Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência Da Computação

ADAPTFlow uma Arquitetura de Seleção de Modelos de Machine Learning

Rômulo Luiz Araujo Souza Soares

JUIZ DE FORA
SETEMBRO, 2024

ADAPTFlow uma Arquitetura de Seleção de Modelos de Machine Learning

Rômulo Luiz Araujo Souza Soares

Universidade Federal de Juiz de Fora

Intituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência Da Computação

Orientador: Regina Marcia Maciel Braga Villela

Coorientador: Jose Maria Nazar David

JUIZ DE FORA

SETEMBRO, 2024

ADAPTFlow uma Arquitetura de Seleção de Modelos de Machine Learning

Rômulo Luiz Araujo Souza Soares

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Regina Marcia Maciel Braga Villela
Doutora em Engenharia de Sistemas e Computação - UFRJ

Jose Maria Nazar David
Doutor em Engenharia de Sistemas e Computação - UFRJ

Prof. Mário A. R. Dantas
PhD em Ciência da Computação - University of Southampton (UK)

Prof. Victor Stroele de Andrade Menezes
Doutor em Engenharia de Sistemas e Computação - UFRJ

JUIZ DE FORA

16 DE SETEMBRO, 2024

Aos meus amigos e irmãos.

A minha mãe, meus avós e família por todo o apoio.

Resumo

Os softwares modernos estão operando cada vez mais dinamicamente em condições incertas, impulsionando o uso de arquiteturas adaptativas em conjunto com técnicas inteligentes, principalmente *Machine Learning* (ML), para lidar com essas incertezas. No entanto, a seleção e gerenciamento dos modelos de ML representam desafios significativos devido às constantes mudanças nos dados. Este Trabalho de Conclusão de Curso propõe uma forma dos sistemas autoadaptativos se tornarem mais assertivos utilizando de uma arquitetura, denominada ADAPTFlow, para gerenciar a seleção de algoritmos de machine learning, de acordo com os dados a serem processados. O objetivo é minimizar a degradação do processamento inteligente em arquiteturas auto-adaptativas, a partir do uso de tecnologias como AutoML. A arquitetura foi avaliada utilizando um conjunto de dados da indústria têxtil, resultando em uma acurácia média de 80%.

Palavras-chave: Inteligencia Artificial, Arquiteturas adaptativas, AutoML, Aprendizado de Máquina, ADAPTFlow

Como os sistemas autoadaptativos podem se tornar mais assertivos com a seleção de modelos de aprendizado de máquina?

Abstract

Mordem software increasingly operates dynamically in uncertain conditions, driving adaptive architectures with intelligent techniques, mainly Machine Learning (ML), to deal with these uncertainties. However, the selection and management of ML models represent significant challenges due to constant changes in data. This TCC proposes a way for self-adaptive systems to become more assertive by using an architecture, called ADAPT, to manage the selection of ML algorithms, according to the data to be processed. The aim is to minimize the degradation of intelligent rocessing in self-adaptive architectures, using technologies such as AutoML. The architecture was evaluated using a dataset from the textile industry, resulting in an average accuracy of 80%.

Keywords: Artificial Intelligence, Adaptive Architectures, AutoML, Machine Learning, ADAPTFlow

Agradecimentos

Agradeço profundamente à minha família, que me ensinou os valores da disciplina, do esforço e da dedicação, além de me apoiar incondicionalmente em cada escolha que fiz ao longo desta jornada.

Minha gratidão também vai aos meus orientadores, Regina e José Maria, que, com paciência e comprometimento, acompanharam cada etapa deste processo. Suas orientações valiosas foram fundamentais, não só para a elaboração deste trabalho, mas para todo o período de iniciação científica. Sem sua colaboração, este TCC não seria possível.

Aos professores do Departamento de Ciência da Computação, expresso meu sincero agradecimento por todos os ensinamentos que enriqueceram meu desenvolvimento pessoal e profissional.

Por fim, agradeço aos meus amigos, que estiveram ao meu lado em todos os momentos, compartilhando tanto as alegrias quanto as dificuldades, e proporcionando momentos de descontração e lazer, inclusive nas longas noites. Um agradecimento **in memoriam** ao meu mestre de RPG, cuja lembrança tornou essa caminhada ainda mais especial e significativa.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
DT	Digital Twins
AutoML	Automated Machine Learning
TCC	Trabalho de conclusão de curso

1 Introdução

A Engenharia de Software Moderna (VALENTE et al., 2020) envolve algumas tecnologias que há pouco tempo não eram consideradas no contexto do desenvolvimento de software convencional. Dentre estas, destaca-se a Internet das Coisas (GILCHRIST, 2016), Inteligência Artificial (MICHALSKI; CARBONELL; MITCHELL, 2013) e Arquiteturas Auto Adaptativas (WEYNS, 2020).

A Internet das Coisas é uma revolução tecnológica cujo principal paradigma é reunir tanto o mundo tecnológico quanto o virtual com o objetivo de distribuir aplicações por meio de objetos do dia a dia. Sendo assim, ela pode gerar uma grande quantidade de dados que podem ser processados de forma a gerar *insights* que podem melhorar o desempenho dos dispositivos e softwares relacionados. O uso dessas análises de dados para melhorar o funcionamento de equipamentos e software faz parte de uma nova abordagem em engenharia de software moderna, denominada sistemas autoadaptativos (WEYNS, 2020). Para esse intuito, um sistema que oferece suporte a uma abordagem adaptativa requer técnicas baseadas no conhecimento, ou inteligência artificial (MICHALSKI; CARBONELL; MITCHELL, 2013) para fornecer soluções, de forma que caracterize uma solução auto adaptável (AGRAWAL; SRIKANT et al., 1994).

No contexto da indústria 4.0, os sistemas que suportam uma grande quantidade de dados para serem analisados com intuito de dar suporte à tomada de decisão são primordiais. As arquiteturas adaptativas apresentam uma abordagem para monitorar, compreender e otimizar o comportamento de outro sistema durante seu tempo de execução. Assim, um serviço com suporte ao processamento inteligente de dados pode oferecer suporte para o desenvolvimento de arquiteturas inteligentes para a indústria 4.0.

Considerando estes desafios, ou seja, dados gerados por sensores de IoT e seu processamento inteligente com vistas a melhorar o funcionamento de equipamentos e software, este trabalho apresenta uma arquitetura para seleção de modelos de ML para apoiar o desenvolvimento de aplicações complexas em Engenharia de Software, ou melhor, aplicações que necessitam de suporte inteligente para o processamento das informações.

O objetivo é processar as informações, a partir do uso de um conjunto de modelos de aprendizado de máquina.

Para verificar a viabilidade da arquitetura, um estudo no domínio da indústria têxtil é explorado. O estudo detalha uma solução arquitetural auto-adaptativa que coleta e analisa dados relacionados a eventos de falha de equipamentos industriais. A arquitetura detecta a falha e os resultados da análise dos dados, que utiliza a abordagem proposta neste trabalho, geram alertas e, consequentemente, servem como uma abordagem de apoio à decisão para melhorar a manutenção preditiva dos equipamentos. Portanto, investigamos como a proposta deste Trabalho de Conclusão de Curso pode auxiliar na seleção de modelos de ML para apoio a análise de dados, a partir da captura de informações oriundas de sensores.

Para isso, este trabalho está organizado da seguinte forma: o próximo Capítulo apresenta os principais conceitos envolvidos nesta pesquisa. O Capítulo 3 detalha uma Revisão Sistemática relacionada ao estado da arte das pesquisas relacionadas às arquiteturas auto adaptativas que utilizam técnicas de IA para a tomada de decisões no contexto da manutenção de equipamentos. O Capítulo 4 apresenta Materiais e Métodos, detalhando a proposta do trabalho. No Capítulo 5 é detalhado um estudo de viabilidade e o Capítulo 6 apresenta as conclusões do trabalho.

2 Referencial Teórico

Neste Capítulo, são apresentados alguns conceitos fundamentais para a compreensão do trabalho.

2.1 Sistemas Adaptativos

Softwares modernos devem operar de forma dinâmica em condições incertas e com foco no cumprimento de múltiplos objetivos (WEYNS, 2020; ANGELOPOULOS et al., 2018). Para (WEYNS, 2020) algumas possíveis causas para essas incertezas são as mudanças no ambiente operacional, a disponibilidade de recursos, dinamicamente, e as variações nos objetivos do usuário. Já (COMBEMALE et al., 2021) diz que a adaptação dos sistemas de forma autônoma, a fim de ajustar suas conFigurações para atender às variações da carga de trabalho, é um importante desafio.

O foco dos sistemas adaptativos é permitir que eles coletem informações adicionais sobre as incertezas durante a operação do sistema com intuito de gerenciar mudanças, com base nas metas do sistema (WEYNS, 2020). Assim, quando um evento adverso é detectado, por exemplo, quando uma nova configuração é efetivada, o sistema se adapta (ANGELOPOULOS et al., 2018) às necessidades mais recentes.

De acordo com (WEYNS, 2020), existem duas interpretações mais comuns para definir os sistemas adaptativos: (i) o sistema deve ser capaz de ajustar seu comportamento em resposta a mudanças que ocorrem em seu ambiente; e (ii) o sistema deve utilizar um mecanismo interno e externo, através dos quais, o mecanismo interno permite que o sistema lide com eventos inesperados, ou indesejados, e o mecanismo externo funciona como um sistema de *feedback* em *loop* que monitora e adapta o sistema durante a sua execução.

