicada. Todos esses três fatores afetam a reprodutibilidade no sentido de que os conjuntos de dados e as versões do modelo precisam ser rastreáveis, outras equipes precisam ser capazes de reproduzir e adaptar o experimento. É difícil conseguir isso se o software tiver um alto acoplamento.

Todos os fatores descritos anteriormente relacionados à reprodutibilidade impactam na confiabilidade do trabalho científico considerando o risco de ser delineado por seus pares. Segundo Pineau et al. [5], o estabelecimento de metodologias que assegurem a reprodutibilidade de trabalhos científicos na área de ML é uma das tendências na academia, uma vez que os mesmos resultados poderia ser obtido experimentando as mesmas condições. Isso também é essencial em um projeto de ML da indústria, que deve seguir um processo de método científico de hipótese, experimento e resultado avaliação.

Para lidar com o problema de reprodutibilidade dos experimentos de ML , este propõe um processo de trabalho composto por ferramentas de automação de pipeline e um fluxo de trabalho bem definido para realizar os experimentos e otimizar os modelos. No processo de desenvolvimento, foram utilizadas as seguintes ferramentas de automação: o Gitlab[1] Continuous Integration, Deployment or Delivery (CI/CD) para executar um pipeline de experimentos, Data Version Control (DVC)[2] para lidar com o versionamento de datasets, Continuous Machine Learning (CML)[3] para fornecer relatórios automaticamente, Git para controle de versão do software e Docker para executar o contêiner que suporta a execução do pipeline.

Este artigo apresenta o processo de trabalho aplicado no desenvolvimento de um Sistema de Detecção de Quedas (FDS) baseado em ML , executado pela academia e indústria em parceria, com a descrição de ferramentas, métodos e resultados esperados. O conteúdo está dividido da seguinte forma. A seção 2 apresenta uma revisão da literatura referente a este trabalho. Ele mostra detalhes de nossa abordagem proposta para o desenvolvimento desta pesquisa na Seção 3. A Seção 4 exibe os principais resultados do nosso trabalho. A Seção 5 apresenta uma discussão dos principais problemas encontrados no desenvolvimento de projetos . Por fim, as conclusões são apresentadas na Seção 6.

2 TRABALHOS RELACIONADOS

A literatura relata alguns problemas relacionados à reprodutibilidade em experimentos computacionais e projetos de ML como uma questão significativa que pode comprometer o trabalho.

Em [4], Beaulieu-Jones e Greene desenvolveram um framework geral para experimentos computacionais usando Docker e práticas de Integração Contínua (CI) . Os autores também descrevem a recorrente

1https://gitlab.com/
2https://dvc.org/
3https://cml.dev/

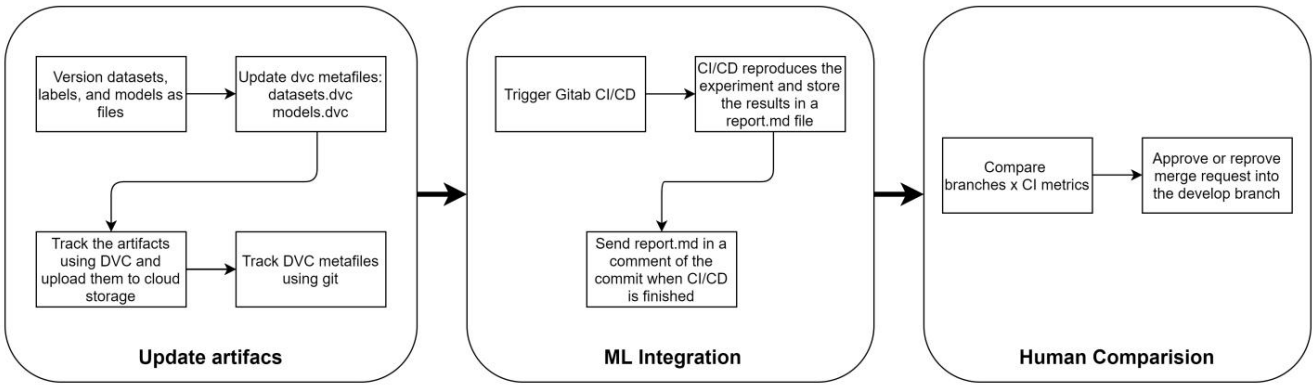

Figura 1: Processo para verificação da reprodutibilidade.

