PUC

ISSN 0103-9741

Relatórios em Ciência da Computação

nº 12/2023

Relatório de Projeto Final de Programação

Eduardo Zimelewicz

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Relatórios em Ciência da Computação, No. 12/2023 ISSN: 0103-9741

Relatório de Projeto Final de Programação

Eduardo Zimelewicz

ezimelewicz@inf.puc-rio.br

Abstract. Pulsar detection, a fundamental challenge in astrophysics, is critical for understanding the universe's celestial bodies for identifying gravity to explore planets never observed before. Leveraging the capabilities of AWS and MLOps, this project presents a robust pipeline for the deployment of a machine learning model designed to identify pulsar signals within vast sets of radio astronomy data. With this, astrophysicists will be able to upload their data to detect the phenomenon by integrating advanced data processing techniques with a meticulously trained model, which is fully extensible for new updates, other models and data integration components to provide a higher rate of adoption over the research period. The MLOps pipeline, implemented on AWS infrastructure, encompasses data preprocessing, model training, hyperparameter optimization, and model evaluation. Leveraging AWS services such as Amazon S3 for data storage, AWS Lambda for serverless computing, and Amazon SageMaker for model development, this project ensures seamless scalability and efficiency through a continuous integration and delivery (CI/CD) practice that facilitates the deployment of model updates and enhancements.

Keywords: MLOps; Cloud Infrastructure; AWS; Pulsar Detection

Resumo. A detecção de pulsares, um desafio fundamental na astrofísica, é crítica para a compreensão dos corpos celestes do universo para identificar a gravidade com intuito de explorar planetas nunca antes observados. Aproveitando os recursos da AWS e de MLOps, este projeto apresenta um pipeline robusto para a implantação de um modelo de aprendizado de máquina projetado para identificar sinais de pulsar em vastos conjuntos de dados de radioastronomia. Com isso, astrofísicos poderão fazer upload de seus dados para detectar o fenômeno, integrando técnicas avançadas de processamento de dados com um modelo meticulosamente treinado, que é totalmente extensível para novas atualizações, outros modelos e componentes de integração de dados para fornecer uma maior taxa de adoção ao longo período de pesquisa. O pipeline MLOps, implementado na infraestrutura AWS, abrange pré-processamento de dados, treinamento de modelo, otimização de hiperparâmetros e avaliação de modelo. Aproveitando serviços da AWS, como Amazon S3 para armazenamento de dados, AWS Lambda para computação serverless e Amazon SageMaker para desenvolvimento de modelos, este projeto garante escalabilidade e eficiência por meio de uma prática de integração e entrega contínua (CI/CD) que facilita a implantação do modelo com atualizações e melhorias.

Palavras-chave: MLOps; Infraestrutura de Nuvem; AWS; Detecção de Pulsares

In charge of publications:
PUC-Rio Departamento de Informática - Publicações
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: publicar@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Descrição e objetivos gerais do sotware

O projeto tem como objetivo desenvolver um sistema de detecção de pulsares usando técnicas de aprendizado de máquina, implementado em uma arquitetura em nuvem AWS. A detecção de pulsares é uma tarefa crítica em astronomia que requer modelos de machine learning precisos e um fluxo de trabalho eficiente para garantir restados confiáveis e rápidos. Pesquisadores em astrofísica serão os principais utilizadores, com o apoio de cientistas de dados para criação de modelos e engenheiros de dados para desenvolver a infraestrutura em nuvem e seu monitoramento, assim como especialistas em segurança para organizar os acessos com políticas robustas. O seguinte projeto é altamente extensível com outros componentes AWS e podendo ser externalizados por meio de APIs, filas de integração ou arquivos compatível. O intuito de projetos MLOps é de serem apresentáveis para quaisquer modelos de aprendizado de máquina que precisam ser contínuamente evoluídos e tratados de acordo com métricas relevantes ao seu objetivo. A facilidade de utilizar serviços gerenciados em nuvem se dá pela delegação de responsabilidades de administração de servidores, e seus componentes, para um provedor (Amazon Web Services) permitindo que o usuário foque no desenvolvimento de funcionalidades e resolução de problemas, implantando a solução apresentada de uma forma automática e rápida, priorizando a diminuição do tempo de início do projeto para sua disponibilidade.