Em (WEYNS, 2020), é definido um modelo conceitual de um sistema adaptativo composto por quatro elementos básicos: (i) ambiente (***environment***), (ii) sistema de gerenciamento (***management system***), (iii) feedback em loop (***feedback loop***) e (iv)

os objetivos de adaptação (***adaptation goals***), (Figura 2.1).

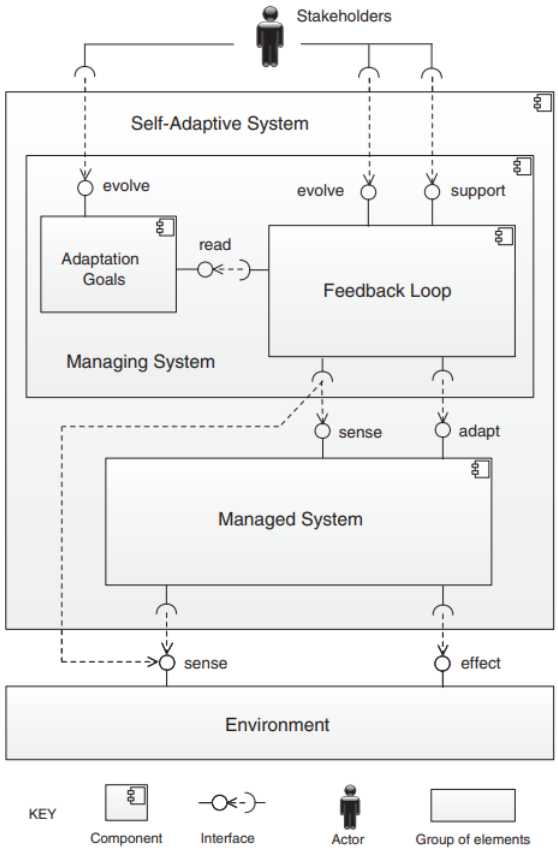

Figura 2.1: Modelo conceitual, retirado de (WEYNS, 2020)

2.2 Aprendizado de Máquina

Aprendizado de Máquina (*Machine Learning* - ML) pode ser definido como um processo de desenvolvimento de algoritmos ou modelos que aprendem com base em exemplos previamente definidos (BURZYKOWSKI et al., 2023). Já (KUFEL et al., 2023) define ML como um subconjunto da Inteligência Artificial que envolve a criação de modelos computacionais capazes de tomar decisões ou prever informações com base em conjunto de dados informados.

Os algoritmos de ML podem ser utilizados com diversos propósitos, como, aplicações de classificação, predição, detecção de anomalias e agrupamento (BURZYKOWSKI et al., 2023). Já (BALOGLU; LATIFI; NAZHA, 2022) informa que os modelos de ML podem ser utilizados para diversos tipos de resultados, como a regressão para predizer valores contínuos, como por exemplo, o comprimento de uma garrafa e a classificação que foi

desenvolvida para produzir valores categóricos, como, por exemplo, os estágios de câncer de um paciente.

Os algoritmos de ML podem ser divididos em modelos supervisionados, não supervisionados e por reforço. Os modelos supervisionados utilizam um conjunto de dados dividido em *labels* (categorias), pré definidos, cujo o foco é aprender como atribuir os *labels* para novos conjuntos de dados que não pertencem aos dados de treinamentos. Os algoritmos não supervisionados utilizam de um conjunto de dados sem *labels* e o objetivo é localizar categorias naturais, e assim classificar novos dados. Modelos por reforço envolvem uma combinação dos sistemas supervisionados e não supervisionados para aprender a classificar novos conjuntos de dados (BIAMONTE et al., 2017).

Os algoritmos de ML envolvem uma grande quantidade de hiperparâmetros que precisam ser informados antes de serem utilizados (PROBST; BOULESTEIX; BISCHL, 2019). Para selecionar as configurações dos hiperparâmetros podem ser utilizados os valores padrões definidos nas implementações dos pacotes de algumas linguagens de programação, ou manualmente se baseando em recomendações obtidas na literatura, por experiência, tentativa e erro ou por estratégias de *tuning* (PROBST; BOULESTEIX; BISCHL, 2019).

2.2.1 Auto ML

A complexidade das aplicações de ML tem criado uma demanda de modelos que podem ser utilizados sem a necessidade de um conhecimento avançado. O resultado disso levou ao desenvolvimento de pesquisas com foco na automação progressiva dos modelos de ML (AutoML) (HUTTER; KOTTHOFF; VANSCHOREN, 2019). AutoML é definido por (HE; ZHAO; CHU, 2021) como a construção automatizada de um pipeline de ML com limitação computacional.

Os métodos de AutoML incluem a preparação dos dados, engenharia dos recursos, geração e avaliação dos modelos (KOREN et al., 2022). Algumas ferramentas que permitem a implementação de pipelines de AutoML são AutoSklearn[1], PyCaret[2] e Au-

[1](FEURER et al., 2020)
[2](ALI, 2020)

toWeka[3].

Como os modelos de aprendizado possuem alta complexidade para implementação, alterações no contexto dos dados podem impactar negativamente na qualidade dos modelos. Dessa forma, no contexto das arquiteturas adaptativas, que estão em constante mudanças, surge a necessidade de que os modelos sejam gerenciados a fim de evitar a degradação dos atributos de qualidade das arquiteturas.

Neste sentido, este trabalho propõe uma abordagem que pode auxiliar neste contexto. No Capítulo 4 será detalhada esta proposta.

[3](KOTTHOFF et al., 2017)

3 Mapeamento Sistemático

Este capítulo apresenta o mapeamento sistemático de literatura com o objetivo de apresentar o estado da arte em relação à utilização de suporte à seleção de modelos de aprendizado de máquina.

3.1 Pergunta Orientadora

Com o objetivo de verificar os trabalhos na literatura relacionados ao tema e como estes solucionam problemas relacionados à pesquisa, foi realizado um mapeamento sistemático da literatura (CORDEIRO et al., 2007). A questão norteadora da pesquisa foi definida da seguinte forma: "**Como os sistemas autoadaptativos podem se tornar mais assertivos com a seleção de modelos de aprendizado de máquina?**".

Para isso, foi especificado o modelo PICOC, apresentado na tabela abaixo:

Tabela 3.1: Tabela PICOC

P	Problema	Sistema para suporte de decisão
T	Intervenção	Modelos de aprendizado de máquina
C	Comparação	-
O	Outcome	Um conjunto de modelos
C	Tempo, tipo de estudo	-

A partir do PICOS, foram derivadas as seguintes palavras-chave:

- machine learning
- adaptive systems
- adaptive architecture
- self-adaptation
- predictive decision
- decision making

A String de busca foi assim definida para cada uma das bibliotecas:

- **Scopus**: *TITLE-ABS-KEY (“machine learning”) AND (TITLE-ABS-KEY (“adaptive systems”) OR TITLE-ABS-KEY (“adaptive architecture”) OR TITLE-ABS-KEY (“adaptive system”) OR TITLE-ABS-KEY (“self-adaptation”)) AND (TITLE-ABS-KEY (“decision making”) OR TITLE-ABS-KEY (“predictive decision”)) AND PUBYEAR ¿ 2012 AND PUBYEAR ¡ 2025 AND (LIMIT-TO (OA , ”all”))*

- **Web of Science**: *ALL= (“machine learning”) AND (ALL= (“adaptive systems”) OR ALL= (“adaptive architecture”) OR ALL= (“adaptive system”) OR ALL= (“self-adaptation”)) AND (ALL= (“decision making”) OR ALL= (“predictive decision”)) and 2023 or 2022 or 2020 or 2021 or 2019 or 2018 or 2017 or 2016 or 2015 or 2014 or 2013 (Publication Years) and Open Access*

Utilizando as strings de buscas nas bibliotecas relacionadas (Scopus e Web of Science), foram retornados 58 artigos. Desses, 38 eram duplicados, resultando em 20 artigos finais. Com base nesse quantitativo, após a leitura de cada resumo dos artigos, foram selecionados 11 artigos para que fosse feita uma leitura mais extensa a fim de identificar quais eram realmente relacionados ao tema deste trabalho. Foram identificados 2 artigos relevantes, a remoção dos artigos ocorreu pois os artigos encontrados, mesmo sendo artigos que abordavam sistemas autoadaptativos ele não abordavam a implementação e impacto de um modelo de aprendizado de máquina. A partir de *snowballing*, foram identificados 2 artigos também aderentes à busca.

Em (GOMES et al., 2023), é apresentada uma plataforma de ecossistema de software que utiliza de componentes de sistemas autoadaptativos a fim de lidar com decisões voltadas para a produção animal. Para a tomada de decisão, são utilizados algoritmos de aprendizado profundo para construir um modelo de predição, porém o autor teve o foco em mostrar como o ecossistema funciona em conjunto com os sistemas adaptativos e não em selecionar o melhor modelo de aprendizado de máquina para o contexto aplicado.