problema da falta de reprodutibilidade e seus impactos nos artigos científicos. O trabalho deles tem várias interseções com o nosso, mas não aborda especificamente os problemas do processo de ML , como sua natureza aleatória. Nosso processo também envolveu versionamento de conjuntos de dados, busca e otimização de hiperparâmetros, relatórios automáticos com ferramentas de CI/CD, entre outros.

Wan et ai. [6] apresentou uma revisão dos métodos de engenharia de software aplicados no desenvolvimento de MLmodel. A revisão aponta algumas direções para escolher o método mais adequado a implementar. Eles também defendem a tese de que a falta de um método de engenharia de software no desenvolvimento de ML pode causar diversos problemas, inclusive a possibilidade de reproduzir o experimento. Em nosso trabalho adotamos algumas práticas listadas em [6], como visualização de dados, pré-processamento, limpeza e manutenção do estado inicial dos experimentos. Além disso, iniciamos o projeto com a mentalidade de que o processo de ML é inerentemente incerto, descobrindo intuitivamente, implementando na prática ou prevendo possíveis problemas.

Wan et ai. também apresentou algumas práticas na engenharia de software ML . A principal diferença entre nossa abordagem e as utilizadas pelos praticantes de ML é que as tarefas foram bem definidas, embora um plano preciso seja mais desafiador de implementar nesse tipo de processo.

Ashmore e outros. [3] apresentou uma pesquisa relacionada às práticas de desenvolvimento de software de ML contendo instruções para definir um ciclo de vida para modelos de ML e alguns requisitos de segurança para cada estágio. Para reprodutibilidade, as atividades mais relevantes são: Pré-processamento para a fase de Gerenciamento de Dados; para a fase de Verificação do Modelo, recomendam métodos formais; para a fase de implantação, a engenharia de software tradicional pode cuidar da integração e monitoramento.
A fase de treinamento do modelo ainda precisa de algum estudo para melhorar a reutilização e interpretação. Em nosso trabalho, implementamos a maioria das sugestões, no entanto, os métodos formais na fase de verificação do modelo ainda precisam ser feitos, embora tenhamos uma abordagem bem definida para isso.

O trabalho de Argesanu e Andreescu [2] apresentou uma análise dos problemas que ocorrem no ciclo de vida do ML relacionados à automação e reprodutibilidade. Os autores desenvolvem um framework para lidar com essas questões e apresentam um estudo de caso aplicando o framework em um problema de processamento de imagens. Sua estrutura é inferência em lote orientados, com algumas diferenças para o nosso, mas ainda semelhantes em vários pontos, como atividades, ferramentas de automação, versionamento e containers.

3 MÉTODO

Aplicamos o fluxo de trabalho proposto em um projeto de ML que desenvolveu um FDS usando dados de dispositivos vestíveis. As soluções de inferência consistem em um modelo implantado em dois ambientes distintos: servidor em nuvem e nó de borda. O projeto teve a duração de seis meses e a equipa era composta por 6 membros da equipa ML , 3 membros da equipa de serviço, 1 membro da equipa QA e 2 gestores.

Nossa proposta focava na reprodutibilidade dos experimentos, manutenção do modelo e validação dos resultados no início do desenvolvimento do projeto. Para atingir esses objetivos, selecionamos as seguintes ferramentas: o Data Version Control (DVC) para versionar artefatos ML rotulando cada novo; GitLab Continuous Integration, Deployment ou Delivery (CI/CD) para lidar com o processo de integração; e o Continuous Machine Learning (CML) para gerar automaticamente relatórios contendo métricas para modelos de ML , que foram anexados a cada confirmação.

