Universidade Estadual de Londrina

JENNIFER DO PRADO DA SILVA

PROPOSTA DE UMA ARQUITETURA DE GERÊNCIA DE DADOS PARA TAREFAS DE APRENDIZADO DE MÁQUINA E ANÁLISE DE DADOS NA AGRICULTURA

LONDRINA
2023

JENNIFER DO PRADO DA SILVA

PROPOSTA DE UMA ARQUITETURA DE GERÊNCIA DE DADOS PARA TAREFAS DE APRENDIZADO DE MÁQUINA E ANÁLISE DE DADOS NA AGRICULTURA

Versão Preliminar de Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Daniel dos Santos Kaster

LONDRINA

2023

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

JENNIFER DO PRADO DA SILVA

PROPOSTA DE UMA ARQUITETURA DE GERÊNCIA DE DADOS PARA TAREFAS DE APRENDIZADO DE MÁQUINA E ANÁLISE DE DADOS NA AGRICULTURA

Versão Preliminar de Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Daniel dos Santos Kaster
Universidade Estadual de Londrina

Prof. Dr. Segundo Membro da Banca
Universidade/Instituição do Segundo Membro da Banca – Sigla instituição

---

Prof. Dr. Terceiro Membro da Banca
Universidade/Instituição do Terceiro Membro da Banca – Sigla instituição

---

Prof. Ms. Quarto Membro da Banca
Universidade/Instituição do Quarto Membro da Banca – Sigla instituição

Londrina, 11 de dezembro de 2023.

Este trabalho é dedicado aos meus pais e a minha irmã, que apesar de enfrentarem suas próprias lutas, nunca deixaram de me ajudar com as minhas.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz[1] e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com LaTeX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação[2] da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*[3] e aos novos voluntários do grupo *abnTeX2*[4] que contribuíram e que ainda contribuirão para a evolução do abnTeX2.

[1] Os nomes dos integrantes do primeiro projeto abnTeX foram extraídos de <http://codigolivre.org.br/projects/abntex/>

[2] <http://www.cpai.unb.br/>

[3] <http://groups.google.com/group/latex-br>

[4] <http://groups.google.com/group/abntex2> e <http://abntex2.googlecode.com/>

"A ciência de hoje é a tecnologia de amanhã.
(Edward Teller)

SILVA, J. P.. **PROPOSTA DE UMA ARQUITETURA DE GERÊNCIA DE DADOS PARA TAREFAS DE APRENDIZADO DE MÁQUINA E ANÁLISE DE DADOS NA AGRICULTURA**. 2023. 47f. Trabalho de Conclusão de Curso – Versão Preliminar (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

É possível observar que atualmente a Inteligência Artificial está ganhando força e rapidamente se tornando o pilar da inovação, com capacidade de identificar padrões, resolver problemas, realizar previsões para automatizar tarefas e proporcionar uma compreensão mais abrangente em relação a abundância de dados disponíveis. Na construção de sistemas de Aprendizado de Máquina, é preciso realizar coleta, transformação e organização dos dados antes de induzir os modelos de IA, sendo estes dados de diferentes tipos, fontes e escalas. Visando esta etapa, o objetivo deste projeto é apresentar uma arquitetura geral de ferramentas para coleta, limpeza e pré-processamento dos dados, filtrando dados qualificados e representativos para tarefas e problemas de Aprendizado de Máquina com aplicações voltadas a área da agricultura. Espera-se que os resultados apresentem uma arquitetura com ferramentas de código aberto integradas, assim como casos de uso, capazes de implementar *pipelines* de situações reais relacionadas a tarefas de suporte ao Aprendizado de Máquina na agricultura.

Palavras-chave: Integração de feramentas. Gerência de dados. Agricultura.

SILVA, J. P.. **PROPOSAL FOR A GENERAL DATA MANAGEMENT ARCHITECTURE FOR MACHINE LEARNING AND DATA ANALYSIS TASKS WITH APPLICATIONS IN AGRICULTURE**. 2023. 47p. Final Project – Draft Version (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

It is possible to observe that Artificial Intelligences (AIs) are currently gaining strength and quickly becoming the pillar of innovation, with the ability to identify patterns, solve problems, perform solutions to automate tasks and provide a more comprehensive understanding of the abundance of data available. In the construction of Machine Learning (ML) systems, it is necessary to collect, transform and organize the data before inducing the AI models, these data being of different types, sources and scales. Aiming at this stage, the objective of this project is to present a general architecture of tools for data collection, cleaning and pre-processing, filtering constructed and representative data for ML tasks and problems with applications outside the area of agriculture. It is expected that the results presented are an architecture with integrated open source tools, as well as use cases, capable of implementing pipelines of real situations related to tasks supporting Machine Learning in agriculture.

Keywords: Tool integration. Data management. Agriculture.

LISTA DE ILUSTRAÇÕES

Figura 1 – Funcionalidade do Apache Sqoop 18
Figura 2 – Arquitetura do Flume 19
Figura 3 – Arquitetura do Kafka 20
Figura 4 – Arquitetura do NiFi 21
Figura 5 – Exemplo de um Gráfico Acíclico Direcionado 23
Figura 6 – Exemplo de um Gráfico Cíclico Direcionado 24
Figura 7 – Gráfico percentual do tempo gasto por cientistas de dados 27
Figura 8 – Estrutura Board Support Package (BSP) em um SO Ångström 30
Figura 9 – Arquitetura lógica de alto nível BOP. 31
Figura 10 – Processo de enriquecimento no BOP. 32
Figura 11 – Visualização gráfica de um PA Digital DAG no Airflow. 35
Figura 12 – DAG criado dinamicamente com base nos dois arquivos de configuração. 36
Figura 13 – Tarefas reais a serem executadas organizadas em Grupos de Tarefas. . 36
Figura 14 – Exemplo de uma aplicação simples do Nansat. 37

LISTA DE TABELAS

Tabela 1 – Indicadores de Performance de Ferramentas de Ingestão de Dados . . . 22

SUMÁRIO

1 INTRODUÇÃO 13

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA 15
2.1 Integração entre sistemas 15
2.2 Ingestão de Dados 16
2.2.1 Tipos de Ingestão de Dados 16
2.2.1.1 Ingestão de dados em lote 17
2.2.1.2 Ingestão de dados em tempo real 17
2.2.2 Ferramentas de Ingestão de Dados 17
2.2.2.1 Apache Sqoop 17
2.2.2.2 Apache Flume 18
2.2.2.3 Apache Kafka 19
2.2.2.4 Apache NIFI 20
2.3 Gestão de fluxo de trabalho 22
2.3.1 Tipos de fluxo de trabalho 23
2.3.1.1 Gráfico Acíclico Direcionado 23
2.3.1.2 Gráfico Cíclico Direcionado 23
2.3.2 Ferramentas de gerenciamento de fluxo de trabalho 24
2.3.2.1 Apache Airflow 24
2.3.2.2 Kedro 25
2.3.2.3 Luigi 25
2.4 Tarefas de preparação de dados para aprendizado de máquina ponta a ponta 26
2.5 Engenharia de *Features* 27

3 TRABALHOS RELACIONADOS 29
3.1 Ingestão de dados 29
3.1.1 ISOBlue HD 29
3.1.2 Billion Object Platform (BOP) 31
3.2 Integração 33
3.2.1 Polly 33
3.3 Gerenciamento de fluxo de trabalho 34
3.3.1 Construindo fluxos de trabalho de agregação de metadados usando Apache Airflow 34
3.3.2 Estrutura de processamento de dados do Sentinel-2 35
3.4 Geoprocessamento 36

3.4.1 Nansat 36
3.4.2 Explorando a visualização de dados geoespaciais baseada em Python . 38

4 ESTUDO DE CASO 39

5 CONCLUSÃO 40

REFERÊNCIAS 41

1 INTRODUÇÃO

Em 2017, o site de publicidade inglês de notícias e assuntos internacionais, *The Economist*, afirmou que o recurso mais valioso do mundo não é mais o petróleo, mas os dados [1]. Diante desta afirmação, é comum observar que empresas de todo porte, instituições e organizações vem lidando com uma quantidade de dados significativamente grande, de todo tipo e de uma variedade de fontes, tornando cada vez mais evidente que a transformação de dados é frequentemente necessária [2].

É comum observar inúmeras aplicações do aprendizado de máquina em diversas áreas do conhecimento, como por exemplo na área da saúde, tecnológica, biológica e agrária. No caso desta última, inúmeras mudanças e inovações tecnológicas ocorreram nos anos atuais, com isso é possível observar que a inteligência artificial (IA) e o aprendizado de máquina são cada vez mais usados para previsão na agricultura [3] [4], uma das principais áreas em que os grandes volumes de dados e tipos de dados complexos são predominantes.

Em uma de suas matérias, no ano de 2019, a empresa ucraniana de TI, *Sciforce*, afirmou que o aprendizado de máquina está em toda parte durante todo o ciclo de cultivo e colheita [5]. Pode-se destacar os estudos realizados por pesquisadores que recorrem aos métodos de aprendizado de máquina e tecnologia de visão computacional para a identificação de doenças e pragas agrícolas [6]. Essas ideias e estudos enfocam que a agricultura pode sim se beneficiar do aprendizado de máquina para um melhor desempenho e resultados em suas atividades.

Em sua maioria, as tarefas de aprendizado de máquina direcionadas a área agrícola envolvem uma grande quantidade de dados. Com isso, a agricultura passa a ser cada vez mais dependente de soluções baseadas em dados e informações, para geração de diferenciais e elaboração de estratégias de impulsionamento do setor [7]. Isso acontece devido ao crescente volume e variedades de dados no campo. As técnicas convencionais de processamento de dados são incapazes de atender às demandas cada vez maiores na nova era da agricultura inteligente, o que é um importante obstáculo para extrair informações valiosas dos dados de campo [3].

Trabalhar com dados agrícolas requer levar em consideração que alcançar alta precisão e interpretabilidade é um desafio [8]. Esse grande volume de dados exigirá abordagens inteligentes a fim de evitar que se tornem apenas um enorme acúmulo [9]. Tal afirmação se direciona aos inúmeros tipos de dados existentes no meio agrícola que, se não preparados, podem prejudicar na previsão de uma dada variável dependente, na acurácia e até mesmo na otimização do processo, ao invés de oferecer *insights* que contribuam no acompanhamento desde a produção até o consumidor final.

Visando este panorama no aprendizado de máquina, muitas empresas se adaptaram e se especializaram para oferecer ferramentas úteis, no quesito de preparação dos dados, capazes de solucionar os problemas existentes no conjunto de dados, como por exemplo o *Azure Databricks* da empresa *Microsoft*, uma plataforma unificada que disponibiliza recursos para ingestão, preparação, análise e monitoramento de dados [10], dividindo o cenário com outras companhias especialistas, assim como o serviço *Oracle Cloud Infrastructure Data Integration*, da empresa *Oracle*, uma interface gerenciada, sem servidor e nativa da nuvem, responsável por extrair, carregar, transformar, limpar e remodelar dados [11], a interface *Amazon SageMaker Data Wrangler*, da empresa *Amazon*, a qual proporciona recursos para limpeza, exploração, visualização e processamento de dados tabulares e de imagem em grande escala [12] e outros sistemas disponíveis das diversas empresas no mercado da tecnologia. Estes serviços oferecem uma gama de recursos favoráveis para a preparação de dados, mas são plataformas limitadas, ja que cobram pelos recursos consumidos ou entregam pacotes de recursos por um valor preestabelecido. Além deste impasse relacionado a monetização, pode ser destacado também a limitação que o cliente tem em relação aos recursos próprios da empresa fornecedora do serviço, privando seus usuários e fazendo com que estes sejam incapazes de aplicar ao seu conjunto de dados, processos de preparação de dados provenientes de companhias concorrentes.

Neste contexto, este projeto propõe apresentar uma arquitetura geral para o processamento de grandes volumes de dados, com ferramentas integradas de código aberto que sejam compatíveis entre as demais, próprias para a coleta e limpeza de dados, bem como extrair informações relevantes destes, de preparar e gerenciar os diversos tipos de dados agro meteorológicos existentes e integrar as ferramentas selecionadas de maneira estratégica, de modo que seja possível desfrutar o máximo de suas funcionalidades e manter informações válidas, relevantes e precisas para aplicações em tarefas e soluções agrícolas baseadas em Aprendizado de Máquina.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

2.1 Integração entre sistemas

A definição de integração de sistemas pode ser descrita como a capacidade de realizar conexões de dados, aplicativos, *APIs* e dispositivos [13], ou seja, diferentes ferramentas conversarem entre si e trocarem dados, reunindo ou incorporando partes em um todo, de maneira que a integração desses elementos contribuam para melhorar a eficiência, intensificar a produtividade, criar rotinas mais inteligentes com comunicação eficiente, e a garantir agilidade dos processos [14].