Em (KACHI; BOUANAKA, 2023), é apresentado um modelo híbrido com foco na

tomada de decisão no contexto de sistemas adaptativos. O modelo definido, denominado DLA4EDM, utiliza de aprendizado profundo e análise quantitativa para tomada de decisão eficiente. Contudo, o modelo DLA4EDM utiliza somente de um modelo de aprendizado de máquina não sendo abrangente a outros tipos de modelo de ML, o que de certa forma impacta os resultados obtidos quando ocorre uma mudança do contexto dos dados devido uma alteração da origem dos dados obtidos.

Em (RODRíGUEZ-GRACIA et al., 2019) é proposto um micro serviço voltado para prédios verdes com foco na adaptação de sensores de forma que controle, por exemplo, a temperatura do ambiente controlando a velocidade de um ventilador. Para realizar a adaptação é utilizado um classificador, que utiliza o método de correlação baseado na seleção de *features* (CFS), definindo as melhores *features* para realizar o treinamento e validação do classificador. Após a geração desse novo modelo de regras, é feito uma comparação com o modelo antigo. De forma geral, esse micro serviço é voltado para a definição de classificadores que funcionem para um contexto de Prédios Verdes e utiliza somente alguns classificadores e não chegando a utilizar modelos de aprendizado de máquina, o qual é o foco deste trabalho.

O trabalho apresentado em (CORDEIRO et al., 2007) detalha a utilização de modelos de aprendizado de máquina por reforço em conjunto com métodos de adaptação de aprendizado de máquina para ajustar o conhecimento para melhorar a tomada de decisão, de forma que sejam montados modelos probabilísticos que sumarizam todas as informações para realizar as predições. No artigo são descritos diversos experimentos utilizando 9 modelos tendo um modelo selecionado como o melhor com base nas métricas. Esses experimentos foram realizados de forma a avaliar o aprendizado por reforço para um contexto de tomada de decisão.

Realizando uma comparação com os artigos apresentados acima, o foco é demonstrar a utilização de diversos modelos dentro de um contexto determinado. Mesmo que alguns selecionem o melhor modelo para aquele contexto, eles não definem o melhor modelo para o contexto de forma automática.

Em específico, (GOMES et al., 2023) não tem foco na escolha do melhor modelo, mas os autores utilizaram a proposta apresentada neste trabalho para aprimorar sua

pesquisa, conforme apresentado em (SILVA, 2024).

4 Materiais e Métodos

Este Capítulo detalha a arquitetura desenvolvida, apresentando seus principais componentes. Para isso, o trabalho foi desenvolvido seguindo a metodologia Design Science Research (DSR) (VENABLE; PRIES-HEJE; BASKERVILLE, 2016).

4.1 DSR

A abordagem Design Science Research (DSR) pode ser caracterizada como uma metodologia de construção de soluções por meio da elaboração, desenvolvimento e avaliação de artefatos. Em contraste com abordagens que se limitam à observação e análise de eventos, a DSR foca em projetar, implementar e verificar soluções que cumpram objetivos específicos (HEVNER et al., 2010). De acordo com (PIMENTEL; FILIPPO; SANTOS, 2020) existem diferentes propostas de condução da DSR, mas em geral todas têm dois objetivos claramente definidos: i) produzir um artefato e ii) produzir conhecimento técnico-científico.

Para a produção do artefato, ou seja a arquitetura proposta, algumas etapas foram realizadas. A condução de um **mapeamento sistemático** representa a etapa de entendimento de trabalhos relacionados e consolidação das bases teóricas. O **desenvolvimento da solução** é a fase dedicada à implementação prática da proposta delineada ao longo da pesquisa. Paralelamente, a execução de **testes e experimentos** desempenha um papel importante, envolvendo a aplicação da solução em conjuntos de dados sintéticos e do mundo real, seguida pela análise dos resultados obtidos.

Para o desenvolvimento da solução proposta nesta pesquisa foi utilizada a linguagem de programação *python*. A escolha dessa linguagem se deve a ampla gama de bibliotecas e tutoriais para implementação dos algoritmos de aprendizado de máquina. A Biblioteca principal que foi utilizada foi o *PyCaret*, que consiste em uma biblioteca *open source* e *low code* para o desenvolvimento de aplicações que envolvem IA. A escolha dessa biblioteca se deu por se tratar de uma biblioteca de fácil utilização e não sendo

necessário implementar cada modelo de *machine learning*, sendo *low code*. Como nesta pesquisa o intuito não é implementar modelos mas sim realizar os estudos de como os métodos de *machine learning* impactam na tomada de decisão, a biblioteca se adaptou a estas necessidades.

Em conjunto foi utilizada também a biblioteca *Streamlit,* para auxiliar no desenvolvimento da interface gráfica. A partir do uso desta biblioteca, é possível exibir os resultados obtidos para cada modelo. Essa biblioteca auxilia no desenvolvimento de interfaces gráficas com integração de códigos em *python*. A aplicação final permite ao usuário informar um conjunto de dados históricos. Com base nesse *dataset* é realizada a seleção dos 3 melhores modelos com base na acurácia dos mesmos. A partir dessa seleção, uma *api* para cada modelo é gerada, o que permite monitorar os modelos, e visualizar a classificação para um conjunto de dados aleatórios baseado no valores presentes dentro do *dataset*, com intuito de analisar e monitorar os modelos selecionados.

4.2 Arquitetura ADAPTFlow

Nessa seção, é detalhada a ADAPTFlow que consiste em uma arquitetura que auxilia a seleção de modelos de aprendizado de máquina, com foco em modelos supervisionados, através da utilização de conceitos de AutoML, dentro do contexto de arquiteturas adaptativas. Vale ressaltar que essa arquitetura funciona para modelos de classificação que utilizam duas classes.

A arquitetura utiliza um processo de treinamento automatizado, aprimoramento dos hiperparâmetros e a seleção de *features* automatizadas, para que possa selecionar o melhor modelo possível. Esses processos fazem parte do conceito de Auto ML. Assim, conforme já dito, para o auxílio do desenvolvimento dessa arquitetura foi utilizado a biblioteca *PyCaret*.

A solução proposta surgiu com o intuito de desenvolver uma arquitetura que auxiliasse na seleção de algoritmos de aprendizado de máquina no contexto de arquiteturas adaptativas, tendo como foco a arquitetura ser de simples integração, onde o desenvolvedor pudesse ter o foco no desenvolvimento de aplicação em particular, sem se preocupar com a seleção do melhor modelo de IA a ser utilizado. O principal objetivo da arquitetura é

dar suporte ao desenvolvimento de aplicações com IA, que tenham necessidade de utilizar modelos de ML para processar auto-adaptação. É importante destacar que o foco é na seleção de modelos ML e não nas características de autoadaptação.

Assim, para o desenvolvimento da arquitetura, foram definidos requisitos funcionais e não funcionais:

Tabela 4.1: Requisitos Funcionais e Não Funcionais

Requisitos Funcionais	
RF1	Classificar mais de 5 tipos de algoritmos
RF2	A arquitetura deve permitir o carregamento dos dados no formato CSV
RF3	Fornecer funcionalidade de pré-processamento dos dados
RF4	Realizar os ajustes dos hiperparâmetros
RF5	Fornecer métricas de avaliação
RF6	Salvar os modelos de forma reutilizável
RF7	Permitir a integração através de API
Requisitos não Funcionais	
RNF1	Deve processar um grande volume de dados
RNF2	Deve ser escalável
RNF3	Deve garantir a segurança dos dados de treinamento
RNF4	Deve ser fácil de manter e atualizar.
RNF5	Deve ser modular
RNF6	Deve permitir uma integração fácil com outros sistemas

De forma mais específica, o **RF1** define que a arquitetura precisa utilizar no mínimo 5 modelos de aprendizado de máquina com o objetivo de comparar e classificar o melhor modelo com base na acurácia ou *recall*. **RF2 está relacionado** aos dados para treinamento, que devem ser fornecidos no formato csv. **RF3** define que a arquitetura deve possuir um processo de pré-processamento que inclui a limpeza e transformação dos dados. **RF4** define que a arquitetura precisa realizar os ajustes dos hiperparâmetros de forma automática. **RF5** define que a arquitetura precisa informar ao usuário as métricas obtidas de forma que justifique a escolha do modelo. **RF6** define que a arquitetura precisa salvar os modelos em um formato que permita a reutilização. **RF7** define que a integração com outros sistemas seja feita através de uma API.

Já relacionado aos requisitos não funcionais eles definem quais as metas de qualidade da arquitetura São eles: i) deve lidar com o volume de dados para realizar o treinamento, ii) deve garantir a segurança dos dados dos usuários,iii) deve ser um código modular e de fácil manutenção, iv) as integrações externas não devem ser complexas e

v) ser uma arquitetura escalável que permite diversos sistemas realizarem integrações de forma síncrona.

Assim, a arquitetura especifica todo o processo para gerar um modelo de aprendizado de máquina de forma que o desenvolvedor da aplicação final precise interagir o mínimo possível, sendo necessário somente enviar os dados e a arquitetura lida com todas as outras etapas, o usuário envia o seu conjunto de dados e recebe uma modelo treinado e o próprio sistema lida com as etapas de pré-processamento, treinamento e teste, salva os dados e por último disponibiliza para o usuário, conforme demonstrado pela Figura 4.1.