Antes do início do processo, a equipe realizou uma Grid Search para encontrar ou otimizar os hiperparâmetros do modelo. O conjunto de dados foi separado aleatoriamente por treinamento e teste.

Embora o repositório GitLab seja uma ótima opção para versionamento de códigos-fonte de projetos, não é recomendado para versionamento de conjuntos de dados e artefatos de modelos devido ao grande tamanho esperado desses artefatos . Portanto, o DVC é responsável pelos conjuntos de dados e modelos de versão e também cria metarquivos para cada artefato do projeto. Em vez de arquivos de controle de versão, o GitLab rastreia apenas esses metarquivos, que o DVC altera sempre que um artefato é atualizado. Portanto, o DVC envia os artefatos atualizados para um serviço de armazenamento externo e atualiza o respectivo metarquivo. Por fim, os desenvolvedores confirmam as alterações nos metarquivos.

Apresentamos na Figura 1 uma descrição visual de cada etapa do nosso processo proposto para este trabalho. Em primeiro lugar, na etapa Atualizar artefatos, os conjuntos de dados e a arquitetura dos modelos são confirmados na versão DVC . O sistema de ativação e seus metarquivos são atualizados e rastreados por meio do Git. Assim, na etapa ML Integration, sempre que os desenvolvedores atualizam códigos ou versões de artefatos no GitLab, o CI/CD é acionado para reproduzir o experimento atual em um ambiente Docker. Depois disso, o

O CML gera um relatório com todas as métricas relevantes do experimento . Na etapa de Comparação Humana, os novos resultados gerados automaticamente ajudam a equipe de desenvolvimento a criar solicitações de mesclagem de novo modelo e versões de software a serem analisadas pelos líderes técnicos , comparando os resultados do relatório gerado pelo CML com os resultados do README.md, arquivo gerado no passado se funde.

Em seguida, os desenvolvedores criaram tags git no repositório para cada nova versão estável de nosso modelo e modelo implantado em ambientes propostos de nuvem e borda.

Para verificar a hipótese de que a aplicação do fluxo de trabalho proposto pode detectar problemas de reprodutibilidade, trabalhamos conforme definido na Figura 2. Primeiramente, a equipe precisa configurar o ambiente e o pipeline de CI/CD . Além disso, um processo interativo realiza a geração, treinamento e execução de novos modelos. O líder da equipe é responsável por registrar e relatar os problemas de reprodutibilidade para a equipe. Portanto, a equipe realiza correções e verificações se o experimento é corretamente reproduzido automaticamente pelo Gitlab CI/CD e revisado pelo líder da equipe.

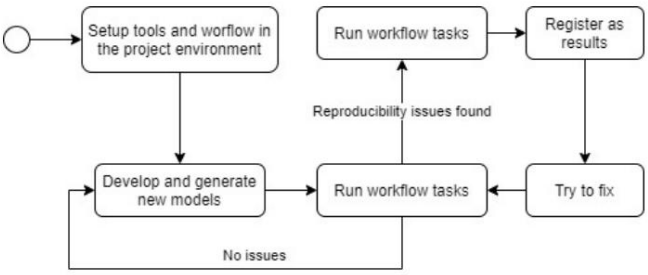

Figura 2: Identificação de problemas de reprodutibilidade

4 RESULTADOS

O projeto estudado consiste em modelos inteligentes implantados em nuvem ou borda, para detectar quedas com base em dados de sensores de acelerômetro e giroscópio . A adoção do fluxo de trabalho de integração de código proposto permite reproduzir experimentos de ML em suas versões estáveis considerando o uso correto de conjuntos de dados, métodos de pré-processamento e hiperparâmetros de modelo em diferentes ambientes, gerando os mesmos modelos e medidas.