Para formar uma integração entre sistemas, alguns fatores devem ser considerados neste processo, como definir objetivos, mapear os processos internos, e definir quais integrações são necessárias. Para atuar na formação deste processo, é essencial contar com o suporte de plataformas tecnológicas [15].

A integração entre sistemas mantém a sincronização entre os sistemas sempre que ocorrer modificações de dados ou eventos, vinculando sistemas a nível funcional, permite criar uma integração orientada a eventos e mensagens de forma dinâmica e extremamente adaptáveis com transferências de dados em alta velocidade. As integrações apresentam ações controladas por eventos, que ocorrem quando um acionador, ou evento, dispara um procedimento ou uma solicitação. Também conta com integrações de *APIs*, mapeamento de dados entre os sistemas para definir como estes dados serão transferidos, facilitando a exportação, agrupamento e análise destas informações [16] [17].

Há inúmeras pesquisas relacionadas a integração de sistemas em diversas áreas, como um exemplo têm-se o livro *'Handbook of Medical Image Computing and Computer Assisted Intervention'*, que apresenta processos, ferramentas e melhores práticas para desenvolvimento, teste e manutenção de software baseado em componentes, na área de integração de sistemas de intervenção assistida por computador (CAI, pela sigla em inglês), discute sobre as diversas considerações de design, opções para estruturas de aplicativos e *middleware* para lidar com a comunicação entre os componentes, como o *Robot Operating System(ROS)*, para sistemas CAI que integram robôs e o *OpenIGTLink*, para lidar com integração de imagens médicas [18]

Da mesma forma, existem diversos softwares e plataformas especializadas que realizam integrações entre sistemas, como um exemplo, é válido mencionar a plataforma *TensorFlow Extended(TFX)*, baseada em *TensorFlow* e implementada no *Google*, própria para implantação e gerenciamento de *pipelines* de produção de aprendizado de máquina [19]. Esta realiza a integração de componentes, como, por exemplo, para gerar modelos

baseados em dados de treinamento, módulos para analisar e validar dados e modelos e infraestrutura para servir modelos em produção, tudo em uma unica plataforma, padroniza os componentes, simplifica a configuração da plataforma e reduz o tempo de produção da ordem de meses para semanas, ao mesmo tempo que proporciona estabilidade da plataforma, minimizando interrupções [20]. Também é importante citar a plataforma de integração *IBM Cloud Pak® for Integration*, que apresenta recursos para aumentar a velocidade e a qualidade do sistema, com gerenciamento de *API*, integração de aplicativos, *streaming* de eventos e outras funcionalidades, além de possuir conectores inteligentes préconstruídos e recursos de automação, como integração de dados com tecnologia de IA, processamento de linguagem natural(PNL, pela sigla em inglês) e ferramentas de baixo código [17].

2.2 Ingestão de Dados

Nos últimos anos, diversos sistemas, dispositivos e maquinários apresentaram um crescimento exponencial de informações geradas, em diferentes contextos da sociedade, após passarem por inúmeros avanços tecnológicos. Isso possibilitou a produção em massa, em um tempo reduzido, de dados em uma escala sem precedentes, beneficiando análises minuciosas de dados e tomada de decisões baseadas em dados. Dentre essas informações geradas em grande escala, há diversos conjuntos de dados definidos como dados multiestruturados. Como exemplo desse tipo de dado, têm-se os textos em larga escala, imagens, vídeos e áudios [21]. Dessa forma, os dados precisam ser organizados para haver a aplicação plena e precisa destes, em diferentes destinos, além de serem facilmente acessíveis e compreensíveis para o usuário[22].

Para solucionar este problema, diversas empresas e organizações começaram a aplicar a ingestão de dados. Na ingestão de dados é realizado a coleta e o transporte de dados de diversas fontes para um meio de armazenamento centralizado, onde estes podem ser acessados pelos usuários, compartilhados para outros sistemas e plataformas e analisados ao fim do processo. Esta camada de ingestão de dados é etapa principal de qualquer arquitetura analítica e muitos sistemas de relatórios e análises dependem de dados consistentes e acessíveis [23][24].

2.2.1 Tipos de Ingestão de Dados

Atualmente, há dois métodos de ingestão de dados: Ingestão de Dados em Lote e Ingestão de Dados em tempo real. Os requisitos mínimos a serem atendidos e as restrições que devem ser respeitadas de cada negócio vão definir qual estrutura da etapa de ingestão de dados será aplicada em um projeto específico.

2.2.1.1 Ingestão de dados em lote

A forma mais implantada de ingestão de dados é o processamento em lote. Nesse contexto, a camada de ingestão coleta e reúne dados de diversas fontes em intervalos regulares e os envia posteriormente ao sistema de destino. Esses conjuntos de dados, podem ser processados com base em uma ordem lógica qualquer, ao ser realizado a ativação de condições específicas pré-determinadas ou em um agendamento simples estabelecido pelo usuário. Quando a coleta de dados em tempo real não é crucial para sistema em questão, a opção utilizada é o processamento em lote, visto que é mais simples e acessível de ser implementada do que a ingestão em tempo real [25][26].

2.2.1.2 Ingestão de dados em tempo real

A ingestão de dados em tempo real, também conhecida como processamento de fluxo ou *streaming*, difere da ingestão citada anteriormente por não requerer qualquer fase de agrupamento. Os conjuntos de dados alvos, são coletados da origem, manipulados e carregados imediatamente, após sua criação ou reconhecimento pela camada de ingestão de dados. Embora mais custoso, visto que é requerido o monitoramento constante das fontes de dados e a coleta imediata de novas informações, essa forma de ingestão é apropriada para análises que demandam dados continuamente atualizados, em um intervalo de tempo relativamente curto ou até mesmo em tempo real.

Muitas vezes, os dados são adquiridos sequencialmente, como um fluxo infinito e crescente. Esses dados de fluxo em tempo real precisam de processamento sequencial, em que a fonte de dados é particionada ao longo dos limites temporais em segmentos ou janelas finitas. Como exemplo, têm-se os dados do mercado de ações, sensores ou feeds do *Twitter*. Ao invés de esperar pela coleta total dos dados, em intervalos longos periódicos, a análise de streaming possibilita a identificação de padrões e a tomada de decisões com base nesses dados coletados, à medida que os dados começam a chegar. Quando os dados a serem ingeridos não são estacionários e apresentam padrões diferentes ao longo do tempo, as análises em tempo real se adaptam. Em casos de dados brutos em grande escala, onde o armazenamento destes torna-se questionável, a análise de streaming permite a persistência de apenas representações menores e mais direcionadas [27][28].

2.2.2 Ferramentas de Ingestão de Dados

2.2.2.1 Apache Sqoop

O *Apache Sqoop* é uma ferramenta projetada para facilitar a transferência eficiente de grandes volumes de dados entre o Hadoop e bancos de dados relacionais ou *mainframes*. Como ilustrado na figura 1, o *Sqoop* pode extrair dados de diversos sistemas de gerenciamento de banco de dados relacional (RDBMS), como *Oracle*, *MySQL* ou *mainframes*, e

transferi-los para o sistema de arquivos distribuídos *Hadoop (HDFS)*. Posteriormente, os dados podem ser processados usando *Hadoop MapReduce*, e os resultados exportados de volta para o RDBMS de origem [29].

O *Sqoop* utiliza conectores JDBC para estabelecer a conexão com os RDBMS. Para isso, requer que o *Java* esteja instalado e que o jar do *driver* JDBC esteja no caminho de classe do tempo de execução e depende do esquema do banco de dados relacional para interpretar os dados importados. Além dessas particularidades, ele também suporta *HSQLDB* (versão 1.8.0+), *MySQL* (5.0+), *Oracle* (10.2.0) e *PostgreSQL* (8.3+) e ainda pode ser usado com outros bancos de dados relacionais, como banco de dados *IBM DB2* e versões. Ao aproveitar o poder do *MapReduce* para processamento paralelo, o *Sqoop* gera vários arquivos no *HDFS* durante o processo de importação, os quais podem ser formatados como arquivos de texto delimitados, binários *Avro* ou arquivos de sequência, contendo os dados importados [30][31].

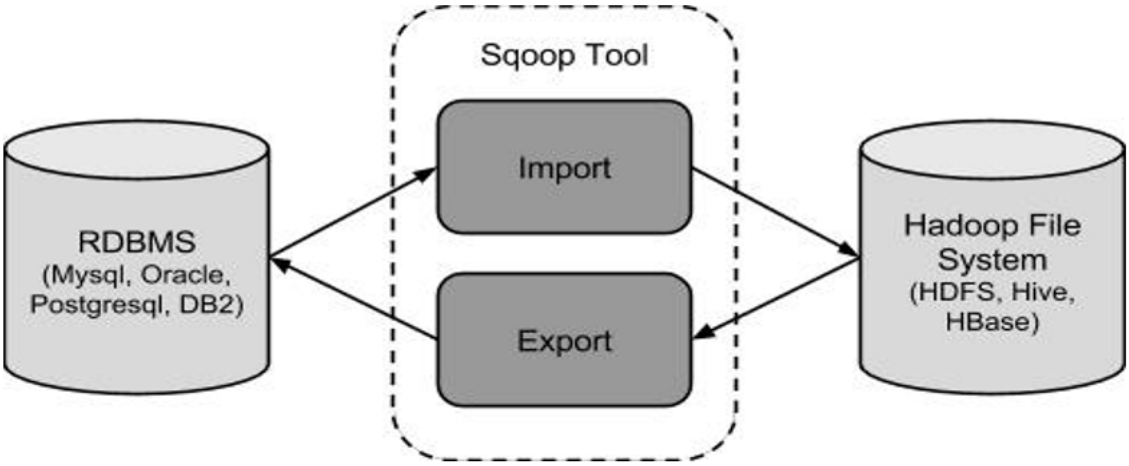

Figura 1 – Funcionalidade do Apache Sqoop. (Adaptado: 32)

2.2.2.2 Apache Flume

O *Apache Flume* é um serviço distribuído confiável, acessível e eficiente e foi projetado para importar, reunir e processar grandes volumes de dados, fornecendo uma plataforma para análise de dados em tempo real. Ele apresenta alguns desafios na tolerância a falhas com consistência precisa e é principalmente aplicado em modelo de dados para análise em tempo real. Desempenha responsabilidades essenciais no refinamento e visualização de dados. O *Flume* oferece uma estrutura para a coleta e análise de dados provenientes de uma rede de sensores, garantindo escalabilidade e alto desempenho no *Hadoop Distributed File System (HDFS)*[33].

O fluxo de dados no *Flume* é comparável a um *pipeline* responsável por capturar dados da origem e entregá-los ao destino. Na arquitetura *Flume*, ilustrada na figura 2, os dados passam por transformações da origem até o destino por meio do agente *Flume*. Esse agente, executado como um processo JVM, hospeda os componentes essenciais durante todo o percurso dos dados, incluindo *Source*, *Channel* e *Sink*. O *Source* recebe dados dos geradores associados e os encaminha para o *Channel*, que atua como uma ponte entre o *Sink* e o *Source*. O *Sink* é a entidade encarregada de enviar os dados ao destino [34].

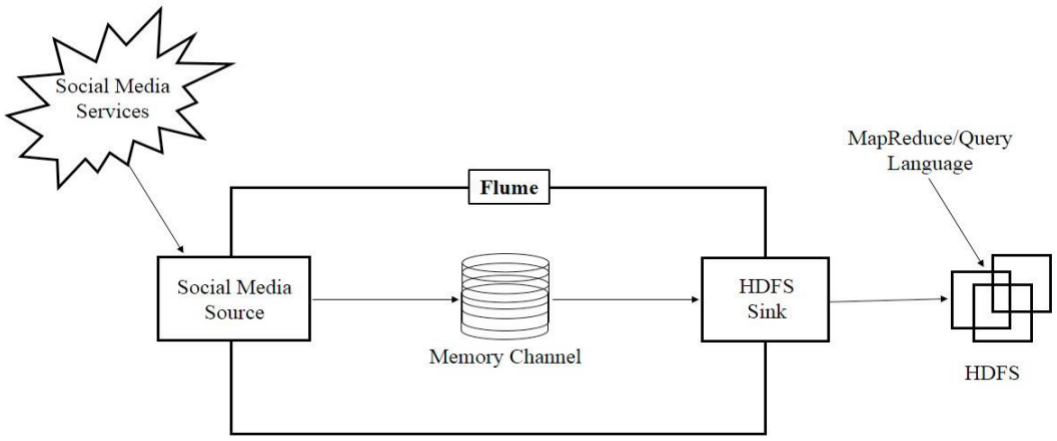

Figura 2 – Arquitetura do Flume. (Fonte: 35)

2.2.2.3 Apache Kafka

É um sistema *open source* que trabalha de forma distribuída para streaming de dados, ele é capaz de processar grandes volumes de dados em uma latência excepcionalmente baixa, visto que todo o processo ocorre na memória para evitar latência de acesso dos discos rígidos. Dentre suas funcionalidades, vale citar as seguintes: transferir uma informação de um sistema para o outro, armazenar essa informação para consultar posteriormente e transformar essa informação e transportá-la para outro sistema [36].