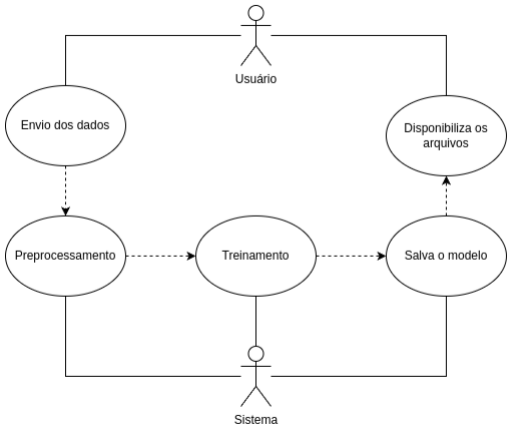

Figura 4.1: Diagrama de caso de uso

A Figura 4.2, apresenta uma visão geral da arquitetura, que pode ser separada em uma aplicação central e uma *dashboard* auxiliar. A aplicação central é dividida nos seguintes módulos: (i) **preparação de dados** onde é realizado um pré processamento do dataset, (ii) **treinamento dos modelos** onde é realizado o treinamento de 15 algoritmos de aprendizado de máquina e são selecionado os "n" melhores, e (iii) o módulo ***tunning*** onde os "n" modelos passam por um processo de tuning dos hiperparâmetros, a fim de realizar um ajuste fino e melhorar a qualidade dos modelos.

Tendo os modelos sido treinados e ajustados, o módulo "**salva o modelo"** salva os modelos treinados em um arquivo no formato *pickle* e, em paralelo, o módulo **deploy de API de predição** realiza a criação de uma API simples para auxiliar o usuário a implementar o modelo final. Por fim, o módulo **disponibilização dos arquivos** converte os arquivos gerados para o formato base64 com o intuito de auxiliar no envio dos dados através de um arquivo json para o usuário final.

Já no lado da *dashboard* encontramos os seguintes módulos: **(i) análise exploratória**, responsável por gerar um relatório do *dataset* contendo informações úteis para auxiliar o usuário/desenvolvedor a entender o *dataset* e realizar algumas alterações no mesmo, se necessário. O módulo **(ii) testes dos modelos**, auxilia como os modelos treinados, a realizar consultas com valores aleatórios a fim de gerar gráficos visuais para o usuário/desenvolvedor entender como o modelo se comporta. E o módulo **(iii) treinamento do modelo** é responsável por permitir que o usuário submeta seu *dataset* ao treinamento.

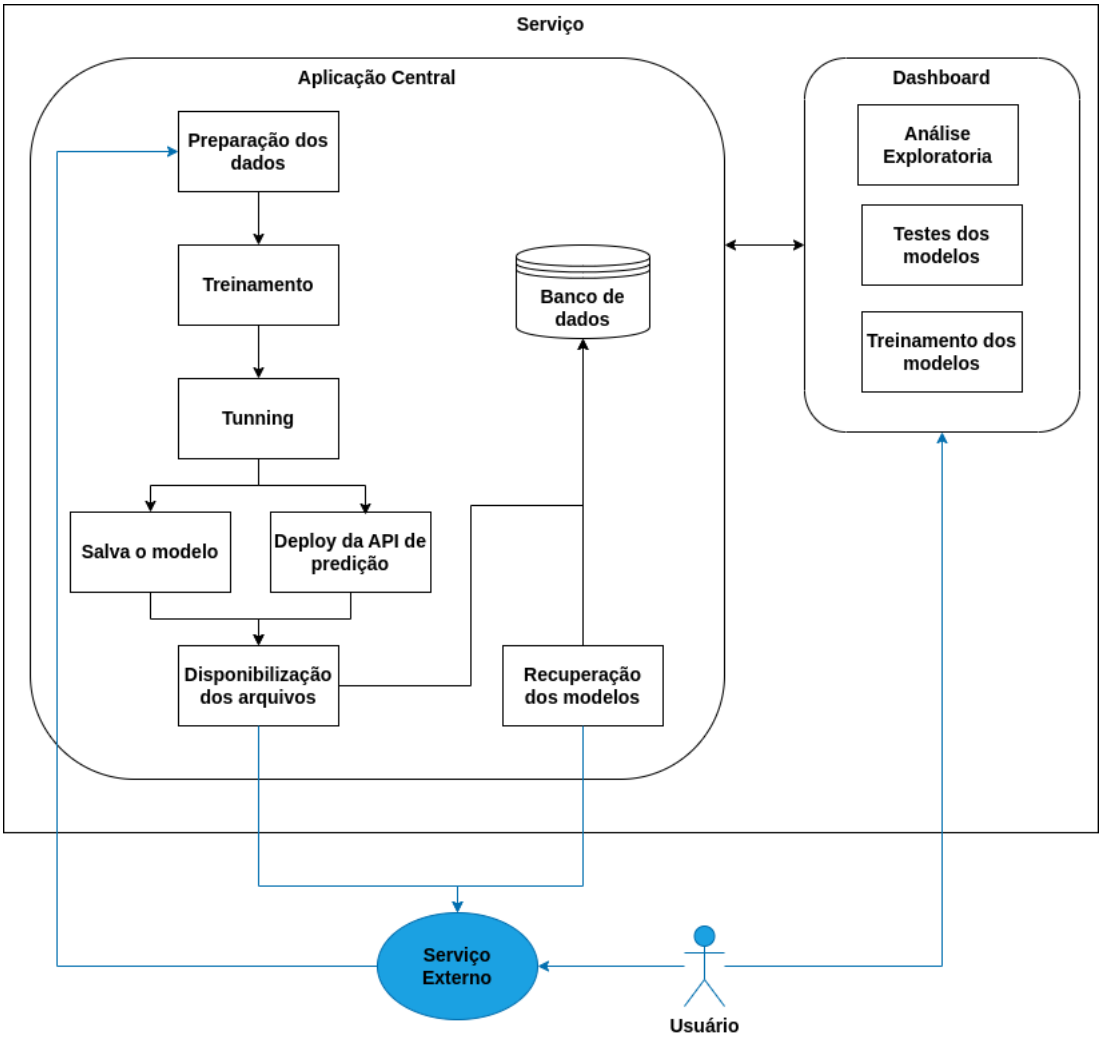

Figura 4.2: Visão Geral da Arquitetura ADAPTFlow

De forma mais específica, o módulo de **preparação dos dados** é responsável por definir qual o alvo para realizar o treinamento, ou seja, qual coluna do *dataset* possui as classes de classificação do dado, qual a proporção da divisão dos dados, tratamento dos valores ausentes, para valores numéricos utiliza do algoritmo *K-Nearest Neighbors* (KNN)

e para os valores categóricos utiliza o valor mais frequente , remoção de dados *outliers* utilizando do algoritmo *IsolationForest* e divisão dos dados para realização da validação cruzada de forma a utilizar um *fold* de 3 para o momento do treinamento dos dados, conforme apresentado na Figura 4.2.

Já o módulo de **treinamento** é configurado para classificar em até 14 modelos pré configurados e pode fazer a seleção com base na acurácia, precisão ou *recall*, todo o processo de treinamento dos modelos é feito em validação cruzada, conforme configurado no modulo de preparação dos dados. Em seguida o módulo de ***tunning*** busca a melhor configuração (hiperparâmetros) de forma randomizada para os modelos selecionados no módulo anterior.

Os módulos **"salva o modelo"** e ***deploy* de API de predição** funcionam em conjunto de forma que ambos salvam o modelo no formato *pickle* e produzir um código simples para facilitar a implementação da ferramenta de predição.

E por último, o módulo de **disponibilização dos arquivos** é responsável por disponibilizar os arquivos gerados para usuário ou para um serviço externo de forma que facilite a integração do modelo gerado a uma arquitetura autoadaptativa.

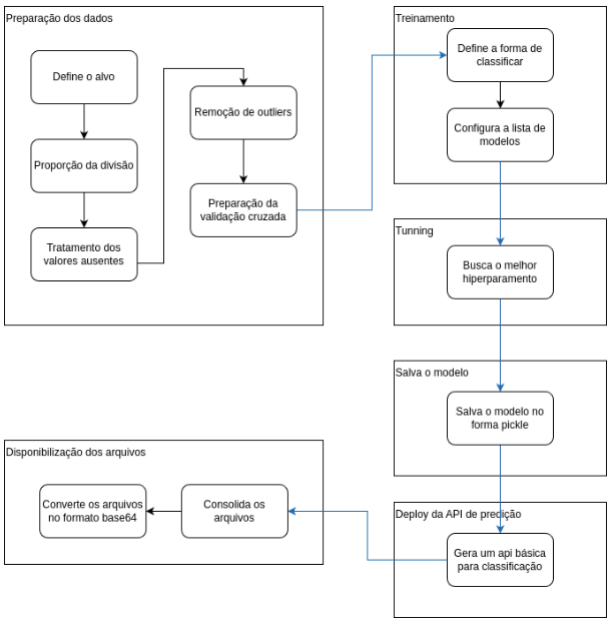

Figura 4.3: Arquitetura ADAPTFlow detalhada

A integração dos componentes da arquitetura ADAPTFlow é ilustrada na Figura 4.4. Um sistema externo pode interagir através de uma API. Neste processo, o sistema faz

uma solicitação API para realizar o treinamento dos modelos. Após a requisição na API, os dados são encaminhados para uma etapa de pré-processamento. Em seguida, ocorre a seleção do número de modelos solicitados. Os modelos então passam pelo processo de ajuste (tuning). Após o treinamento dos modelos e definição dos hiperparâmetros, eles são consolidados em arquivos que facilitam sua utilização por outros sistemas. Por fim, esses arquivos são disponibilizados novamente através da API.