2 Especificação de requisitos funcionais e não-funcionais do sotware

2.1 Requisitos Funcionais

- **Pré-processamento de Dados:** O sistema deve ser capaz de pré-processar os dados brutos provenientes de telescópios de rádio, incluindo filtragem de ruído e normalização dos dados.
- **Treinamento de Modelo:** Deve haver um processo de treinamento contínuo para o modelo de machine learning, utilizando o algoritmo de Ensembles de Random Forest [7].
- **Validação de Modelo:** O sistema deve ser capaz de avaliar a precisão do modelo por meio de métricas como precisão, recall e F1-score, para garantir a confiabilidade dos resultados.
- **Implantação Automatizada:** Deve ser possível automatizar o processo de implantação do modelo em um ambiente de produção, garantindo que as atualizações sejam aplicadas de maneira rápida e eficiente.
- **Monitoramento de Desempenho**: É necessário implementar um sistema de monitoramento que rastreie o desempenho do modelo em tempo real, alertando sobre possíveis falhas ou quedas na precisão.

2.2 Requisitos Não-Funcionais

- **Eficiência e Velocidade:** O sistema deve ser capaz de processar grandes conjuntos de dados de forma eficiente, garantindo tempos de resposta rápidos para o processa-

mento e detecção de pulsares.

- **Escalabilidade:** O sistema deve ser facilmente escalável para lidar com aumentos repentinos no volume de dados ou na demanda por recursos computacionais.
- **Confiabilidade:** O sistema deve ser altamente confiável, garantindo uma detecção precisa e consistente de pulsares, minimizando assim os falsos positivos e negativos.
- **Segurança de Dados:** Deve ser implementada uma forte segurança de dados para proteger as informações confidenciais relacionadas aos dados astronômicos e aos modelos de machine learning.
- **Manutenibilidade:** O sistema deve ser facilmente mantido e atualizado, permitindo a adição de novos recursos e a correção de bugs de maneira eficiente.

3 Modelo de arquitetura e de dados

Para alinhamento do projeto com a proposta, é necessária a utilização de MLOps para melhorias contínuas ao modelo, aprimorando sua inferência e re-treino quando drifts são detectados por ferramentas de monitoramento. A Figura 1 representa o diagrama de arquitetura de referência definida pelo guia de implementação do orquestreador de workloads da Amazon Web Services, um projeto modelo para implementação de MLOps na nuvem [3]:

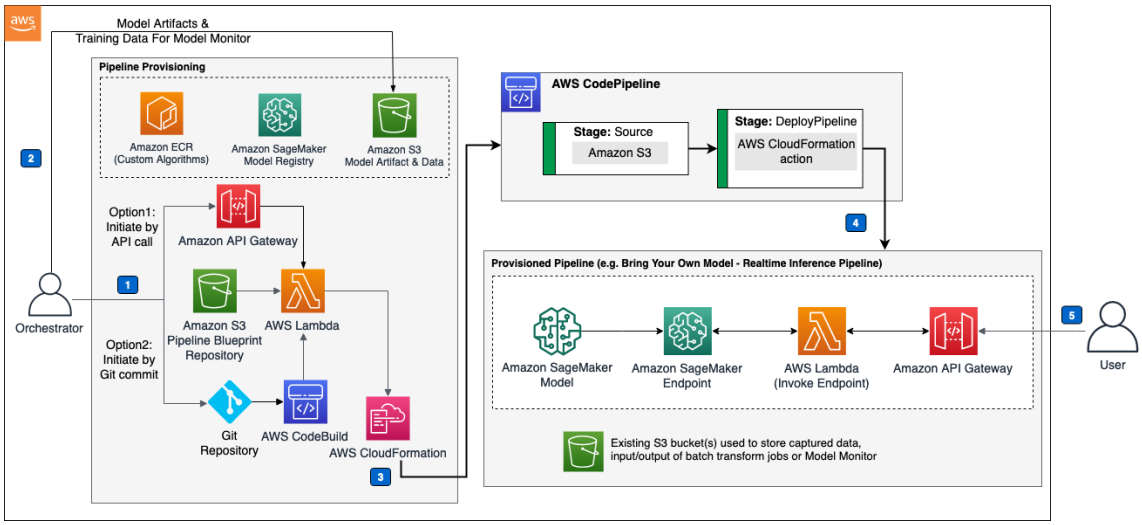

Figure 1: Diagrama arquitetural do projeto de MLOps workloads orchestrator

Prosseguindo com a descrição dos passos enumerados, no passo 1, orquestradores (desenvolvedores) acionam a construção de uma mudança no modelo por uma chamada REST API ou commit em um repositório Git. Simultâneamente, o passo 2 tem o orquestrador publicando imagens de container, carregando modelos pré-instalados no Amazon SageMaker Registry e artefatos para uso do modelo no armazenamento em Amazon S3. A seguir, no passo 3, a automação de infraestrutura como código (AWS CDK) modifica/aciona

a esteira de implantação do modelo contida no serviço AWS CodePipeline. O passo 4 executa a esteira que busca as dependências armazenadas para re-treinamento do modelo. Enfim, o serviço Amazon SageMaker gera a nova versão do modelo e a disponibiliza pelo Amazon SageMaker Endpoint. Ao final, o passo 5 demonstra como ocorre a disponibilização do modelo para inferência, também por REST API para usuários. O projeto exemplo está disponível no repositório Github em [5]

4 Modelo funcional do software

O software é composto exatamente pelos estágios de construção de um modelo de aprendizado de máquina, iniciando-se pela exploração do conjunto de dados HTRU2 [1] com 17898 instâncias de dados sobre fenômenos, contendo 16259 ruídos (não-pulsares) e 1639 pulsares. Desta forma, é explicitamente definido como um problema de classificação ou clusterização [2], especificado por 9 atributos descritos abaixo e separados como 4 estatísticas do perfil integrado (vetor de variáveis contínuas que descrevem a versão do sinal em longitude resolvida pela média de tempo e frequência) e 4 estatísticas iguais mas baseadas na curva DM-SNR (Delta Modulation Signal to Noise Ratio) [4] dos sinais e, finalmente, a classe do corpo:

- **mean** - Média do perfil integrado.
- **standard_deviation** - Desvio padrão do perfil integrado.
- **excess_kurtosis** - Excesso de curtose do perfil integrado.
- **skewness** - Distorção do perfil integrado.
- **dm_snr_mean** - Média da curva DM-SNR.
- **dm_snr_standard_deviation** - Desvio padrão da curva DM-SNR.
- **dm_snr_excess_kurtosis** - Excesso de curtose da curva DM-SNR.
- **dm_snr_skewness** - Distorção da curva DM-SNR.
- **class** - Classe

Em todo projeto de criação de um modelo de aprendizado de máquina, é necessário um maior conhecimento do conjunto de dados a partir de sua exploração. Para tal, o repositório público contém o arquivo de Jupyter notebook [6] com nome **pulsar_complete_project.ipynb** com todas os códigos em células escritos em Python, onde é registrado com o passo o passo para a replicação e execução do estudo para encontrar o modelo que possuir maior acurácia para classificar pulsares (No momento de criação deste projeto, a melhor acurácia continua sendo o método de Ensemble de Random Forest)

Após avaliação dos modelos pela exploração, as práticas de preparação do modelo são construídas pelo paradigma programação orientado a objetos. Com os arquivos criados no notebook contidos no diretório **pulsar_ml**, também escritos em Python. As classes especificadas, e suas responsabilidades, são:

- **Loader:** Carregamento do conjunto de dados para construção

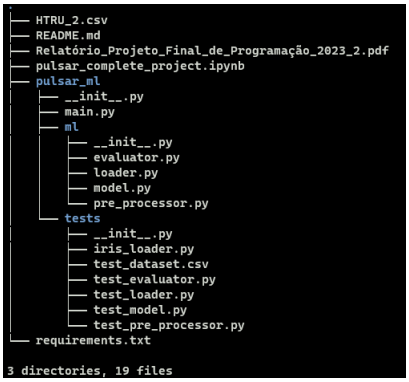

Figure 2: Estrutura de diretórios do projeto

- **Pre-processor:** Tratamento do conjunto de dados
- **Model:** Criação do modelo
- **Evaluator:** Avaliação da capacidade de classificação do modelo

Há também a presença de testes unitários, em código Python, para constante acompanhamento das funcionalidades propostas pelo projeto também dentro do mesmo diretório, só que separado na pasta **tests**.

Para a parte de infraestrutura em nuvem na AWS, são utilizados recursos providenciados pelo guia de implementação em [3], a Figura 2 demonstra a organização.

5 Manual de utilização

O projeto descrito possui dois tipos de execução, a local, onde apenas o conjunto de dados de pulsares são processados e um modelo é criado junto de seu projeto orientado a objetos. E a remota, em que toda uma infraestrutura de MLOps pode ser criada para acionamento de mudanças para melhorar o modelo ou atualizá-lo com base nas métricas monitoradas. Pesquisadores em Astrofísica, assim como proffisionais de TI com foco em dados (engenheiros de dados, arquitetos de dados e cientista de dados), poderão utilizar o software das duas formas, a escolha de execução se dará por disponibilidade e execução em coletivo, quanto maior a contribuição das partes, maior será o aproveitamento do projeto

5.1 Utilização com sucesso

Caso 1: Um astrofísico consegue carregar novos dados no conjunto existente e reexecuta todas as células do Jupyter notebook. Ele é experiente em importar dados e sabe caminhar por um notebook. Testando localmente as suas mudanças antes de aplicá-las no repositório principal para implatanção em produção, que deverá acionar a pipeline de MLOps.

Caso 2: Um engenheiro de dados está procurando modificar a integração de dados para execução da pipeline por streaming em vez de batch. Ele é um experiente arquiteto de dados onde conhece a arquitetura fim-a-fim. Isso o permite aplicar mudanças em um

ambiente na nuvem para testes, já que os arquivos de CDK são modularizados por estruturas, possibilitando a criação mais rápida de uma nova versão da infraestrutura para implantação em produção.

5.2 Utilização com problemas

Caso 1: Um astrofísico precisa gostaria de criar um novo ambiente de infraestrutura para um colega inexperiente, sem que a infraestrutura fique comprometida. Como não possui experiência em arquitetura de dados, não possui capacidade para criar diferentes ambientes, precisando acompanhar a criação pela ajuda de um engenheiro de dados.

Caso 2: Um engenheiro de dados precisa gerar uma nova versão do conjunto de dados para um colega astrofísico que não pôde realizar. Ele não conhece muito bem e jupyter notebbok mantido, é preciso que ele acompanhe a descrição das células para realizar o procedimento. Como não é experience em ciência de dados, executa as células e registra os testes localmente para avaliação posterior do colega astrofísico

References

[1] https://archive.ics.uci.edu/dataset/372/htru2.

[2] https://datacamp.com/blog/classification-machine-learning.

[3] https://docs.aws.amazon.com/solutions/latest/mlops-workload-orchestrator/solution-overview.html.

[4] https://en.wikipedia.org/wiki/Delta_modulation.

[5] https://github.com/aws-solutions/mlops-workload-orchestrator.

[6] https://jupyter.org/about.

[7] https://towardsdatascience.com/random-forests-an-ensemble-of-decision-trees-37a003084c6c.