O *Kafka* possibilita trabalhos no formato de filas, assim como o *HabbitMQ* e *Amazon Simple Queue Service*. Ele possui mais recursos além de mensageria, que se trata do envio de dados de um lado para outro. O *Kafka* também apresenta esquema de escalabilidade, permitindo que os dados sejam armazenados por tempo indeterminado ou em um limite de tempo, pré-determinado pelo usuário, em que esses dados serão persistidos. Além dessas características, esta ferramenta dispõe de grande disponibilidade e tolerância a falhas, visto que é estruturado para manter cópias dessas informações [37][38].

Em sua arquitetura, ilustrada na figura 3, o *Kafka* apresenta três elementos principais: o *broker*, o consumidor e o produtor. Mesmo que o consumidor e o produtor sejam desenvolvidos em linguagens de programação distintas, o sistema opera de maneira efi-

ciente, conectando diversas plataformas. O *broker* desempenha a função de servidor no *Kafka* e é crucial para assegurar a tolerância a falhas, sendo este o recurso mais importante do *Kafka*. O produtor envia mensagens ao consumidor por meio do *broker*, que atua como um canal para a diferenciação das mensagens. O *Kafka* apresenta um canal de dados em tempo real de alto desempenho e para fins de gerenciamento e coordenação, o *broker Kafka* utiliza o *ZooKeeper* para notificar o produtor e o consumidor sobre a falha ou até mesmo inclusão de qualquer *broker* [39][40].

O *Apache Kafka* possui uma configuração que permite o usuário conectar sistemas externos nele, por meio de seus conectores. Logo após a realização de uma conexão, o *Kafka* inicia automaticamente a leitura nos conectores, se houverem. Esse recurso ainda permite a seleção de informações hospedadas no *Kafka* e o envio destas para outros sistemas de forma totalmente integrada, utilizando poucas linhas de código. O *Kafka* é aplicável para grandes e pequenas estruturas, neste ultimo caso, conforme o desenvolvimento da estrutura, o *Kafka* consegue auxiliar no crescimento de forma que seja possível escalar infinitamente. Algumas empresas implantaram o *Kafka* em seus serviços de dados, como a *Netflix*, *LinkedIn*, *Uber*, *Twitter*, *Dropbox*, *Spotify*, *Paypal* e *Banks* [32][41].

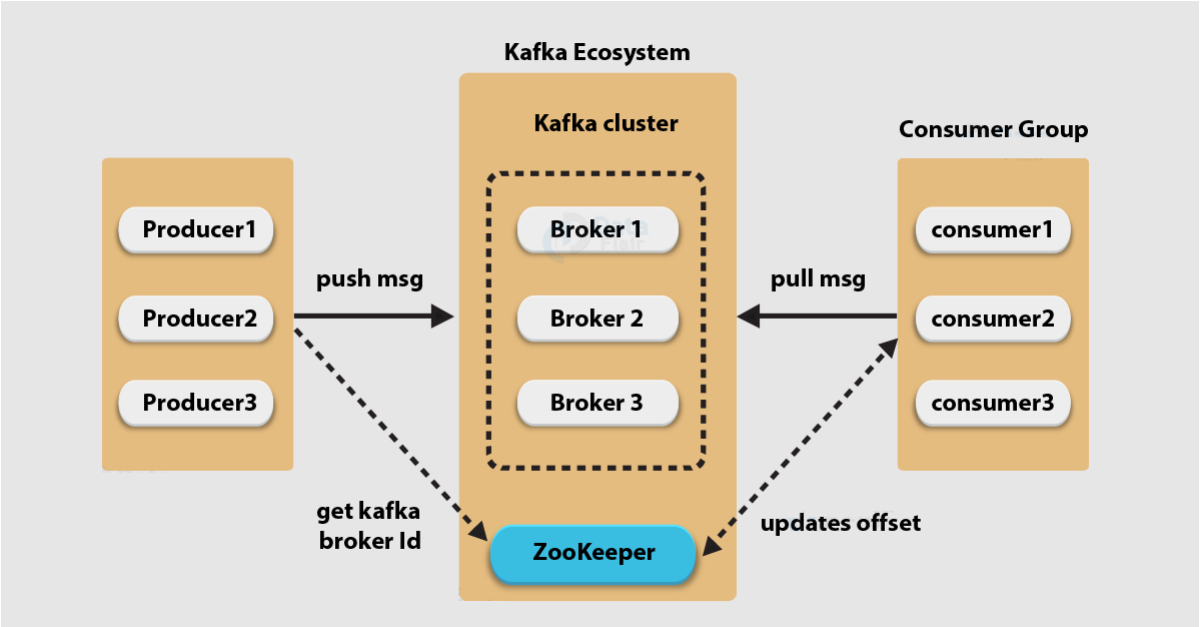

Figura 3 – Arquitetura do Kafka. (Fonte: 42)

2.2.2.4 Apache NIFI

O *NIFI* é um sistema de fluxo de dados projetado para coletar, transformar, processar e rotear dados. Sua arquitetura segue o conceito de programação baseada em fluxo, visando automatizar e gerenciar o fluxo de dados entre sistemas [43].

Desenvolvido em *Java*, o *NIFI* opera em uma *Máquina Virtual Java (JVM)* em um sistema operacional *host*, também é capaz de ser executado em um *cluster*, cada nó no *cluster NIFI* completa as mesmas tarefas, mas interage com conjunto diferente de dados. O *cluster* é gerenciado pelo coordenador do *cluster*, eleito pelo *Zookeeper* [44].

A arquitetura do *NIFI*, apresentada na figura 4, é composta por diversos componentes, incluindo o Servidor *Web*, responsável por hospedar comandos baseados em *HTTP* e permitir que os usuários acessem o *NIFI* por meio de uma interface *web*. Outros componentes essenciais são o Controlador de Fluxo, encarregado de fornecer e agendar *threads* para execução, o Repositório *FlowFile*, onde o *NIFI* registra as atualizações de *status* dos arquivos de fluxo, o Repositório de Conteúdo, que armazena o conteúdo dos *flowfiles*, e o Repositório de Proveniência, que contém dados relacionados aos eventos de proveniência[45].

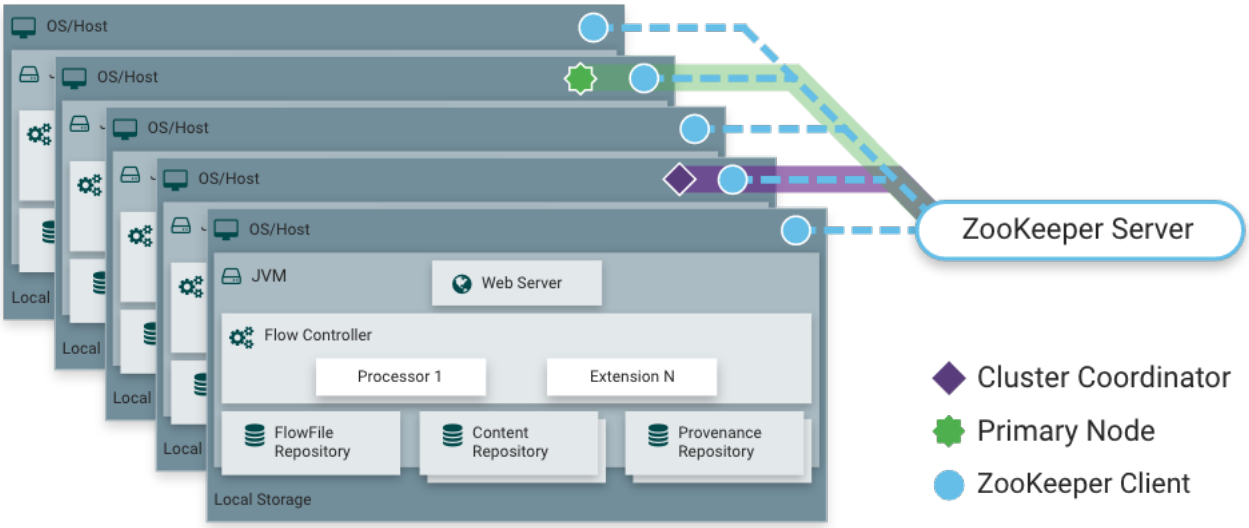

Figura 4 – Arquitetura do NiFi. (Fonte: 45)

Na tabela 1, comparações foram realizadas entre as ferramentas de ingestão de dados mencionadas, baseadas em diversos indicadores, incluindo tipo de carregamento, arquitetura e o tipo de dados que podem ser ingeridos. Todas as ferramentas descritas são de código aberto e escritas em *Java*. O *Sqoop* destaca-se como uma escolha sólida para a ingestão de dados em lote de sistemas de gerenciamento de banco de dados relacionais (RDBMS), suportando compactação de dados, assim como as demais ferramentas e oferecendo um modo bidirecional para interação com o HDFS, permitindo a importação e exportação de dados. O *NIFI* é uma opção recomendada para estabelecer fluxos de dados entre sistemas diversos. Ele é capaz de realizar a ingestão tanto em lotes quanto em fluxo, proporcionando a priorização de eventos e o carregamento dos dados de forma orientada e não orientada a eventos. Quanto ao *Kafka* e o *Flume*, ambos lidam com fluxos de dados

e armazenam mensagens por mais tempo em relação as demais ferramentas, mas o *Kafka* é especialmente usado para mensagens. Destaca-se por sua alta escalabilidade, permitindo a fácil adição de diferentes números de consumidores para atender às necessidades específicas do usuário.

Tabela 1 – Indicadores de Performance de Ferramentas de Ingestão de dados. (Adaptado de: [32][37])

Indicador	**Flume**	**NiFi**	**Kafka**	**Sqoop**
Primeira linguagem	Java	Java	Java	Java
Licença	Código aberto	Código aberto	Código aberto	Código aberto
Natureza básica	Para streaming de dados fontes gerados continuamente	Para a criação de fluxo de dados entre diferentes sistemas.	Para mensagens de Streaming de dados	Para RDBMS que possui JDBC(Java EE Database Connectivity) como o oracle
Tipo de dados	Dados em tempo real	Dados em tempo real e em lote	Dados em tempo real	Dados em lote
Escalabilidade	Médio	Médio	Alto	Médio
Tipo de carregamento	Orientado a eventos	Ambos (evento e não-evento)	Orientado a eventos	Não orientado a eventos
Arquitetura	Baseado em agente	Baseado em fluxo	Baseado na topologia de processo	Baseado em conector
Priorização de evento	Suportado	Suportado pelo conceito (processador de coletor de failover)	Programável	Não possui
Duração de mensagens	Alto	Baixo	Alto	Baixo

2.3 Gestão de fluxo de trabalho

Os sistemas de gerenciamento de fluxo de trabalho têm despertado considerável interesse nos últimos anos, devido à sua capacidade de integrar aplicações distribuídas e heterogêneas em diferentes ambientes de processamento. Mesmo apresentando algumas limitações, as ferramentas existentes de gerenciamento de fluxo de trabalho possuem um processamento rápido e estruturado de tarefas, eficiência de implantação para diversas aplicações e vários recursos disponíveis para o usuário orquestrar seus trabalhos de forma organizada [46] [47].

Um gerenciador de fluxo de trabalho se trata de um sistema responsável por administrar processos e tarefas repetitivas, que precisam ocorrer em uma ordem específica. Ele pode coordenar desde uma série simples de tarefas individuais até um processo de negócios, considerado mais complexo, por apresentar vários fluxos de trabalho, sistemas de informação, dados e padrões de atividade. Um fluxo de trabalho é geralmente visualizado

em um diagrama e possui diversas formas, isso devido à dificuldade de planejamento e repetitividade que os processos podem apresentar [48].

A maioria dos sistemas de gerenciamento de fluxo de trabalho disponibilizam recursos para criar linhas de processamento estruturados como um Gráfico Acíclico Direcionado, conhecido como *DAG* do inglês *Directed Acyclic Graphs*, ou como um Gráfico Cíclico Direcionado, conhecido como *DCG* do inglês *Directed Cyclic Graphs*.