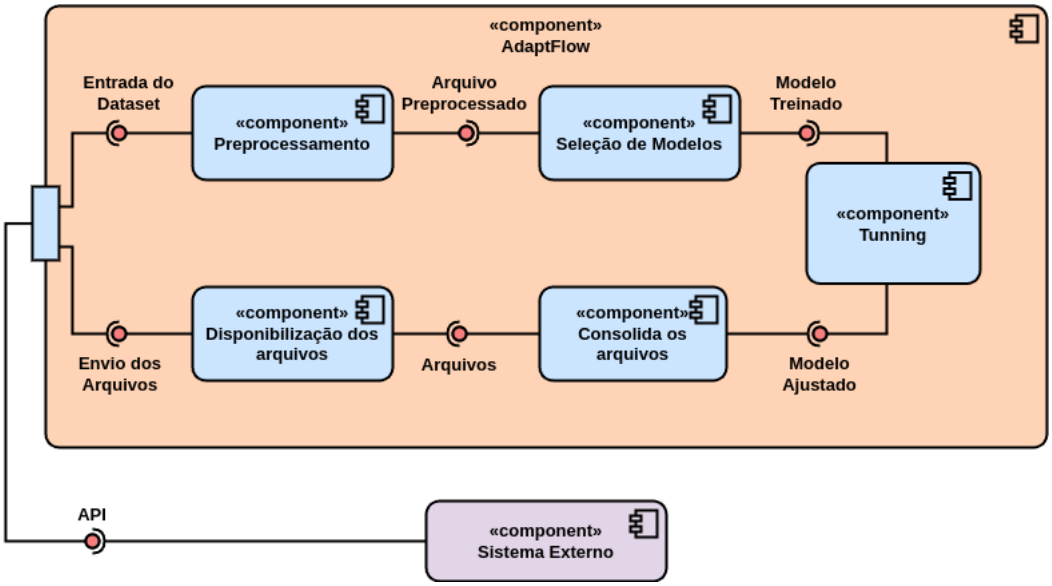

Figura 4.4: Diagrama de componentes

4.3 Uso dos serviços disponibilizados pela Arquitetura ADAPTFlow

Para facilitar a utilização da arquitetura ADAPTFlow, foi proposto um processo para integrar os serviços externos de forma simples e prática, esse fluxo é apresentado na Figura 4.5, na qual inicialmente o usuário precisa fornecer um *dataset* histórico do seu ambiente contendo os conjuntos de dados e o significado dos mesmos (*target*). Em seguida, o usuário necessita definir quantos modelos devem ser treinados, e qual a métrica para classificar os modelos. Essas informações devem ser passadas para API. Com esses dados, a aplicação pode realizar a requisição na *API* e, com o retorno, deve converter o arquivo dos modelos treinados do formato base64 para arquivo no formato *pickle*, e assim carregar

o modelo utilizado como, por exemplo, a biblioteca *joblib* do *python*. Por fim, realizar as consultas dos novos dados para que seja feita a classificação dos dados.

Assim, para a utilização da arquitetura, é disponibilizada uma *api*, que permite o usuário realizar a requisição informando o *dataset*, o quantitativo de modelos de aprendizado de máquina (até 14 modelos) e qual a métrica de classificação (Acurácia ou *Recall*). Com estes dados, a ADAPTFlow irá identificar os n melhores modelos e irá retornar os modelos no formato base64.

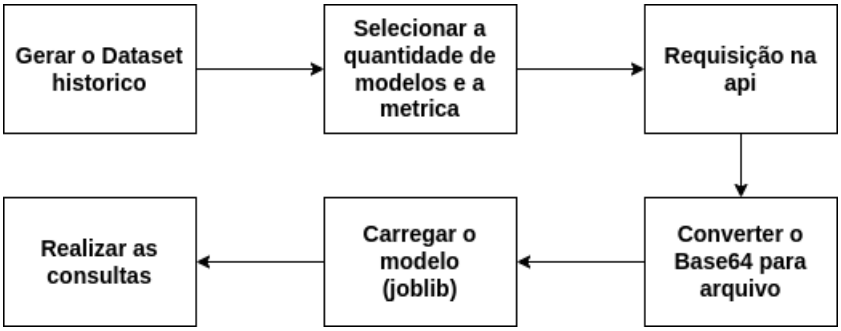

Figura 4.5: Fluxo de utilização proposto

4.3.1 Serviços disponibilizados no Dashboard

Em conjunto com a API da ADAPTFlow, foi desenvolvido também um *dashboard*, na Figura 4.6 é apresentado um recorte geral desse *dashboard*, cada parte dele será descrito mais detalhadamente abaixo. Neste *dashboard*, o usuário pode realizar testes para entender o funcionamento da arquitetura, além de poder validar o dataset antes de utilizar de forma automática. Esse *dashboard* é dividido em 3 módulos que funcionam de forma independente entre si, são eles: i)Análise exploratória dos dados, ii) Teste dos modelos e iii) Treinamento dos modelos.

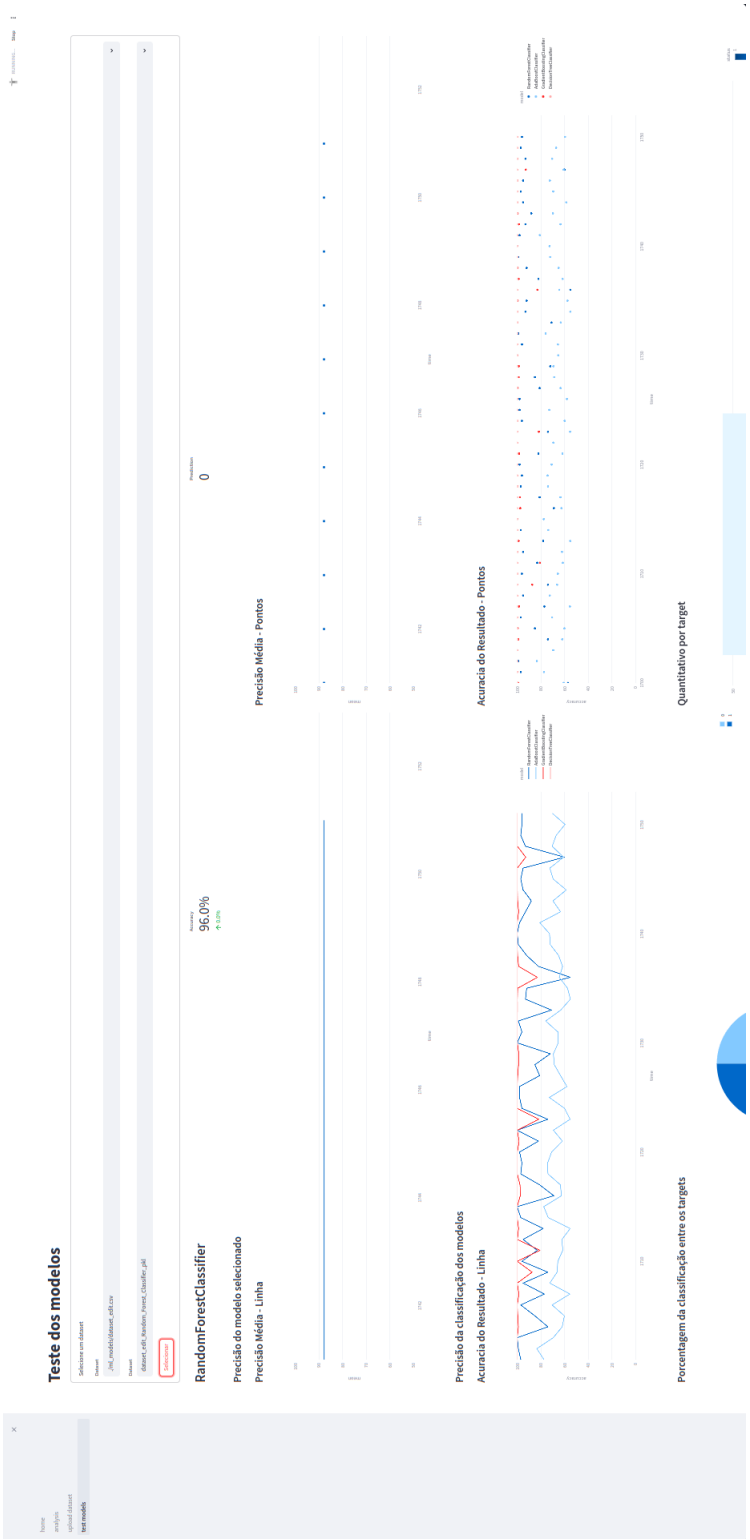

Figura 4.6: Matriz de Correlação

i) Análise exploratória dos dados: A interface de análise exploratória possui a função de gerar relatórios que proporcionam aos usuários insights sobre o conjunto de dados de maneira clara e visualmente atrativa. Foi utilizado um conjunto de técnicas de visualização para facilitar a compreensão rápida e profunda dos dados. Os recursos for-

necidos, são: (i) a descrição detalhada do *dataset* (Figura 4.7) utilizando estatísticas descritivas como média, mediana, desvio padrão, mínimo, máximo e quartis para a variáveis numéricas; (ii) matriz de correlação (Figura 4.8) que exibe visualmente as relações entre as diferentes variáveis do conjunto de dados, ajudando a identificar correlações significativas. Análise das Médias por *Target* (Figura 4.9) que calcula e apresenta a média dos valores para cada categoria do *target*, oferecendo insights sobre como o *target* se comporta em relação às variáveis do *dataset*.