A abordagem iterativa proporcionou um aumento gradativo na reprodutibilidade , pois permitiu rodar os pipelines e o experimento como um todo em diferentes ambientes. A quantidade total de Merge Requests é de 44, sendo 38 mescladas e 6 encerradas. Por meio dessas Merge Requests, foram executados 469 pipelines dos quais 240 já passaram. Os pipelines restantes foram cancelados ou algum problema impediu o sucesso. No início do projeto, um índice de 50,02% dos trabalhos foram bem-sucedidos e vários falharam por questões ambientais. Ao final do projeto, obteve-se um índice de aproveitamento de 87,90% dos postos de trabalho. Os pipelines e experimentos estão sendo executados em diferentes ambientes com os mesmos resultados, o que indica um aumento na reprodutibilidade.

Nas subseções seguintes, apresentamos as principais questões identificadas definidos pelo uso do fluxo de trabalho de IC e como foram resolvidos.

4.1 Incompatibilidade com versões de frameworks Muitas bibliotecas de ML compatíveis com a linguagem Python possuem um grande número de versões que podem apresentar variações internas em suas funções de uma para outra. Assim, a menor variação na versão da biblioteca pode comprometer a compatibilidade de integração do sistema, gerando erros que demandam algum tempo para a equipe encontrar a solu
A equipe relatou falhas em relação à execução da compilação de jobs CI/CD do Gitlab enquanto o ambiente do desenvolvedor funcionava corretamente . A utilização de um ambiente virtual da virtualenv tool4 resolveu esse problema, com a configuração de um conjunto de requisitos em um arquivo re quirements.txt que armazena todas as dependências do Python e suas respectivas versões, possibilitando ter ambientes semelhantes para desenvolvimento e CI. Esses ajustes resolveram problemas de compilação para seguir as execuções de trabalhos Gitlab CI/CD . Além disso, outros possíveis erros relativos a versões específicas de bibliotecas podem ser verificados comparando as métricas geradas pelo trabalho Gitlab CI/CD e as métricas obtidas no ambiente de desenvolvimento.

4.2 Problemas de sementes aleatórias

As sementes são parâmetros cruciais para produzir números pseudo-aleatórios, que, em geral, são usados para inicializar modelos de ML antes do treinamento. Os modelos de ML , mesmo que treinados nas mesmas condições, não podem produzir as mesmas métricas se as sementes escolhidas forem diferentes. Como as sementes aleatórias mudam sempre no processo de treinamento de ML , os resultados podem não ser os esperados devido à diferença das sementes. Para evitar isso, os desenvolvedores tiveram que definir sementes fixas para todas as execuções de experimentos. No aplicativo desenvolvido, as sementes aleatórias podem impactar o particionamento do conjunto de dados em treinamento e teste e os pesos e vieses iniciais aleatórios de um modelo de aprendizado profundo.

4.3 Questões de vários conjuntos de dados O conjunto de dados usado no processo de treinamento e validação de ML é composto por vários conjuntos de dados com dados de sensor compatíveis e saídas de classe após a limpeza e padronização. A geração de um único conjunto de dados de várias fontes pode afetar a reprodutibilidade do experimento devido aos procedimentos para manipular e padronizar os tipos de dados. Para resolver os problemas, o DVC foi usado para criar versões não apenas dos conjuntos de dados originais, mas também do conjunto de dados final usado para treinamento.
Uma vez que o código de treinamento usa um conjunto de dados específico (o DVC permite organizar conjuntos de dados por nome e versão), se o desenvolvedor confirmar corretamente o código e os arquivos de metadados DVC, o trabalho de treinamento do CI seleciona exatamente o mesmo conjunto de dados usado pelo desenvolvedor e repete o experimento com o mesmo conjunto de dados.

4.4 Variabilidade de hiperparâmetros A otimização de modelos de ML requer o ajuste de hiperparâmetros , que são, por exemplo, o número de kernels da Support Vector Machine (SVM) ou estimadores em Random Forest. Os desenvolvedores precisam rastrear hiperparâmetros para cada modelo gerado devido à variação das métricas dos modelos. O rastreamento da lista de hiperparâmetros em um arquivo separado e seu uso no treinamento do modelo resolveram esse problema.