2.3.1 Tipos de fluxo de trabalho

2.3.1.1 Gráfico Acíclico Direcionado

Nos fluxos de trabalho baseados em *DAGs*, ilustrado na figura 5, os nós do grafo representam as tarefas a serem executadas e as arestas indicam as dependências entre essas tarefas. Existem dois tipos principais de dependências a serem consideradas: dependências de dados, em que a saída de uma tarefa serve como entrada para as tarefas subsequentes, e dependências de controle, em que certas tarefas precisam ser concluídas antes do início de uma tarefa individual ou de um conjunto de tarefas. O *DAG* pode ser empregado para representar um conjunto de programas, nos quais a entrada, saída ou execução de um, ou mais programas está condicionada à conclusão bem-sucedida de um ou mais programas específicos [47][49].

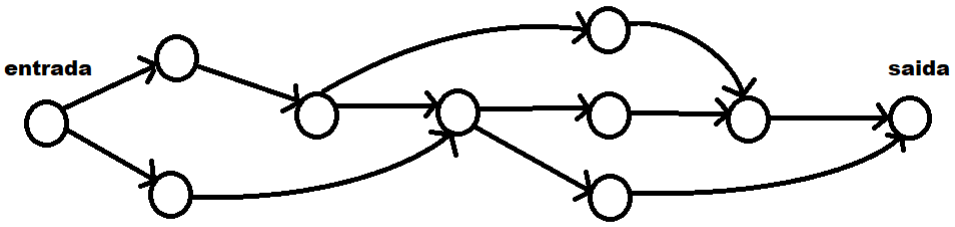

Figura 5 – Exemplo de um Gráfico Acíclico Direcionado.

2.3.1.2 Gráfico Cíclico Direcionado

Os fluxos de trabalho baseados em *DCG*, como o exemplo ilustrado na figura 6, são caracterizados por ciclos que denotam um *loop*, seja implícito ou explícito, ou mecanismos de controle de iteração. Nesse contexto, o gráfico de fluxo de trabalho geralmente descreve uma rede de tarefas, onde os nós representam processos, instâncias de componentes de software ou objetos de controle mais abstratos. Neste tipo de fluxo, os processos podem ser executados uma ou mais vezes, dependendo das condições atendidas [47].

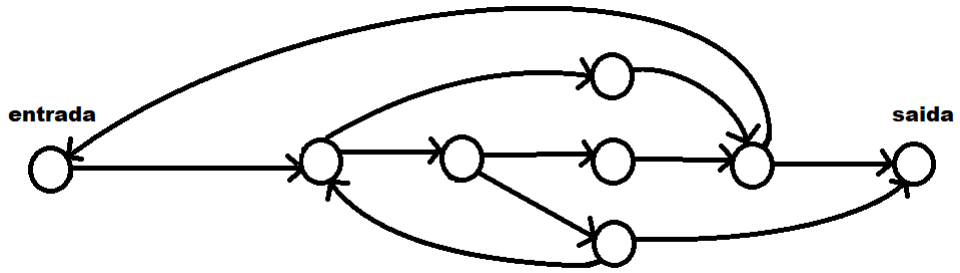

Figura 6 – Exemplo de um Gráfico Cíclico Direcionado.

2.3.2 Ferramentas de gerenciamento de fluxo de trabalho

2.3.2.1 Apache Airflow

Dentre várias ferramentas de gerenciamento de fluxo de trabalho, foi selecionado o *Apache Airflow*, uma ferramenta *Open Source* destinada à criação, monitoramento e agendamento de fluxo de trabalho de forma programática. O *Apache Airflow* é um recurso altamente integrável e foi desenvolvido seguindo alguns princípios importantes para gerenciamento de *workflows* e atribuindo características úteis.

O primeiro princípio é ser uma ferramenta escalável, para isso, conta com uma arquitetura modular, em que apresenta uma fila de mensagens para administrar uma quantidade arbitrária de trabalhadores. O segundo é o dinamismo. O *Apache Airflow* apresenta *pipelines* definidos em *Python*, isso possibilita a geração dinâmica de pipelines, por meio de códigos que os instanciam dinamicamente. Por fim, o terceiro princípio se trata de ser uma ferramenta extensível, permitindo estabelecer facilmente os operadores desejados e estender as bibliotecas a fim de definir e controlar o nível de abstração conforme o ambiente esperado [50].

Em relação às características, o *Apache Airflow* apresenta diversos aspectos importantes, focados em torná-lo um recurso simples e completo. A criação e gerenciamento de *workflows*, é realizada utilizando inteiramente a linguagem *Python*, possibilitando a aplicação de diversas funcionalidades, como, por exemplo, agendamento de pipelines utilizando diferentes formatos de data e hora e aplicação de loops aos fluxos de trabalho visando gerar tarefas dinamicamente. Com isso, o *Python* ajuda a garantir total flexibilidade durante a criação e gerenciamento dos *pipelines* [51][52].

O *Airflow* também conta com uma robusta aplicação *web* que auxilia e facilita no monitoramento, agendamento e gerenciamento das tarefas criadas, proporcionando uma visão completa da situação de cada *pipeline* em execução e dos registros de tarefas concluídas. Além destas características, é importante destacar que o *Apache Airflow* não limita o escopo dos *pipelines*, pode ser utilizado para diversos fins e disponibiliza diversos operadores *plug-and-play* para executar tarefas em vários serviços, como, por exemplo, o

Google Cloud Platform e o *Amazon Web Services*. Essas características permitem que o *Airflow* seja extensível e fácil de se adaptar às novas infraestruturas [53][54].

2.3.2.2 Kedro

Kedro é uma estrutura *Python* de código aberto hospedada pela *Linux Foundation*, foi projetada para facilitar a construção de pipelines de aprendizado de máquina e dados complexos. Ele adota melhores práticas para desenvolver código limpo em ciência de dados, reduzindo o tempo gasto em tarefas de "encanamento". Com recursos como a criação de versões de dados, cálculos incrementais e automação da ordem de execução do *pipeline*, o *Kedro* simplifica o desenvolvimento. Ao integrar o *Kedro-Viz*, ele oferece um plano de desenvolvimento de dados, linhagem de dados, acompanhamento de experimentos de aprendizado de máquina e facilita a comunicação com partes interessadas do negócio [55].

Alguns orquestradores de fluxo de trabalho como o *Airflow*, *Luigi* e *Prefect* se dedicam à execução, agendamento e monitoramento de *pipelines*. Em contrapartida, o *Kedro* tem como foco o processo de criação de *pipelines*. Portanto, se os objetivos do gerenciamento de fluxo de trabalho incluem solucionar problemas de como a tarefa será executada ou como a computação será gerenciada, o *Kedro* pode não ser a solução mais adequada [56][57].

O *kedro* possui um catálogo de dados com uma série de conectores de dados leves usados para salvar e carregar dados em diversos formatos e sistemas de arquivos [58].

2.3.2.3 Luigi

Luigi é um pacote *Python* projetado para facilitar a construção de *pipelines* complexos de trabalhos em lote. Ele lida com resolução de dependências, gerenciamento de fluxo de trabalho, visualização, tratamento de falhas, integração de linha de comando e diversos outros recursos. Os usuários têm a capacidade de especificar as dependências entre tarefas utilizando gráficos acíclicos direcionados (DAGs), assegurando a execução e repetição adequadas das tarefas na ordem correta [59].

Ele proporciona a flexibilidade de criar virtualmente qualquer tarefa e ainda oferece uma caixa de ferramentas contendo modelos de tarefas comuns para simplificar o processo. Essa ferramenta abrange suporte para a execução de *jobs mapreduce* em *Python* no *Hadoop*, assim como *jobs Hive* e *Pig*. Além disso, *Luigi* fornece abstrações de sistema de arquivos para *HDFS* e arquivos locais, garantindo operações atômicas no sistema de arquivos. Essa característica é crucial, pois assegura que o *pipeline* de dados não fique preso em um estado que contenha dados parciais [60].

O servidor *Luigi* também apresenta uma interface *web*, permitindo que o usuário

pesquise e filtre todas as tarefas de maneira conveniente. Tudo no *Luigi* é implementado em *Python*. Ao contrário de configurações em XML ou arquivos de dados externos similares, o gráfico de dependência é especificado diretamente em *Python*. Essa abordagem simplifica a construção de gráficos de dependência complexos de tarefas, nos quais as dependências podem envolver álgebra de datas ou referências recursivas a outras versões da mesma tarefa [61][62].

2.4 Tarefas de preparação de dados para aprendizado de máquina ponta a ponta

O aprendizado de máquina é uma ramificação da inteligência artificial e da ciência da computação, e tem como foco, o uso de dados, algoritmos e modelos estatísticos para simular o modo como os humanos aprendem. Para isso, essa ramificação aprimora gradualmente sua precisão de forma autônoma, utilizando redes neurais e aprendizado profundo sem a necessidade de ser programado explicitamente, confiando em padrões e inferências. Com isso, quanto maior for a alimentação de dados no sistema, mais precisos serão os resultados [63] [64].

O conceito de aprendizado de máquina é muito importante para se ter ideia de como funciona este subconjunto da inteligência artificial, mas para compreender todo o processo e funcionamento de um projeto que abrange a concepção deste subconjunto é essencial estar inteirado sobre as etapas do aprendizado de máquina ponta a ponta. Este é um processo, em que uma máquina aprende um mapeamento entre uma série de entradas (X) para algumas saídas conhecidas (y), sem ser explicitamente programada [65]. Pode ser definido em três etapas. A primeira, foca em possuir ampla compreensão dos dados, meio de coleta e métodos para limpeza de dados. Em seguida, a segunda etapa, destina-se em selecionar e implementar o modelo. Por fim, a terceira etapa, é a definição dos parâmetros do modelo e ajuste de dados [66][67].

No aprendizado de máquina ponta a ponta, como ilustrado na figura 1, os cientistas de dados gastam 19% de seu tempo na coleta de conjuntos de dados [68], uma das tarefas mais longas na construção de um modelo. Isso acontece, pois na execução deste processo há dois fatores fundamentais a serem considerados, a qualidade e quantidade. Contando com um conjunto de dados grande o suficiente é possível ter mais análises significativas e dispor de dados de alta qualidade garante que as informações sejam relevantes o bastante para o caso de uso em questão e proporcionam modelos resultantes com altas taxas de precisão [69].

Já a limpeza e organização dos dados é responsável por cerca de 60% do trabalho dos cientistas de dados [68] e com isso a etapa mais demorada e importante na construção de um modelo. Neste processo, são realizadas diversas verificações, como tratamento de

valores ausentes com dados inferidos a partir dos dados existentes, modificação de variáveis categóricas, identificação de valores discrepantes, expandir *features* existentes, limpeza e várias outras transformações dos dados de treinamento para que estejam prontos para análise de dados e otimização de modelo [69] [70].

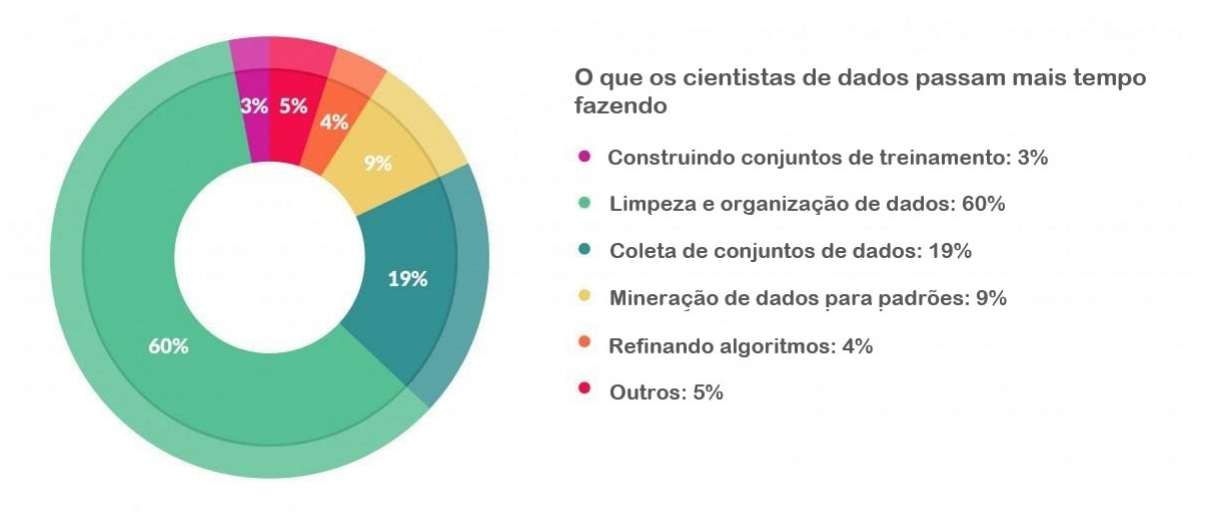

Figura 7 – Gráfico percentual do tempo gasto por cientistas de dados. (Adaptado: 68)

2.5 Engenharia de *Features*

Após o longo processo de filtragem e limpeza dos dados, é efetuada a etapa de engenharia de *features* sobre os dados, caso os usuários desejem realizar a criação de modelos de predição e/ou classificação baseada nestes dados previamente tratados. Neste contexto, a noção de *features* pode ser descrita como a capacidade de denotar a estrutura funcional e as propriedades visíveis de um sistema, ou seja, são entradas que os modelos de aprendizado de máquina utilizam durante o treinamento e a inferência para fazer previsões [71].