Descrição

Total de **7** colunas e **9999** linhas

As colunas são **['type_of_failure', 'time_repair', 'cost', 'criticality', 'humid', 'temp', 'label']**

	count	mean	std	min	25%	50%	75%	max
type_of_failure	9,999	5.4719	2.8577	1	3	5	8	10
time_repair	9,999	0.326	0.2356	-0.6305	0.2124	0.282	0.344	1.9778
cost	9,999	0.4272	0.2241	-0.843	0.295	0.422	0.539	1.712
criticality	9,999	0.4744	0.1761	0	0.325	0.476	0.559	0.912
humid	9,999	45.0221	23.3578	5	25	45	65	85
temp	9,999	86.6711	36.8131	24	54	86	119	150
label	9,999	0.1703	0.3759	0	0	0	0	1

Figura 4.7: Tabela de Descrição (Retirado do *dashboard*)

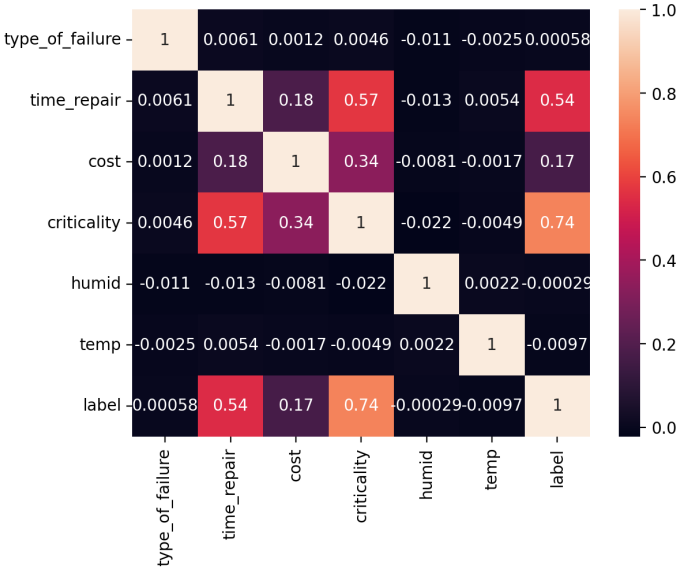

Figura 4.8: Matriz de Correlação (Retirado do *dashboard*)

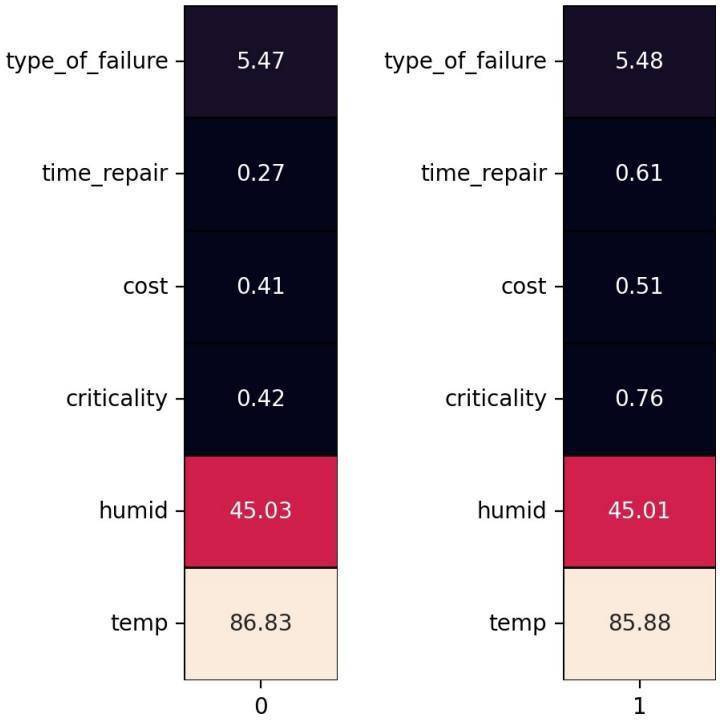

Figura 4.9: Valores médios por *target* (Retirado do *dashboard*)

ii) Testes dos Modelos: O módulo de testes é responsável por executar testes dos modelos gerados utilizando de valores novos gerados de forma aleatória para demonstrar o funcionamento de um algoritmo de aprendizado de máquina, durante esse processo, uma série de gráficos são apresentados em tempo real. Inicialmente a interface apresenta qual algoritmo está sendo utilizado, a acurácia atual e qual o valor da predição (Figura 4.10). No primeiro conjunto de gráficos (Figura 4.11) é exibido a precisão média, em um gráfico de linha e ponto, para um modelo selecionado. O segundo conjunto de gráficos (Figura 4.12) apresenta a acurácia de todos os modelos existentes para o *dataset* e por último mostra o quantitativo de acerto por *target* do *dataset*. O último conjunto de gráficos apresenta a distribuição dos resultados classificados (Figura 4.13). Dessa forma, permite ao usuário conhecer o funcionamento dos algoritmos selecionados e assim auxiliando ao mesmo entender se o determinado modelo atende suas metas.

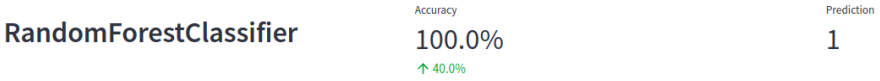

Figura 4.10: Resultado classificado (Retirado do *dashboard*)

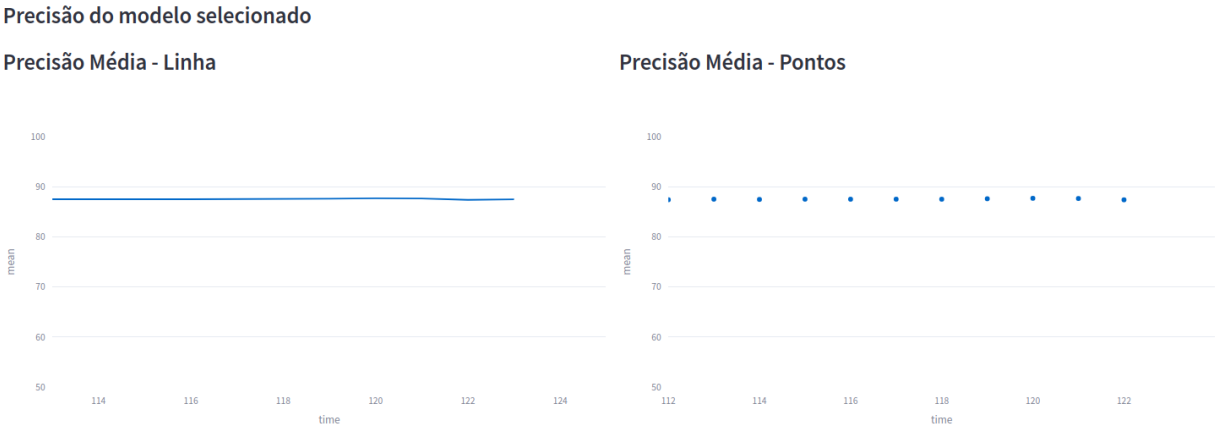

Figura 4.11: Precisão média por modelo (Retirado do *dashboard*)

Figura 4.12: Acerto dos modelos (Retirado do *dashboard*)

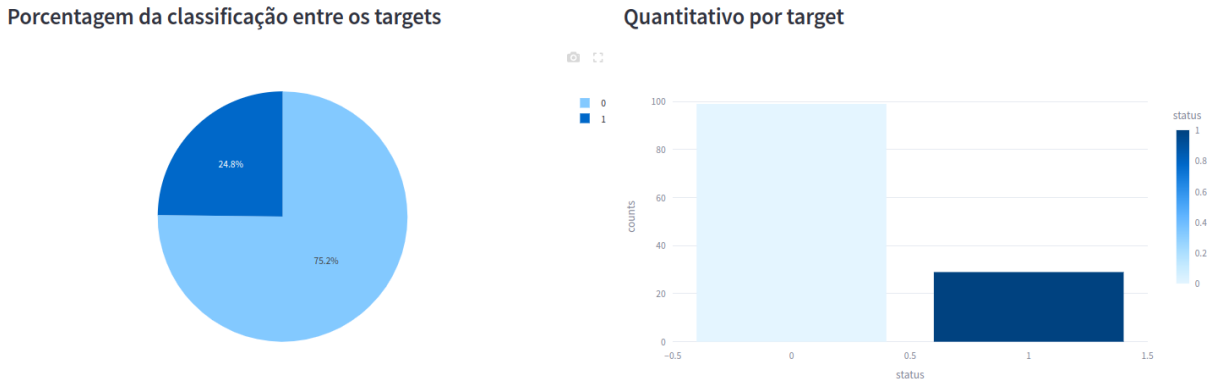

Figura 4.13: Distribuição da Predição (Retirado do *dashboard*)

iii) Testes dos Modelos: Por último a *dashboard* apresenta uma interface simples para que o usuário possa enviar um conjunto de dados para teste, o que permite o mesmo testar seu *dataset* set e validar se a arquitetura ADAPTFlow atende suas expectativas.