4Python Virtual Env - https://docs.python.org/pt-br/3/library/venv.html

5 DISCUSSÃO

Nossa abordagem para o desenvolvimento de ML fornece um pipeline automatizado para CI, o que ajuda a obter a reprodução do modelo enquanto aumenta a confiabilidade do sistema de ML . Conforme afirmado em [6], a principal mudança da engenharia de software tradicional para o desenvolvimento de ML são as incertezas no processo, o que sugere que um pipeline bem definido não garante a reprodutibilidade do sistema. Os autores em [3] apresentaram fatores cruciais para aplicar em cada etapa do processo com alta prioridade e requisitos de risco. Portanto, adotar as práticas descritas neste trabalho com a mentalidade certa melhora a reprodutibilidade do sistema.

Conforme descrito na Seção 3, os modelos de ML para o sistema proposto deveriam poder ser implantados em dois ambientes: nuvem e borda. O ciclo de vida do ML utilizado no projeto avaliado incluiu testes nessas duas arquiteturas, portanto a reprodutibilidade foi um requisito desde o início do projeto. A equipe do projeto relatou que, antes deste trabalho, havia adotado uma integração de ciclo de vida flexível semelhante com foco na reprodutibilidade conforme proposto por Beaulieu-Jones e Greene [4]. A principal preocupação deste projeto é tornar os modelos reproduzíveis na nuvem, borda e sistemas de teste. No entanto, seu principal objetivo é armazenar todos os artefatos necessários para reproduzir um experimento com a sugestão de imagens do Docker como componente central. Embora a abordagem proposta foque no desenvolvimento de ML, há uma grande interseção com este pipeline e o apresentado em [4] , com a diferença de que este trabalho requer mais componentes para lidar com tarefas específicas de ML.

Os autores em Argesanu e Andreescu desenvolveram uma plataforma e um caso de estudo para implantar um sistema ML com menos esforço. Como em [4], Argesanu e Andreescu usaram imagens do Docker para preservar o estado do software usado no experimento e obter resultados idênticos na implantação e na produção. Ambos os trabalhos geram os mesmos resultados em um ambiente diferente usando ferramentas automatizadas de CI/CD . Assim, tem suas limitações para um produto final, como os resultados que só podem ser recuperados a partir de uma imagem Docker, que não vai inteiramente para um ambiente de produção. Além disso, neste trabalho, os modelos e os resultados foram reproduzidos fora de um contêiner, facilitando o acesso aos artefatos resultantes, ou seja, modelos e conjuntos de dados .

Além disso, o IC proposto aumentou a produtividade porque, antes de nossa abordagem ser adotada, a reprodutibilidade era verificada pelo revisor em seu ambiente. Após aplicarmos nossa abordagem, a ferramenta de revisão Merge Request permitiu uma verificação imediata da reprodutibilidade, reduzindo o tempo gasto na revisão. A maioria dos problemas relatados teve solução rápida, diminuindo o tempo e a necessidade de reuniões de equipe.

A adaptação de metodologias de engenharia de software em projetos de ML que permite reprodutibilidade em projetos da indústria pode aumentar a produtividade. Esse tipo de processo aceita a natureza incerta do processo e lida com isso da melhor maneira possível. Isso significa lidar com o ciclo de hipóteses, testes e adaptações com direção e objetivo. Além disso, isso pode gerar um sistema baseado em ML aplicável em requisitos de alta prioridade e garantia.

Para a academia, os principais pontos de contribuição são os pipelines e a reprodutibilidade para simulações computacionais, que ainda é um ponto fraco desses trabalhos [4]. Outra contribuição são os confiabilidade que depende da reprodutibilidade, implicando que os trabalhos acadêmicos de ML precisariam de menos retrações.

Uma melhoria identificada pela equipe do projeto foi que a verificação da reprodutibilidade de um experimento depende de um humano que precisa analisar a métrica e rejeitar o merge/pull request.