Esta etapa, promove as *features* como objetos de primeira classe durante todo o ciclo de vida do sistema e em todos os domínios de problemas e soluções, além de envolver combinações de diversas análises de dados, conhecimento e domínio da área relacionada aos dados selecionados. É essencial considerar, quais dados são necessários e relevantes, se informando com especialistas, realizando *brainstorming* ou por meio de pesquisas aprofundadas sobre o assunto. Concluindo esses exercícios, é possível evitar a perda de variáveis preditoras importantes no processo de criação de modelos[72][73].

É possível utilizar a engenharia de *features* em diversas áreas de aplicação, como exemplo, têm-se o projeto realizado por estudantes da Universidade Federal da Bahia(UFBA), que aplicou engenharia de *features* linguísticas para classificação de triplas relacionais no Galego, Português Brasileiro e Espanhol Europeu [74]. Em outro projeto, foi aplicada em

conjunto com a avaliação de *features*, para realizar extração de informações em Notas Fiscais. Neste último, a engenharia de *features* foi utilizada para determinar conjuntos de *features* que possuíam maior relevância para identificação de elementos de notas fiscais eletrônicas [75].

3 TRABALHOS RELACIONADOS

A construção de uma arquitetura de preparação de dados, de forma geral, envolve estabelecer um fluxo de dados, e, com isso, é preciso definir quais técnicas e ferramentas de gerenciamento serão aplicados ao conjunto de dados. Este processo é de suma importância para ser possível organizar e administrar as demais etapas existentes na arquitetura, desde a coleta e ingestão de dados até a transformação, a distribuição e o consumo[76].

3.1 Ingestão de dados

A abundância de fontes geradoras de informações existentes hoje em dia, em diversas áreas do conhecimento e em setores empresariais, desencadeou o fortalecimento do *Big Data*. Com isso, tornou-se essencial a criação de diferentes métodos e técnicas para a organização e tratamento destes dados, que em sua maioria, chegam a atingir uma taxa inigualável e inflexível. Este problema, fez com que se tornasse necessário, o planejamento de diversas formas de coleta de dados, das mais variadas fontes existentes. Além da organização de coleta, foi preciso determinar técnicas e desenvolver ferramentas, especializadas em realizar a ingestão de dados para os diferentes destinos possíveis. Para esta etapa de Extrair, Transformar e Carregar (ETL), é imprescindível selecionar as ferramentas corretas para atender aos requisitos atuais e futuros [77][78][79].

3.1.1 ISOBlue HD

O trabalho [80] apresenta o *ISOBlue HD*, uma plataforma de código aberto para aquisição de dados em máquinas agrícolas. Essa plataforma, composta por um computador de placa única, modem celular, armazenamento local e *switch Power-over-Ethernet* para sensores, possibilita a obtenção de dados ricos em contexto, permitindo a identificação do *status* da máquina e da logística agrícola. O sistema oferece recursos como diagnóstico e acesso remoto, inicialização e desligamento automático por meio das operações do veículo e utiliza o *Apache Kafka* para possibilitar a troca robusta de dados. O *ISOBlue HD* foi testado em uma colheitadeira durante uma colheita de trigo em 2019, capturando dados significativos com foco na plataforma e nas ações do operador. As análises destacam a capacidade de inferir conhecimento contextual sobre a semântica dos dados dos sensores e a logística da colheita, como, por exemplo, caminhos, velocidades, *status* da plataforma, transferência de material, entre outros fatores.

O software desenvolvido, utilizou as customizações de um *Board Support Package (BSP)* de código aberto. A figura a seguir apresenta a estrutura definida do *BSP*, o qual contém o *Ångström*, um sistema operacional leve, aplicativos que encapsulam programas,

como o de gerenciamento de sistema, programas de gerenciamento de energia, um *cluster Kafka* e registradores de dados, além de *drivers* de dispositivo e *middlewares* adicionais.

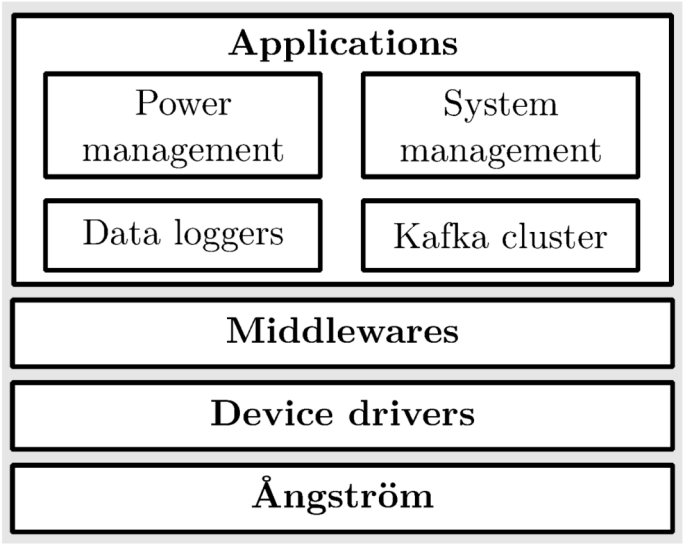

Figura 8 – Estrutura Board Support Package (BSP), desenvolvida com quatro áreas de aplicativos executados em um sistema operacional Ångström: gerenciamento de sistema, gerenciamento de energia, cluster de dados e registro de dados. (Fonte: 80)

Neste trabalho, o *Apache Kafka* atuou com um *cluster* responsável por gerenciar quatro tópicos: *imp* (dados de barramento Implement), *tra* (dados de barramento de trator), *gps* (dados do GPS) e *remote* (GPS e dados de diagnóstico). Os registradores de dados conectam-se ao *cluster* como produtores Kafka para publicar dados de sensores para esses tópicos. O software *MirrorMaker* do kafka sincroniza os dados que fornecem informações de diagnóstico e geoespaciais da plataforma, armazenados no tópico remoto para um *cluster* Kafka remoto. Três produtores Kafka personalizados foram implementados para registrar *CAN*, *GNSS* e dados de diagnóstico: *kafka-can-log*, *kafka-gps-log* e *heartbeat*. Dois desses produtores, *kafka-can-log* e *heartbeat*, foram escritos em C e *kafka-gps-log* foi escrito em Python.

Cada aplicativo inicia sua operação conectando-se ao *cluster Kafka*, e, em seguida, tenta estabelecer conexão com uma fonte de dados específica para, posteriormente, entrar em um ciclo de leitura, serialização e, por fim, publicação de dados no *cluster Kafka*, por meio de *APIs* de cliente *Kafka*. As variações surgem das distintas fontes de dados utilizadas. Para ilustrar, o *kafka-can-log* cria um soquete de rede *SocketCAN* para escutar em um barramento *CAN*. Já o *kafka-gps-log* conecta-se ao *gpsd* via *gps3* para obter dados de *GPS*. Enquanto isso, o *heartbeat* verifica diretamente o *status* da conexão com a Internet e a intensidade do sinal celular a cada segundo por meio de um comando do sistema. Após a publicação dos dados de diferentes registradores no *cluster Kafka*, esses

dados são armazenados no *SSD*.

3.1.2 Billion Object Platform (BOP)

Neste trabalho [81], com o apoio financeiro da *Sloan Foundation* e *Harvard Dataverse*, o *Harvard Center for Geographic Analysis (CGA)*, foi desenvolvida a *Billion Object Platform*, conhecida como *BOP*, uma robusta plataforma de visualização para dados espaço-temporais. O principal propósito desse projeto é eliminar as barreiras enfrentadas por acadêmicos que buscam acessar extensos conjuntos de dados espaço-temporais em tempo real. Dada a rápida expansão dos dados de *streaming* após sua arquivagem e a limitação da maioria dos sistemas *GIS* em lidar com a visualização interativa de milhões de objetos, a criação de uma nova plataforma tornou-se substancial.

Na arquitetura principal da *Billion Object Platform (BOP)*, ilustrada na figura abaixo, o *BOP* é alimentado pelos mais recentes bilhões de *tweets* geográficos e recebe um fluxo constante de aproximadamente 1 milhão de *tweets* por dia em tempo real. Desde 2012, a *CGA* tem coletado e arquivado *geo-tweets*, enriquecendo-os com informações de sentimentos e dados censitários para facilitar análises mais profundas. A transmissão e armazenamento dos dados de entrada e intermediários são realizados através do *Apache Kafka*, arquivando *geo-tweets* usando um tópico de longo prazo. O núcleo do *BOP* é o *Apache Solr*, que oferece suporte a buscas rápidas. Significativas melhorias foram implementadas no *Solr*, incluindo a contribuição notável do facetamento de mapa de calor 2D para aprimorar a visualização espacial.

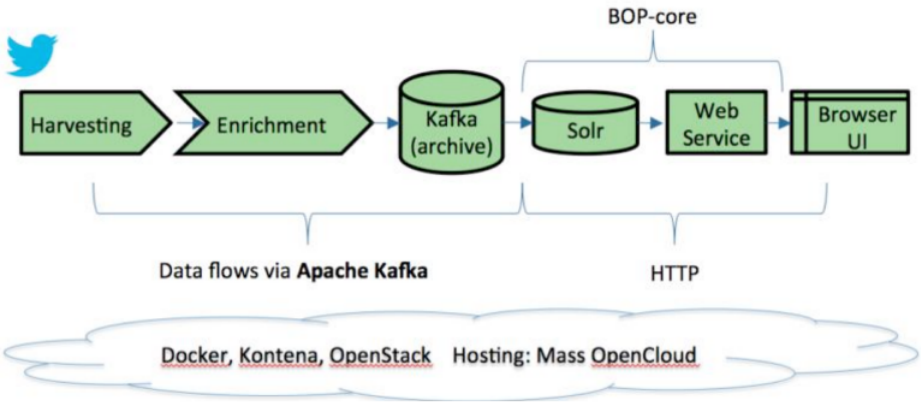

Figura 9 – Arquitetura lógica de alto nível BOP. (Fonte: 81)

O *BOP* disponibiliza ao *Solr* um serviço *web RESTful*, proporcionando uma *API* segura de pesquisa e extração, acessível a partir de um cliente baseado em navegador. O cliente desenvolvido dinamicamente exibe distribuições temporais e espaciais de resultados para conjuntos que contêm centenas de milhões de recursos. O sistema é de código aberto, compatível com *hardware* convencional e hospedado no *Massachusetts Open Cloud*

(MOC), uma plataforma *OpenStack*. Todos os componentes são implantados em contêineres *Docker* e orquestrados pela *Kontena*. O termo *BOP-core* é utilizado para referenciar tanto o índice *Solr* quanto o serviço da *web* associado.

Neste projeto, a *API Kafka Streams* é empregada para consumir de um *stream*, transferir conteúdo para outro *stream* e mover dados do *Kafka* para sistemas como o *Solr*. O *Kafka* é utilizado de maneira não convencional como um armazenamento persistente de dados a longo prazo. Os *tweets* de arquivos são armazenados em partições mensais em um tópico *Kafka* de longo prazo, simplificando o processo, mas apresentando desafios como a impossibilidade de classificar ou desduplicar dados no *Kafka*. Foi configurado um *cluster Kafka* com 3 *zookeepers* e 3 *brokers*. Criaram 2 consumidores para enriquecimento, um ingeridor e 2 tópicos para entrada e saída. No processo de enriquecimento do *BOP*, ilustrado na figura 10, o tópico de entrada contém tweets em tempo real, alimentado pela colheitadeira. Este tópico tem os *tweets* consumidos pelo processo de enriquecimento e encaminhados para o tópico de saída. Posteriormente, os *tweets* enriquecidos são consumidos deste ultimo tópico pelo ingeridor *BOP*, que os transforma em documentos *Solr* antes de enviá-los para o *Solr*. A *API Kafka Manager* do *Yahoo* é utilizado para administrar o *Kafka*.