Para acessar o código da arquitetura e do dashboard apresentados acima estão disponíveis através do links **GitHub ADAPTFlow API** e **GitHub ADAPTFlow Dashboard**

5 Ambiente e Resultados

Para realizar os testes comparativos do sistema ADAPTFlow, foi utilizado um *dataset* obtido na internet, através da plataforma Kaggle. O *dataset* selecionado é voltado para a manutenção preditiva. A escolha deste *dataset* se deu por conta dos dados serem relacionados ao mesmo contexto do trabalho que deu origem a essa pesquisa e pela quantidade de registros presente. A Figura 5.1 apresenta brevemente o *dataset* utilizado.

UDI	Product ID	Type	Air_temperature_K	Process_temperature_K	Rotational_speed_rpm	Torque_Nm	Tool_wear_min	Target	Failure_Type
6974	L54153	L	300.7	311	1462	41.9	126	0	No Failure
8313	M23172	M	298.8	310	1529	38.5	77	0	No Failure
3903	L51082	L	302.2	311.1	1630	34.6	94	0	No Failure
3163	L50342	L	300.5	309.9	1480	40	148	0	No Failure
1376	M16235	M	298.8	310.3	1528	37.1	99	0	No Failure
6984	M21843	M	300.7	311	1334	53.9	150	0	No Failure

Figura 5.1: **Dataset para testes**

O foco dos testes foi classificar se um determinado conjunto de dados apresenta falha ou não. Para a realização dos testes foi feita um divisão no *dataset* de forma que diferentes tipos de tratamentos fossem aplicados e cada *dataset* foi processado pela ADAPTFlow. Ao final, os resultados foram comparados com os resultados obtidos pelos usuários no Kaggle, conforme apresentado na Figura 5.2. Para que os testes tivessem resultados satisfatórios, os conjuntos de dados foram submetidos várias vezes (10 no total) a ADAPTFlow e foi utilizado o resultado médio da acurácia dos modelos.

O foco desses testes foi analisar e obter *insights* sobre o funcionamento da ADAPTFlow, buscando entender quais tipos de tratamentos são obrigatórios e como esses tratamentos afetam os resultados obtidos.

5.1 Pré-processamento dos dados

Para realizar os testes foram feitos três tipos de tratamentos diferentes. Para os três tratamentos, as colunas *UID*, *ProductID* e *FailureType*, foram removidas por se tratar de dados que identificam de quais máquinas os dados eram oriundos.

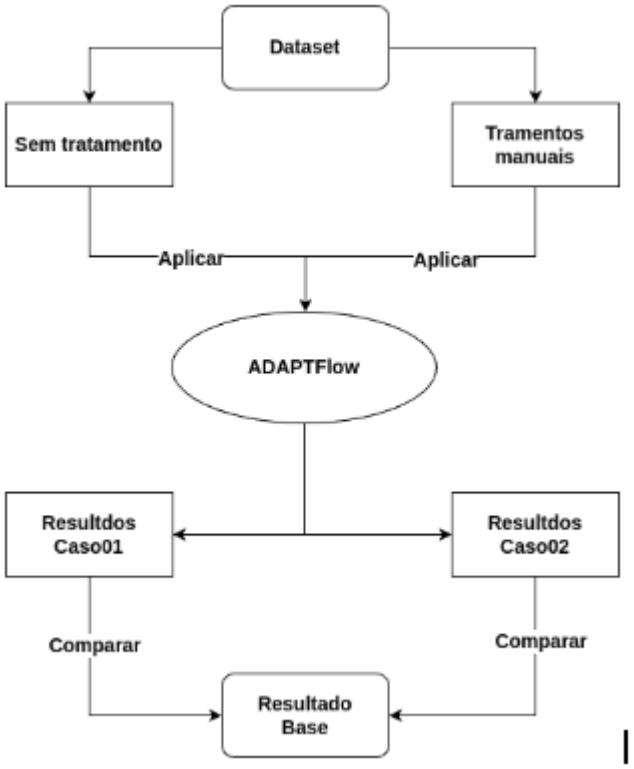

Figura 5.2: Fluxo de teste

O primeiro tratamento teve o foco de fazer o mínimo de alterações possíveis no *dataset*, de forma que os dados ficassem o mais original possível. Dessa forma, a única alteração feita foi o ajuste da coluna Type para valores numéricos.

O segundo e o terceiro tratamento foram realizado seguindo um dos códigos apresentados na plataforma, sendo ajustado a coluna Type para valores numéricos e as colunas que apresentam as temperaturas (Air_temperature_K e Process_temperature_K) tiveram a conversão de Kelvin para Celsius. Além disso, o terceiro dataset teve um tratamento para corrigir o desbalanceamento dos dados.

5.2 Resultados

Os resultados obtidos são apresentados na Figura abaixo. Para cada modelo de aprendizado de máquina, é apresentada a acurácia média obtida. Com base nos resultados pode-se observar que os pré-processamentos realizados não tiveram um impacto significativo no resultado final, dado que comparando os resultados para um mesmo modelo a

diferença ficou abaixo de 1%. Também é possível observar que o arquivo 3 teve melhores resultados de forma geral. Considerando este resultados, as próximas análises serão feitas utilizando esse arquivo.

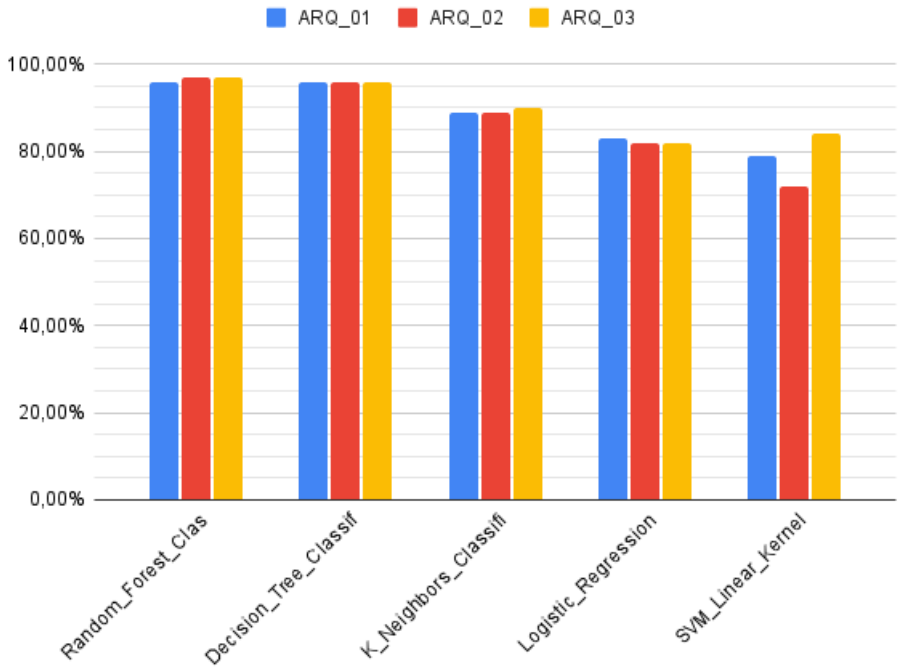

Figura 5.3: Resultados Classificados por arquivo

No gráfico seguinte (Figura 5.4) é apresentado a comparação entre o arquivo 03 e os resultados obtidos na internet. É possível observar que a ADAPTFlow teve resultados bem próximos para os algoritmos de floresta randômica e árvore de decisão, tendo uma diferença menor que 5%, já para os outros modelos essa diferença ficou superior a 10%.

De forma geral, os ajustes feitos nos dados não tiveram um impacto significativo nos resultados obtidos, de forma que dependendo do treinamento, essa diferença dos resultados pode variar. Com isso, é interessante realizar um conjunto de testes para cada resultado e utilizar a média dos valores para obter uma comparação mais apropriada.

Mesmo a ADAPTFlow apresentando resultados abaixo, os resultados obtidos foram bons. Vale ressaltar que a ADAPTFlow obteve o melhor resultado com o algoritmo LightGBM, tendo uma acurácia de 98%. Esse modelo não foi apresentado nas comparações, pois não foi identificado um resultado para que fosse possível executar comparações.