A principal restrição dessa abordagem é sua dependência de software de terceiros , limitando sua aplicação no tempo conforme a tecnologia continua evoluindo. Assim, o trabalho precisaria ser atualizado indefinidamente enquanto as tecnologias utilizadas não tivessem mais suporte estendido. A automação dessa tarefa é viável pelo uso de estruturas de teste automatizadas integradas aos trabalhos do Gitlab CI.

Nosso trabalho lida com conjuntos de dados de pequeno porte. Em cenários de Big Data ou visão computacional, o uso de DVC pode não ser adequado. O mesmo vale para problemas com o treinamento do modelo tem um alto custo de tempo. No nosso caso, o conjunto de dados, composto apenas pelos dados do sensor, possui 54,2 MB, e o tempo de treinamento foi inferior a cinco minutos.

6. CONCLUSÃO

Este trabalho apresentou um fluxo de trabalho prático para auxiliar no desenvolvimento de um projeto de ML considerando os requisitos de integração, manutenção e reprodutibilidade . Durante a aplicação do processo proposto, encontramos problemas semelhantes aos descritos na literatura. O ciclo de vida bem definido ajudou a equipe de desenvolvimento a reduzir o tempo gasto com reuniões devido à agilidade na reprodução dos experimentos. Além disso, o sistema de controle de versão e a integração contínua tornaram o processo rastreável, permitindo encontrar um conjunto de dados, modelo ou versão de experimento específico com facilidade por meio do sistema de tags.

Como proposta de melhorias nas estratégias de fluxo de trabalho de ML , é necessário estabelecer critérios para interromper automaticamente a integração contínua com base no desempenho das métricas do sistema e desenvolver um sistema de notificação para alertar os desenvolvedores sobre falhas ou problemas na execução do pipeline. Apesar da necessidade de melhorias, o fluxo de trabalho de CI proposto apresentou mais segurança no desenvolvimento e na organização, além de auxiliar a equipe a gerenciar a incerteza dos projetos de ML e melhorar a produtividade.

REFERÊNCIAS

[1] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi e Thomas Zimmermann. 2019. Engenharia de Software para Aprendizado de Máquina: Um Estudo de Caso. Em 2019 IEEE/ACM 41ª Conferência Internacional sobre Engenharia de Software: Engenharia de Software na Prática (ICSE-SEIP). IEEE, 291-300.

[2] Adrian-Ioan Argesanu e Gheorghe-Daniel Andreescu. 2021. Uma plataforma para gerenciar o ciclo de vida de ponta a ponta dos modelos de aprendizado de máquina de previsão em lote. Em 2021 IEEE 15º Simpósio Internacional de Inteligência Computacional Aplicada e Informática (SACI). 000329–000334. https://doi.org/10.1109/SACI51354.2021. 9465588

[3] Rob Ashmore, Radu Calinescu e Colin Paterson. 2021. Garantindo o Ciclo de Vida do Machine Learning: Desiderata, Métodos e Desafios. Computação ACM. Sobreviver 54, 5 (05 2021), 39.

[4] Brett K. Beaulieu-Jones e Casey S. Greene. 2017. A reprodutibilidade de fluxos de trabalho computacionais é automatizada usando análise contínua. Nature Biotechnology 35, 4 (01 de abril de 2017), 342–346. https://doi.org/10.1038/nbt.3780 [5] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox e Hugo Larochelle. 2020. Melhorando a reprodutibilidade na pesquisa de aprendizado de máquina (um relatório do programa de reprodutibilidade NeurIPS 2019). arXiv:2003.12206 [cs.LG]

[6] Zhiyuan Wan, Xin Xia, David Lo e Gail C. Murphy. 2019. Como o aprendizado de máquina muda as práticas de desenvolvimento de software? IEEE Transactions on Software Engineering (2019), 1–1. https://doi.org/10.1109/TSE.2019.2937083