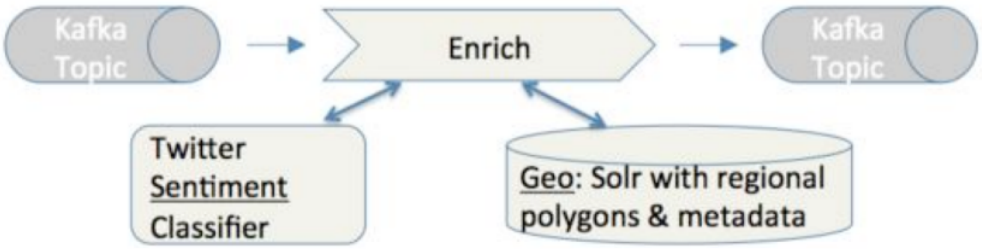

Figura 10 – Processo de enriquecimento no BOP. (Fonte: 81)

Diferente do *ISOBlue HD*, que utiliza o *Apache Kafka* para a transferência robusta de dados de uma colheitadeira, capturando dados significativos com foco na plataforma e nas ações do operador para análise posterior e do *BOP*, que realiza a transmissão e armazenamento dos dados *geo-tweets* usando um tópico de longo prazo do *Apache Kafka* para enriquecimento e disponibilização dos dados para pesquisa e extração, este projeto tem como foco a coleta de dados agrícolas de diversas fontes, como dados provenientes de bancos de dados, arquivos estruturados e não estruturados e dados de diferentes estações meteorológicas, além de realizar a transmissão desses dados para diversas tarefas escritas em *Python*, focadas em preparação de dados, geoprocessamento, engenharia de *features* para criação de modelos e visualização em um servidor *web* para análise de resultados.

3.2 Integração

3.2.1 Polly

Neste trabalho [82], a arquitetura é composta por uma interface que possibilita aos usuários carregar seus dados em formatos como *Excel* ou *CSV*. Essa interface permite a aplicação de algoritmos de aprendizado de máquina, a escolha de diferentes parâmetros como variáveis de treinamento e a especificação de uma variável alvo para previsão. Após a seleção de atributos, os usuários podem realizar diversas análises estatísticas e operações de limpeza nos dados. Uma vez concluída a fase de pré-processamento, os usuários escolhem entre diversos algoritmos disponíveis de aprendizado de máquina. Após a seleção do algoritmo, ajustes nos hiperparâmetros podem ser feitos para atender aos requisitos específicos do aplicativo.

A interface *front-end* encaminha cada operação para o serviço web, implementado em *Python* usando *Flask*, um *framework web* leve. A arquitetura foi projetada com foco na escalabilidade, executando cada operação do *Polly* em contêineres Docker separados. Isso proporciona flexibilidade e capacidade de resposta, permite modelagem, limpeza e análise de dados de forma eficiente.

O fluxo de trabalho da arquitetura foi separado em quatro etapas. A primeira é alimentação de dados, permite o usuário realizar *upload* de seus dados em formato *Excel* ou conectar-se ao banco de dados *SQL* remoto. A segunda é de análise estatística e processamento de dados, em que o usuário pode visualizar estatísticas básicas de seu conjunto de dados, realizadas automaticamente pela arquitetura, incluindo cálculo de média, moda, mediana, desvio padrão, valores percentuais e quantidade de valores nulos de cada coluna de dados e preenchimento ou exclusão de linhas e colunas que possuem valores ausentes. A terceira etapa é de seleção de algoritmo. Para a análise de regressão, têm-se os seguintes: *Linear, Lasso, Lars (LLR)*, *Elastic Net Regression (ENR)* e *Support Vector Regression (SVR)*. Para análise de classificação, incorporaram seis algoritmos: *Gaussian Naive Bayes (GNB)*, *Bernoulli Naive Bayes*, *Ada Booster*, *Decision Tree*, *Random Forest Tree* e *Support Vector Classifier*. Nesta etapa, também é possível o usuário ajustar o espaço de hiperparâmetros dos algoritmos selecionados. Por fim, na quarta etapa é realizado a visualização de resultados, o usuário pode avaliar os resultados do algoritmo selecionado com o auxílio de métricas definidas por cada tipo de análise, bem como comparar o desempenho de cada algoritmo selecionado.

O caso de uso utilizado, foi um conjunto de dados biofísicos e bioquímicos de plantas de soja coletados em um campo agrícola aberto no Missouri, EUA. O trabalho demonstra as capacidades de regressão e aprendizado de máquina de *Polly* na estimativa da clorofila foliar e concentração de nitrogênio, índice de área foliar, biomassa fresca acima do solo e biomassa seca da soja.

Assim como a ferramenta *Polly*, este trabalho também busca aplicar contêineres *Docker*. Porém, diferente da ferramenta *Polly* que implantou contêineres *Docker* separados para integrar as execuções de cada operação, este projeto visa a aplicação dos contêineres *Docker* para a integração de diferentes ferramentas, bibliotecas e sistemas, como o *Apache Airflow*, *Apache Kafka* e bibliotecas *Python* de geoprocessamento, facilitando a criação do ambiente de execução dos processos, estabelecendo conexões eficientes entre as ferramentas integradas e permitindo escalar a arquitetura posteriormente.

3.3 Gerenciamento de fluxo de trabalho

3.3.1 Construindo fluxos de trabalho de agregação de metadados usando Apache Airflow

A PA Digital é uma rede estadual na Pensilvânia que desempenha um papel central como o *hub* de serviços para a Biblioteca Pública Digital da América (DPLA). Em 2014, o grupo desenvolveu um sistema de agregação local, utilizado para reunir registros de coleções digitais de diversas instituições colaboradoras. Este sistema não apenas valida e transforma os metadados dessas coleções, mas também entrega os registros agregados à DPLA. Desde o seu lançamento inicial, a PA Digital experimentou um crescimento significativo, expandindo sua base de colaboradores e incorporando uma diversidade de sistemas de repositório. Com a introdução de cada novo sistema, a complexidade do software agregador personalizado aumentou, tornando-se desafiador de manter. Em 2018, a equipe da PA Digital reconheceu a necessidade de uma nova abordagem.

Com isso, entre 2019 e 2021, uma equipe multifuncional implementou a solução [52], mais flexível e escalável para a agregação de metadados na PA Digital. Para gerenciar os fluxos de trabalho, foi adotado o *Apache Airflow*, enquanto para a revisão interna de metadados, optaram pelo *Solr/Blacklight*.

Para administrar a diversidade de fontes de dados externas agregadas, a equipe da PA Digital concebeu um modelo para criar um Grafo Direcionado Acíclico (DAG) para cada instituição contribuinte. As tarefas dentro de cada DAG e as funções que desempenham são amplamente padronizadas, com base nesse modelo. Para personalizar esses fluxos de trabalho de forma individual, a equipe do projeto utiliza a configuração de variáveis no *Airflow*, associando cada *DAG* a um conjunto específico de valores personalizados para a instituição contribuinte. Essas variáveis são armazenadas em um arquivo *JSON* e são atualizadas regularmente no ambiente do *Airflow*.

Cada *DAG* da PA Digital, ilustrada na figura abaixo, executa um fluxo de trabalho que realiza a coleta, validação, transformação e publicação de metadados no *Solr*. Esse processo está conectado à instância de bibliotecas *Apache SolrCloud* da *Temple University*, uma abordagem distribuída para a indexação do *Solr*.

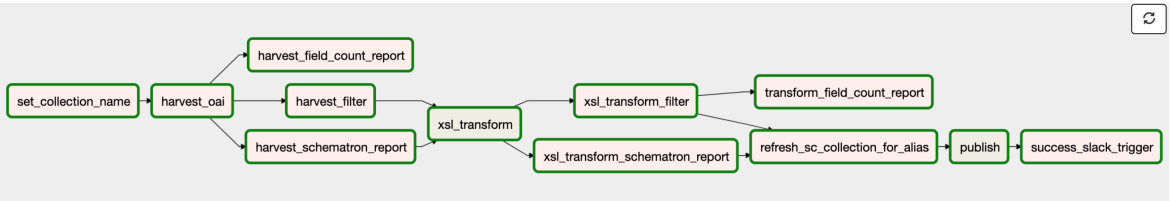

Figura 11 – Visualização gráfica de um PA Digital DAG no Airflow. (Fonte: 52)

3.3.2 Estrutura de processamento de dados do Sentinel-2

O estudo de [83] apresentou uma estrutura de *Big Data* voltada para o processamento de dados, no âmbito das salvaguardas nucleares, provenientes da estação meteorológica *Sentinel-2*, vinculada ao programa *Copernicus*. Dada a diversidade potencial dos dados, uma preparação abrangente foi essencial para adequar os dados a formatos utilizáveis no processo, permitindo assim a aplicação de algoritmos de análise apropriados.

A fonte de dados é integrada ao sistema de gerenciamento de fluxo de trabalho *Apache Airflow*, que tem a capacidade de baixar, validar, pré-processar e armazenar os dados em um banco de dados *Rasdaman*. Para finalizar, a eficiência do *Google Earth Engine* foi explorada para executar de maneira efetiva fluxos de trabalho de *Big Data*, utilizando técnicas de aprendizado de máquina fornecidas pelo *Google*. O potencial da estrutura desenvolvida foi avaliado por meio de estudos de caso relacionados a locais vinculados ao ciclo do combustível. O objetivo principal consistiu em classificar o uso da cobertura do solo, uma vez que essas características oferecem informações cruciais, para identificar e monitorar mudanças no estado operacional, construção de novos edifícios, expansões de fábricas, entre outros.

Neste trabalho, a implementação do gerenciamento de fluxo de trabalho foi definida em *DAGs*, com base em dois arquivos de configuração. O primeiro contém uma visão geral das áreas de interesse, definidas por um nome, o polígono correspondente e o satélite ao qual os dados devem ser solicitados. A configuração do satélite correspondente é armazenada em outro arquivo, onde seu nome, provedor e tipo de produto definem cada satélite. O *Open Access Hub* disponibilizou os dados no Nível 2A, corrigidos e ortorretificados.

Para cada provedor de satélite, foi criado um DAG, como ilustrado na figura 12. No entanto, como o *Copernicus Hub* só pode processar duas solicitações de servidor por vez, o nó *CopernicusHub_start* possui apenas dois nós filhos.

No user interface (UI), os nós azuis representam os grupos de tarefas, que são um conceito de agrupamento de UI e úteis para criar repetições de padrões. Em cada grupo de tarefas, como ilustrado na figura 13, a mesma sequência de tarefas é executada, alterando apenas os parâmetros de entrada.

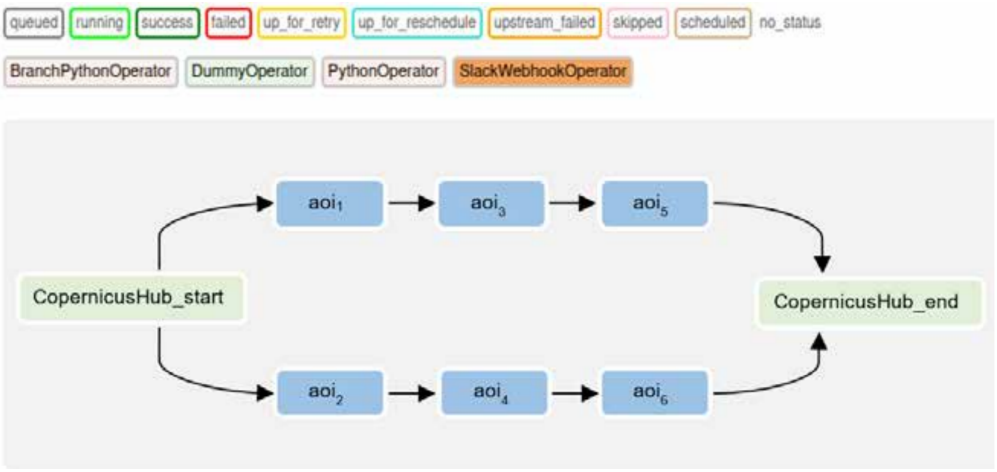

Figura 12 – DAG criado dinamicamente com base nos dois arquivos de configuração. (Fonte: 83)

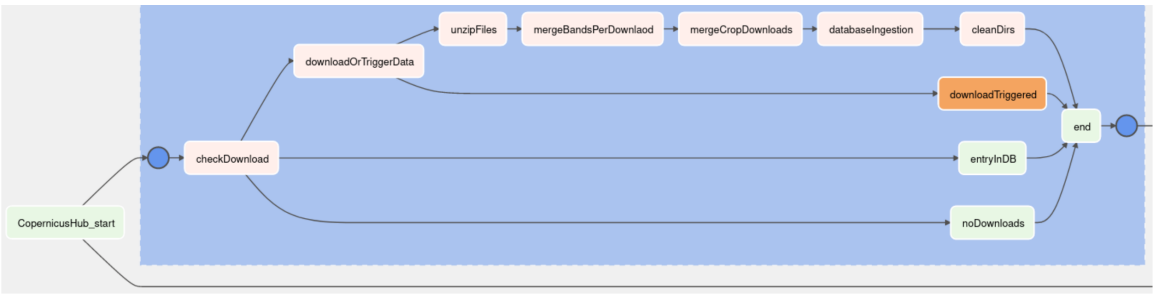

Figura 13 – Tarefas reais a serem executadas por área organizadas em Grupos de Tarefas. (Fonte: 83)

Assim como nos trabalhos citados acima, nesta sessão, que orquestraram seus fluxos de trabalho com grafos acíclicos direcionados, a arquitetura de preparação de dados agrícolas que será desenvolvida também implantará o gerenciamento dos fluxos com a ferramenta *Apache Airflow*. Porém, diferente dos projetos citados, em que apenas um pipeline de execução foi desenvolvido para diversos sistemas, alterando apenas as variáveis de entrada, este projeto visa o desenvolvimento de diversos pipelines de execução, para diferentes tipos de dados, atendendo mais de um tipo de processamento de dados e criação de modelos.