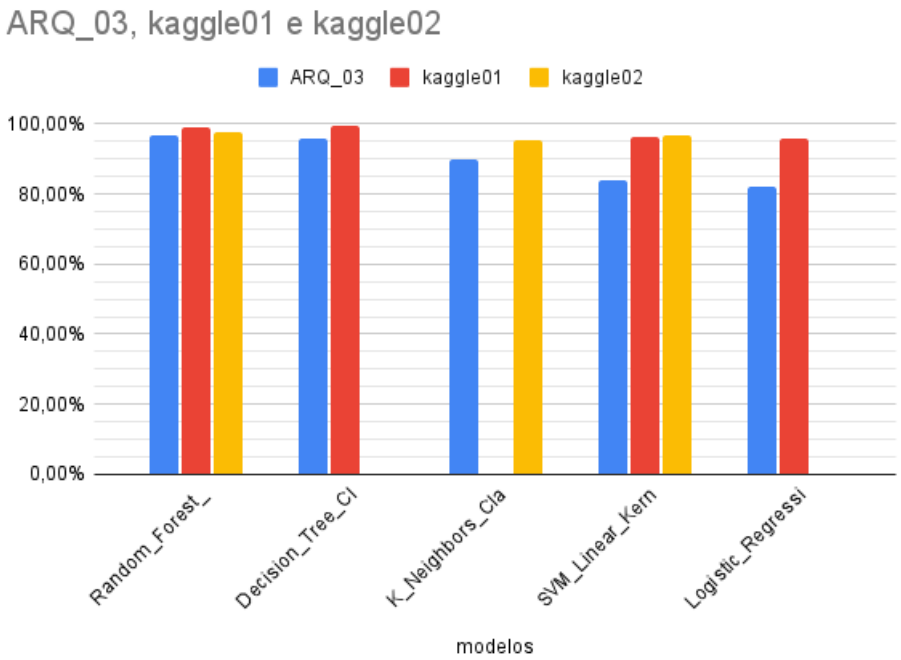

Figura 5.4: Comparação dos Resultados

5.2.1 Processamento de dados da indústria têxtil

Conforme ressaltado na introdução deste trabalho, a arquitetura ADAPTFlow foi utilizada para seleção de modelos de ML para um conjunto de dados relacionados a industria têxtil. Este processamento faz parte de um trabalho de dissertação de mestrado do Programa de Pós-Graduação de Ciência da Computação. Detalhes do uso da abordagem e de sua evolução considerando dados de uma industria de motores são apresentados em (SILVA, 2024)

6 Conclusões e Trabalhos Futuros

Neste trabalho, foi apresentada a arquitetura ADAPTFlow, que visa auxiliar aplicações com processamento inteligente na seleção de modelos de aprendizado de máquina, com ênfase em modelos supervisionados de classificação. A arquitetura utiliza conceitos de AutoML para automatizar o processo de seleção de modelos. Para facilitar o uso da arquitetura e seleção de modelos, também foi desenvolvido um *dashboard* interativo que auxilia o desenvolvedor a selecionar modelos.

Para avaliar a arquitetura proposta, foram realizados testes de desempenho com resultados de referência disponíveis na literatura, demonstrando a qualidade e eficiência do funcionamento da ADAPTFlow. Os resultados obtidos foram próximos aos encontrados na literatura com um erro de aproximadamente 5%, evidenciando a eficácia da solução proposta.

A arquitetura ADAPTFlow também foi utilizada no desenvolvimento do trabalho (SILVA, 2024), de forma que o processo de tomada de decisão sobre qual o modelo de aprendizado de máquina poderia ser utilizado para os dados selecionados, foi apoiado pela ADAPTFlow.

Como trabalhos futuros podemos citar a implementação de modelos de classificação para mais de duas classes e de outros modelos de aprendizado de máquina, como regressão linear, além do aprimoramento da *dashboard* para integração com uma API. Esse aprimoramento permitirá que o usuário envie conjuntos de dados diretamente pela API, possibilitando uma análise prévia do desempenho do modelo antes de sua aplicação em ambiente de produção.

Além disso, um projeto de iniciação científica em andamento, propõe o uso da arquitetura ADAPTFlow para outros domínios de aplicação, por exemplo, a classificação de dados do mercado financeiro. Essa aplicação tem contribuído para identificar pontos de melhoria na arquitetura, com o objetivo de expandir sua aplicabilidade para diferentes contextos de dados.

Bibliografia

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: SANTIAGO. *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.], 1994. v. 1215, p. 487–499.

ALI, M. *PyCaret: An open source, low-code machine learning library in Python*. [S.l.], 2020. PyCaret version 1.0. Disponível em: ⟨https://www.pycaret.org⟩.

ANGELOPOULOS, K. et al. Engineering self-adaptive software systems: From requirements to model predictive control. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, ACM New York, NY, USA, v. 13, n. 1, p. 1–27, 2018.

BALOGLU, O.; LATIFI, S. Q.; NAZHA, A. What is machine learning? *Archives of Disease in Childhood - Education and Practice*, Royal College of Paediatrics and Child Health, v. 107, n. 5, p. 386–388, 2022. ISSN 1743-0585. Disponível em: ⟨https://ep.bmj.com/content/107/5/386⟩.

BIAMONTE, J. et al. Quantum machine learning. *Nature (London)*, Nature Publishing Group, England, v. 549, n. 7671, p. 195–202, 2017. ISSN 0028-0836.

BURZYKOWSKI, T. et al. Introduction to machine learning. *American Journal of Orthodontics and Dentofacial Orthopedics*, v. 163, n. 5, p. 732–734, 2023. ISSN 0889-5406. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0889540623000793⟩.

COMBEMALE, B. et al. A hitchhiker's guide to model-driven engineering for data-centric systems. *IEEE software*, IEEE, Los Alamitos, v. 38, n. 4, p. 71–84, 2021. ISSN 0740-7459.

CORDEIRO, A. M. et al. Revisão sistemática: uma revisão narrativa. *Revista do Colégio Brasileiro de Cirurgiões*, Colégio Brasileiro de Cirurgiões, v. 34, n. 6, p. 428–431, Nov 2007. ISSN 0100-6991. Disponível em: ⟨https://doi.org/10.1590/S0100-69912007000600012⟩.

FEURER, M. et al. Auto-sklearn 2.0: Hands-free automl via meta-learning. *arXiv:2007.04074 [cs.LG]*, 2020.

GILCHRIST, A. *Industry 4.0: the industrial internet of things*. [S.l.]: Springer, 2016.

GOMES, J. et al. A scientific software ecosystem architecture for the livestock domain. *Information and Software Technology*, v. 160, p. 107240, 2023. ISSN 0950-5849. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950584923000940⟩.

HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, v. 212, p. 106622, 2021. ISSN 0950-7051. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950705120307516⟩.

HEVNER, A. et al. Design science research in information systems. *Design research in information systems: theory and practice*, Springer, p. 9–22, 2010.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. *Automated Machine Learning: Methods, Systems, Challenges*. Cham: Springer International Publishing AG, 2019. ISBN 3030053172.

KACHI, F.; BOUANAKA, C. A hybrid model for efficient decision-making in self-adaptive systems. *Information and Software Technology*, v. 153, p. 107063, 2023. ISSN 0950-5849. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950584922001720⟩.

KOREN, O. et al. Automl classifier clustering procedure. *International Journal of Intelligent Systems*, v. 37, n. 7, p. 4214–4232, 2022. Disponível em: ⟨https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22718⟩.

KOTTHOFF, L. et al. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, v. 18, n. 25, p. 1–5, 2017. Disponível em: ⟨http://jmlr.org/papers/v18/16-261.html⟩.

KUFEL, J. et al. What is machine learning, artificial neural networks and deep learning?—examples of practical applications in medicine. *Diagnostics*, v. 13, n. 15, 2023. ISSN 2075-4418. Disponível em: ⟨https://www.mdpi.com/2075-4418/13/15/2582⟩.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. *Machine learning: An artificial intelligence approach*. [S.l.]: Springer Science & Business Media, 2013.

PIMENTEL, M.; FILIPPO, D.; SANTOS, T. M. dos. Design science research: pesquisa científica atrelada ao design de artefatos. *RE@ D-Revista de Educação a Distância e eLearning*, v. 3, n. 1, p. 37–61, 2020.

PROBST, P.; BOULESTEIX, A.-L.; BISCHL, B. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, v. 20, n. 53, p. 1–32, 2019. Disponível em: ⟨http://jmlr.org/papers/v20/18-444.html⟩.

RODRíGUEZ-GRACIA, D. et al. Microservices and machine learning algorithms for adaptive green buildings. *Sustainability*, v. 11, n. 16, 2019. ISSN 2071-1050. Disponível em: ⟨https://www.mdpi.com/2071-1050/11/16/4320⟩.

SILVA, I. E. D. *DT-CREATE – SUITE DE SERVIÇOS PARA ESPECIFICAÇÃO DE DIGITAL TWINS NA INDÚSTRIA 5.0*. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, INSTI- TUTO DE CIÊNCIAS EXATAS, Juiz de Fora, 2024. Disponível em: ⟨https://repositorio.ufjf.br/jspui/handle/ufjf/16844⟩.

VALENTE, M. T. D. O. et al. Engenharia de software moderna: princípios e práticas para desenvolvimento de software com produtividade. Universidade Federal de Minas Gerais, 2020.

VENABLE, J.; PRIES-HEJE, J.; BASKERVILLE, R. Feds: a framework for evaluation in design science research. *European journal of information systems*, Taylor & Francis, v. 25, n. 1, p. 77–89, 2016.

WEYNS, D. *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. 1. ed. Newark: Wiley, 2020. (Wiley - IEEE). ISBN 9781119574941.