3.4 Geoprocessamento

3.4.1 Nansat

Nansat [84] representa uma caixa de ferramentas *Python* dedicada à análise e processamento de dados geoespaciais bidimensionais, abrangendo imagens de satélite, resultados de modelos numéricos e dados *in-situ* em grade. Desenvolvido com uma ênfase

marcante, na simplificação da pesquisa e no avanço de algoritmos e sistemas de processamento autônomo, *Nansat* estende as funcionalidades da Biblioteca de Dados de Abstração Geoespacial (GDAL). Isso é realizado ao adicionar significado científico aos conjuntos de dados, por meio de metadados e incorporar funcionalidades comuns para análise e manipulação de dados, incluindo a exportação para diversos formatos de dados.

Um dos cases desenvolvidos, cobre a colocação espacial de duas imagens adquiridas em maio de 2016, pelo instrumento Radar de Abertura Sintética (SAR), a bordo do satélite *Sentinel-1A* e pelo Espectrorradiômetro de Imagem de Resolução Moderada (MODIS), a bordo do satélite Aqua. Os dois conjuntos de dados originais de nível 1 têm referência espacial, resolução e cobertura diferentes, e são colocados e projetados na mesma grade espacial e, em seguida, visualizados usando *Nansat*. As imagens resultantes, ilustradas na figura a baixo, mostram radiâncias do topo da atmosfera MODIS, medidas em 469, 555 e 858 nm, exibidas em RGB (Figura 14A) e uma imagem em escala de cinza da mesma área gerada, a partir de SAR HH (transmissão horizontal e recepção horizontal) (Figura 14B), cobrindo partes do Mar da Noruega. Nestas imagens, é possível visualizar redemoinhos oceânicos de meso escala, manifestados tanto pela cor do oceano (Figura 14A), quanto pela rugosidade da superfície do mar (Figura 14B).

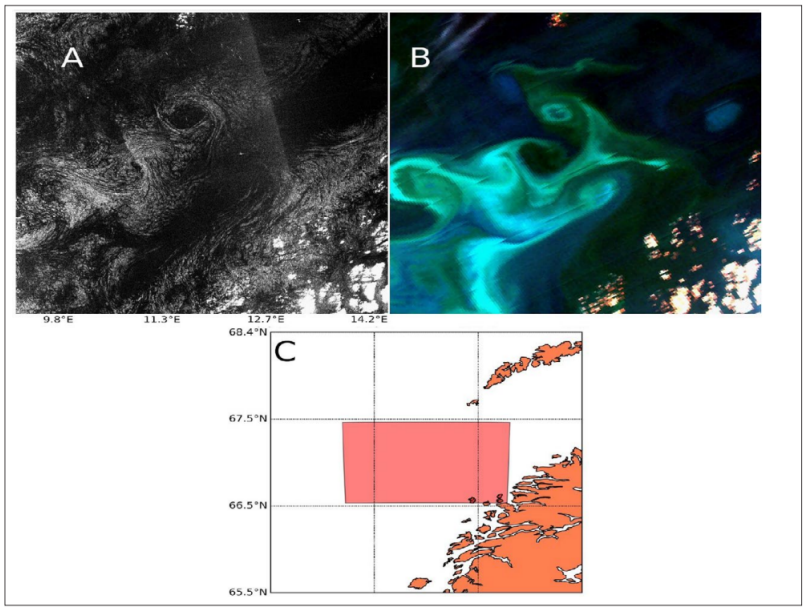

Figura 14 – Exemplo de uma aplicação simples do Nansat para colocar e visualizar observações síncronas adquiridas pelo MODIS e Sentinel1A em 25 de maio de 2011. (A) Uma imagem RGB de uma aquisição MODIS/Aqua sobre o Mar da Noruega, (B) uma imagem SAR em escala de cinza do Sentinel- 1A e (C) a cobertura de dados. (Fonte: 84)

3.4.2 Explorando a visualização de dados geoespaciais baseada em Python

O *OpenStreetMap (OSM)* é uma vasta fonte de dados de informação geográfica de código aberto, oferecendo informações geográficas editáveis de classe mundial. Esses dados constituem uma valiosa fonte para a mineração, análise e visualização de dados geoespaciais. O estudo [85] utiliza os dados do OSM e aplica a tecnologia de processamento de dados de informação geográfica do *Python*, técnicas de visualização de dados geoespaciais e um método de sobreposição de camadas baseado em algoritmos de análise de dados vetoriais. A abordagem empregada permite a visualização e anotação de informações provenientes de diversos tipos de dados geoespaciais, oferecendo métodos confiáveis para o processamento, visualização e análise de dados. Esses métodos são aplicados na exploração de recursos domésticos, no setor de transporte, na prevenção e controle de epidemias, e em outros campos relevantes.

Neste projeto, conduziu-se uma análise de buffer e mapeamento de projeção, empregando a função *from_file('CHN_adm1.shp')* do *GeoPandas* para extrair dados geográficos do conjunto de dados aberto *CHN_adm1.shp* do *OpenStreetMap*. A leitura foi realizada no formato *GeoDataFrame*, e métodos construtivos do *GeoPandas* foram aplicados para gerar novos dados vetoriais a partir do *GeoDataFrame*. Além disso, foram calculados diversos atributos com base em *GeoSeries*, como limites, linhas de contorno internas e externas, centro de gravidade, entre outros critérios.

Para ajustar a projeção dos geodados, foi utilizada a função *to_crs('EPSG:4490')* do *GeoDataFrame*, permitindo a modificação do código EPSG. Essa mudança é essencial, uma vez que diferentes códigos EPSG proporcionam diferentes projeções da superfície terrestre, corrigindo distorções específicas.

Posteriormente, diversas análises foram conduzidas sobre a estrutura de dados do *GeoDataFrame*. Notadamente, a coluna *geometria* no quadro de dados geográficos, foi definida para representar formas geométricas, polígonos ou polígonos múltiplos, dependendo das características específicas do conjunto de dados.

Como nos trabalhos descritos acima, a arquitetura de preparação de dados também visa a utilização de ferramentas de geoprocessamento, como o *GDAL* e o *GeoPandas* citados, além de outras ferramentas como o *Raster.IO* e *QGIS*, para o tratamento de dados geoespaciais, como, por exemplo, normalização, remoção de ruídos, recortes, destaque de áreas de interesse e vários outros processamentos.

4 ESTUDO DE CASO

5 CONCLUSÃO

REFERÊNCIAS

[1] ECONOMIST, T. *The world's most valuable resource is no longer oil, but data*. 2017. Disponível em: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. Acesso em: 21 jun 2023.

[2] RYO, M. Explainable artificial intelligence and interpretable machine learning for agricultural data analysis. *Artificial Intelligence in Agriculture*, v. 6, p. 257–265, 2022. ISSN 2589-7217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2589721722000216>.

[3] BENOS, L. et al. Machine learning in agriculture: A comprehensive updated review. *Sensors*, v. 21, n. 11, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/11/3758>.

[4] LIAKOS, K. G. et al. Machine learning in agriculture: A review. *Sensors*, v. 18, n. 8, 2018. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/18/8/2674>.

[5] SCIFORCE. *Machine Learning in Agriculture: Applications and Techniques*. 2019. Disponível em: https://medium.com/sciforce/machine-learning-in-agriculture-applications-and-techniques-6ab501f4d1b5. Acesso em: 05 jul 2023.

[6] YUAN, Y. et al. Advanced agricultural disease image recognition technologies: A review. *Information Processing in Agriculture*, v. 9, n. 1, p. 48–59, 2022. ISSN 2214-3173. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214317321000032>.

[7] KÜHN, D. *Pesquisa e Análise de Dados: problematizando o rural e a agricultura numa perspectiva científica (DERAD604)*. PLAGEDER, 2017. (Série Ensino, Aprendizagem e Tecnologias - UFRGS). ISBN 9788538603788. Disponível em: <https://books.google.com.br/books?id=YZU6DwAAQBAJ>.

[8] BREIMAN, L. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, Institute of Mathematical Statistics, v. 16, n. 3, p. 199 – 231, 2001. Disponível em: <https://doi.org/10.1214/ss/1009213726>.

[9] EMBRAPA. *Avanço da ciência de dados e big data, inteligência artificial, aprendizado de máquina e cooperativas de dados*. Disponível em: https://www.embrapa.br/visao-de-futuro/agrodigital/sinal-e-tendencia/avanco-da-ciencia-de-dados-e-big-data-inteligencia-artificial-aprendizado-de-maquina-e-cooperativas-de-dados. Acesso em: 05 jul 2023.

[10] MICROSOFT. *What is Azure Databricks?* 2023. Disponível em: https://learn.microsoft.com/pt-br/azure/databricks/introduction/. Acesso em: 09 ago 2023.

[11] ORACLE. *Build a secure OCI Data Integration environment with pre-built tasks from templates*. 2023. Disponível em: https://docs.oracle.com/en/solutions/oci-data-integration/. Acesso em: 09 ago 2023.

[12] AMAZON. *Amazon SageMaker Data Wrangler*. 2023. Disponível em: https://aws.amazon.com/pt/sagemaker/data-wrangler/. Acesso em: 09 ago 2023.

[13] HAT, R. *What is integration?* 2017. Disponível em: https://www.redhat.com/en/topics/integration/what-is-integration. Acesso em: 12 set 2023.

[14] BEURDEN, I. van. *The Meaning of Tool Integration*. 2016. Disponível em: https://www.exida.com/blog/the-meaning-of-tool-integration. Acesso em: 12 set 2023.

[15] SYDLE. *Systems Integration: Learn the Kinds, Challenges, and Importance*. 2022. Disponível em: https://www.sydle.com/blog/systems-integration-6140d39a84679b13bf127a93. Acesso em: 12 set 2023.

[16] SAP. *What is application integration?* 2023. Disponível em: https://www.sap.com/brazil/products/technology-platform/what-is-enterprise-integration/application-integration.html. Acesso em: 13 set 2023.

[17] EDUCATION, I. C. *What Is an Integration Platform? Do I Need One?* 2021. Disponível em: https://www.ibm.com/blog/integration-platform/. Acesso em: 13 set 2023.

[18] LASSO, A.; KAZANZIDES, P. Chapter 35 - system integration. In: ZHOU, S. K.; RUECKERT, D.; FICHTINGER, G. (Ed.). *Handbook of Medical Image Computing and Computer Assisted Intervention*. Academic Press, 2020, (The Elsevier and MICCAI Society Book Series). p. 861–891. ISBN 978-0-12-816176-0. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128161760000405>.

[19] TENSORFLOW. *TFX is an end-to-end platform for deploying production ML pipelines*. 2023. Disponível em: https://www.tensorflow.org/tfx. Acesso em: 13 set 2023.

[20] MODI, A. N. et al. Tfx: A tensorflow-based production-scale machine learning platform. In: *KDD 2017*. [S.l.: s.n.], 2017.

[21] SAWANT, N.; SHAH, H. Big data ingestion and streaming patterns. In: _____. *Big Data Application Architecture Q & A: A Problem-Solution Approach*. Berkeley, CA: Apress, 2013. p. 29–42. ISBN 978-1-4302-6293-0. Disponível em: <https://doi.org/10.1007/978-1-4302-6293-0_3>.

[22] INTEGRAçãO de big data. In: 2013 IEEE 29ª Conferência Internacional sobre Engenharia de Dados (ICDE). [S.l.: s.n.].

[23] STITCH. *Data ingestion: the first step to a sound data strategy*. 2023. Disponível em: <https://www.stitchdata.com/resources/data-ingestion/>. Acesso em: 05.12.2023.

[24] MAVROGIORGOU, A. et al. Batch and streaming data ingestion towards creating holistic health records. *Emerging Science Journal*, v. 7, n. 2, p. 339–353, 2023.

[25] HOW, M. The ingestion architecture. In: _____. *The Modern Data Warehouse in Azure: Building with Speed and Agility on Microsoft's Cloud Platform*. Berkeley, CA: Apress, 2020. p. 105–132. ISBN 978-1-4842-5823-1. Disponível em: <https://doi.org/10.1007/978-1-4842-5823-1_4>.

[26] INGESTãO de Big Data e Arquitetura de Aprendizagem ao Longo da Vida. In: 2018 Conferência Internacional IEEE sobre Big Data (Big Data). [S.l.: s.n.].

[27] INGESTãO em tempo real de Big Data e aprendizado de máquina. In: 2018 IEEE Segunda Conferência Internacional sobre Mineração e Processamento de Fluxo de Dados (DSMP). [S.l.: s.n.].

[28] PICCARDI, M. L. S.; PALOMO, L. E. Del big data al fast data: enfoques modernos de streaming de datos para el procesamiento de datos masivos en tiempo real. *Difusiones*, v. 21, n. 21, p. 38–58, 2021.

[29] APACHE. *Apache Sqoop*. 2019. Disponível em: <https://sqoop.apache.org/>. Acesso em: 07.12.2023.

[30] TING, K.; CECHO, J. J. *Apache sqoop cookbook: Unlocking hadoop for your relational database*. [S.l.]: "O'Reilly Media, Inc.", 2013.

[31] VOHRA, D. Using apache sqoop. In: _____. *Pro Docker*. Berkeley, CA: Apress, 2016. p. 151–183. ISBN 978-1-4842-1830-3. Disponível em: <https://doi.org/10.1007/978-1-4842-1830-3_11>.

[32] ALWIDIAN, J. et al. Big data ingestion and preparation tools. *Modern Applied Science*, Canadian Center of Science and Education, v. 14, n. 9, p. 12–27, 2020.

[33] VOHRA, D. Apache flume. In: _____. *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*. Berkeley, CA: Apress, 2016. p. 287–300. ISBN 978-1-4842-2199-0. Disponível em: <https://doi.org/10.1007/978-1-4842-2199-0_6>.

[34] SRINIVASA, K. G.; M., S. G.; H., S. Apache flume. In: _____. *Network Data Analytics: A Hands-On Approach for Application Development*. Cham: Springer International Publishing, 2018. p. 95–107. ISBN 978-3-319-77800-6. Disponível em: <https://doi.org/10.1007/978-3-319-77800-6_6>.

[35] BIRJALI, M.; BENI-HSSANE, A.; ERRITALI, M. Analyzing social media through big data using infosphere biginsights and apache flume. *Procedia Computer Science*, v. 113, p. 280–285, 2017. ISSN 1877-0509. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050917317088>.

[36] APACHE. *Apache Kafka*. 2023. Disponível em: <https://kafka.apache.org/>. Acesso em: 07.12.2023.

[37] MĂTĂCUŢĂ, A.; POPA, C. Big data analytics: Analysis of features and performance of big data ingestion tools. *Informatica Economica*, v. 22, n. 2, 2018.

[38] VOHRA, D. Apache kafka. In: _____. *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*. Berkeley, CA: Apress, 2016. p. 339–347. ISBN 978-1-4842-2199-0. Disponível em: <https://doi.org/10.1007/978-1-4842-2199-0_9>.

[39] GARG, N. *Apache kafka*. [S.l.]: Packt Publishing Birmingham, UK, 2013.

[40] RAPTIS, T. P.; PASSARELLA, A. A survey on networked data streaming with apache kafka. *IEEE Access*, v. 11, p. 85333–85350, 2023.

[41] RAPTIS, T. P.; PASSARELLA, A. A survey on networked data streaming with apache kafka. *IEEE access*, IEEE, Piscataway, v. 11, p. 1–1, 2023. ISSN 2169-3536.

[42] DATAFLAIR. *Kafka Architecture and Its Fundamental Concepts*. 2023. Disponível em: <https://data-flair.training/blogs/kafka-architecture/>. Acesso em: 07.12.2023.

[43] ANKAM, V. *Big data analytics*. [S.l.]: Packt Publishing Ltd, 2016.

[44] RACKA, K. Apache nifi as a tool for stream processing of measurement data. *Nauki Ekonomiczne*, 2022.

[45] APACHE. *Apache NiFi Overview*. 2023. Disponível em: <https://nifi.apache.org/docs/nifi-docs/html/overview.html>. Acesso em: 09.12.2023.

[46] ALONSO, G.; MOHAN, C. Workflow management: the next generation of distributed processing tools. In: *Advanced Transaction Models and Architectures*. [S.l.]: Springer, 1997. p. 35–59.

[47] TALIA, D. Workflow systems for science: Concepts and tools. *International Scholarly Research Notices*, Hindawi, v. 2013, 2013.

[48] IBM. *What is a workflow?* 2023. Disponível em: <https://www.ibm.com/br-pt/topics/workflow>. Acesso em: 09.12.2023.

[49] HE, J. et al. Consensus mechanism design based on structured directed acyclic graphs. *Blockchain: Research and Applications*, v. 2, n. 1, p. 100011, 2021. ISSN 2096-7209. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2096720921000063>.

[50] HARENSLAK, B. P.; RUITER, J. de. *Data Pipelines with Apache Airflow*. [S.l.]: Simon and Schuster, 2021.

[51] SINGH, P.; SINGH, P. Airflow. *Learn PySpark: Build Python-based Machine Learning and Deep Learning Models*, Springer, p. 67–84, 2019.

[52] FINNIGAN, L.; TONER, E. Building and maintaining metadata aggregation workflows using apache airflow. *Temple University Libraries*, 2021.

[53] SAINI, S.; BHATT, R. Airflow automator-streamlining workflows with dags. Jaypee University of Information Technology, Solan, HP, 2023.

[54] SHUBHA, B.; PRASAD, A. Airflow directed acyclic graph. *J Signal Process*, v. 5, n. 2.

[55] KEDRO. *Introduction to Kedro*. Disponível em: <https://docs.kedro.org/en/stable/introduction/index.html>.

[56] AMAZON. *Using Kedro pipelines to train Amazon SageMaker models*. 2023. Disponível em: <https://aws.amazon.com/pt/blogs/opensource/using-kedro-pipelines-to-train-amazon-sagemaker-models/>. Acesso em: 09.12.2023.

[57] WANG, J. *Encyclopedia of Data Science and Machine Learning*. IGI Global, 2023. (Advances in Data Mining and Database Management Series). ISBN 9781799892212. Disponível em: <https://books.google.com.br/books?id=I2-pEAAAQBAJ>.

[58] KEDRO. *Why Kedro?* 2023. Disponível em: <https://kedro.org/>. Acesso em: 09.12.2023.

[59] PERCHAIS, G. *Why We Switched Our Data Orchestration Service*. 2022. Disponível em: <https://engineering.atspotify.com/2022/03/why-we-switched-our-data-orchestration-service/>. Acesso em: 09.12.2023.

[60] LUIGI. *Introduction to Luigi*. 2023. Disponível em: <https://luigi.readthedocs.io/en/stable/>. Acesso em: 09.12.2023.

[61] DATAREVENUE. *What is Luigi?* 2023. Disponível em: <https://www.datarevenue.com/ml-tools/luigi>. Acesso em: 09.12.2023.

[62] LUIGI. *The Enterprise-Ready Micro Frontend Framework*. 2023. Disponível em: <https://luigi-project.io/>. Acesso em: 09.12.2023.

[63] IBM. *O que é machine learning?* 2023. Disponível em: https://www.ibm.com/br-pt/topics/machine-learning. Acesso em: 11 set 2023.

[64] GOOGLE. *What is Machine Learning?* 2023. Disponível em: https://cloud.google.com/learn/what-is-machine-learning. Acesso em: 11 set 2023.

[65] VICKERY, R. *A Beginner's Guide to End to End Machine Learning*. 2021. Disponível em: https://towardsdatascience.com/a-beginners-guide-to-end-to-end-machine-learning-a42949e15a47. Acesso em: 10 set 2023.

[66] MALIK, F. *End To End Guide For Machine Learning Project*. 2018. Disponível em: https://medium.com/fintechexplained/end-to-end-guide-for-machine-learning-project-146c288186dc. Acesso em: 10 set 2023.

[67] KAZIL, J.; JARMUL, K. *Data Wrangling with Python: Tips and Tools to Make Your Life Easier*. O'Reilly Media, 2016. ISBN 9781491956809. Disponível em: <https://books.google.com.br/books?id=XmeDCwAAQBAJ>.

[68] PRESS, G. *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says*. 2016. Disponível em: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=28d139436f63. Acesso em: 12 set 2023.

[69] REILLY, J. *End to End Machine Learning Workflow*. 2022. Disponível em: https://www.akkio.com/post/end-to-end-machine-learning-workflow. Acesso em: 12 set 2023.

[70] AMAZON. *What is Data Preparation?* 2023. Disponível em: https://aws.amazon.com/pt/what-is/data-preparation/. Acesso em: 12 set 2023.

[71] AMAZON. *What Is Feature Engineering?* 2023. Disponível em: <https://aws.amazon.com/what-is/feature-engineering/>. Acesso em: 15.11.2023.

[72] ENGENHARIA de recursos [desenvolvimento de software]. In: PROCEEDINGS Nono Workshop Internacional sobre Especificação e Design de Software. [S.l.: s.n.].

[73] Reid Turner, C. et al. A conceptual basis for feature engineering. *Journal of Systems and Software*, v. 49, n. 1, p. 3–15, 1999. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016412129900062X>.

[74] LUZ, E.; SILVA, C.; CLARO, D. Engenharia de features linguísticas para classificação de triplas relacionais. In: *Anais do XIII Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*. Porto Alegre, RS, Brasil: SBC, 2021. p. 381–388. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/stil/article/view/17818>.

[75] DARRAZãO, E. et al. Engenharia e avaliação de features para extração de informação em notas fiscais. In: *Anais da XVIII Escola Regional de Banco de Dados*. Porto Alegre, RS, Brasil: SBC, 2023. p. 80–89. ISSN 2595-413X. Disponível em: <https://sol.sbc.org.br/index.php/erbd/article/view/24349>.

[76] IBM. *What is a data architecture?* 2023. Disponível em: <https://www.ibm.com/br-pt/topics/data-architecture>. Acesso em: 03.12.2023.

[77] SHARMA, G.; TRIPATHI, V.; SRIVASTAVA, A. Recent trends in big data ingestion tools: A study. In: KUMAR, R. et al. (Ed.). *Research in Intelligent and Computing in Engineering*. Singapore: Springer Singapore, 2021. p. 873–881. ISBN 978-981-15-7527-3.

[78] MEEHAN, J. et al. Data ingestion for the connected world. In: *CIDR*. [S.l.: s.n.], 2017. v. 17, p. 8–11.

[79] CHAWLA, H.; KHATTAR, P. Data ingestion. In: _____. *Data Lake Analytics on Microsoft Azure: A Practitioner's Guide to Big Data Engineering*. Berkeley, CA: Apress, 2020. p. 43–85. ISBN 978-1-4842-6252-8. Disponível em: <https://doi.org/10.1007/978-1-4842-6252-8_4>.

[80] WANG, Y. et al. Isoblue hd: An open-source platform for collecting context-rich agricultural machinery datasets. *Sensors*, v. 20, n. 20, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/20/5768>.

[81] KAKKAR, D. et al. The billion object platform (bop): um sistema para reduzir barreiras para suportar grandes fontes de dados espaço-temporais de streaming. *Software Livre e de Código Aberto para Geoespaciais (FOSS4G) Conference Proceedings*, v. 17, 2017. Disponível em: <https://scholarworks.umass.edu/foss4g/vol17/iss1/15>.

[82] MUHAMMAD, W. et al. Polly: A tool for rapid data integration and analysis in support of agricultural research and education. *Internet of Things*, v. 9, p. 100141, 2020. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S254266051930246X>.

[83] BEUMER, L.; NIEMEYER, I. Developing a big data framework for processing sentinel-2 data in the context of nuclear safeguards evaluation of apache airflow, rasdaman and google earth engine. *ESARDA BULLETIN*, p. 75, 2022.

[84] KOROSOV, A. A. et al. Nansat: A scientist-orientated python package for geospatial data processing. *Journal of Open Research Software*, v. 4, n. 1, p. e39–e39, 2016.

[85] HU, N. et al. Exploring geospatial data visualization based on python. In: *Proceedings of the 2022 2nd International Conference on Control and Intelligent Robotics*. [S.l.: s.n.], 2022. p. 858–862.