	Especificação de Projeto	
c.e.s.a.r school	**DISCIPLINA:** Aprendizado de Máquina	**PERÍODO:** 2025.2
		UNIDADE: 2a

1. Introdução

O projeto da disciplina Aprendizado de Máquina (AM) tem como objetivo a reprodução e melhorias de um artigo científico que explore a aplicação de técnicas de *machine learning* para problemas reais.

Nesta edição, o projeto deverá ser implementado sobre a arquitetura de dados e aprendizado em contêineres, integrando coleta, processamento, modelagem e visualização em um pipeline executável via Docker Compose.

O ambiente deverá contemplar as seguintes camadas, conforme a **Figura 1**:

- **Ingestão** (FastAPI): responsável por receber e disponibilizar os dados para análise.
- **Armazenamento** (MinIO/S3): guarda dados brutos e modelos treinados.
- **Modelagem** (JupyterLab): espaço para experimentos e desenvolvimento de modelos preditivos.
- **Rastreamento** (MLFlow): registro de experimentos, métricas e parâmetros dos modelos.
- **Visualização** (ThingsBoard/Trendz Analytics): dashboards interativos com resultados e insights.

Essa integração visa consolidar as habilidades práticas em ciência de dados aplicada, com foco em **reprodutibilidade, documentação e engenharia de aprendizado de máquina**.

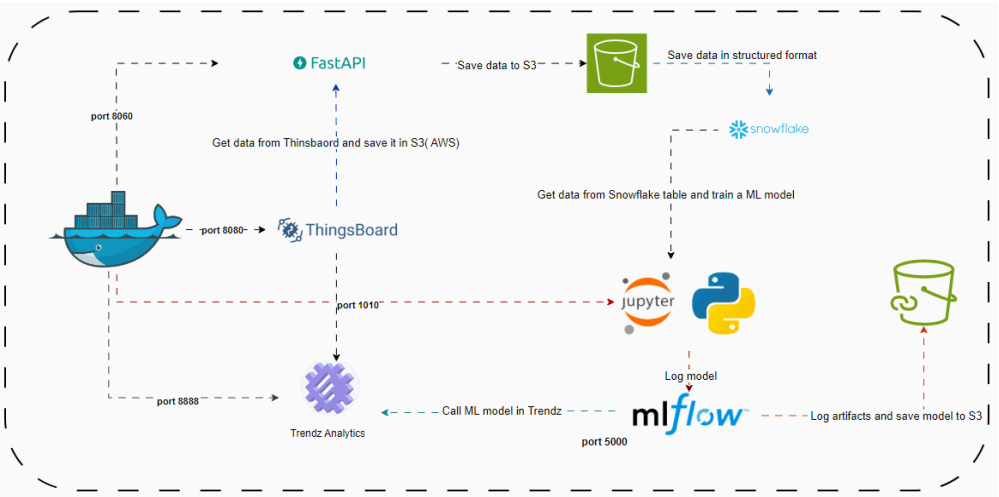

Figura 1 – Arquitetura integrando coleta, processamento e visualização em contêineres.

O projeto deverá ser desenvolvido em grupos de até 6 integrantes e corresponderá a 100% da nota da segunda unidade. A avaliação será composta pela entrega técnica, escrita de um relatório e qualidade da documentação.

2. Objetivos

Desenvolver um pipeline de análise e visualização que integre:

- Reproduzir e avaliar o desempenho de modelos apresentados em um artigo científico;
- Implementar o estudo dentro de um **pipeline executável via Docker Compose**, utilizando a arquitetura de BI integrada.
- Aplicar técnicas de **pré-processamento, modelagem, avaliação e visualização** dos resultados obtidos.
- Documentar todas as etapas em um **relatório técnico** com descrição detalhada das etapas, metodologia e principais achados.

Para implementação do pipeline, as seguintes tarefas deverão ser executadas:

- Criar um fluxo de ingestão via **FastAPI**, salvando dados em **S3/MinIO**;
- Estruturar os dados no **Snowflake** (ou base local em **SQLite/PostgreSQL**);
- Processar e modelar dados com **Python e Jupyter Notebook**;
- Registrar experimentos no **MLFlow** (port 5000);
- Exibir resultados e predições em dashboards **ThingsBoard (8080)** e **Trendz (8888)**.

3. Avaliação

O projeto deverá ser entregue até o dia 03/12. A nota final será composta por entrega técnica, dashboard funcional, e relatório técnico descritivo, considerando também aspectos de organização, clareza e integração entre as camadas do pipeline.

Critério	Descrição	Peso
Integração entre camadas	Coerência e funcionamento do pipeline completo (coleta, tratamento, modelagem, visualização conforme Figura 1).	3.0
Relatório técnico (modelo)	Documento técnico explicando todo o pipeline: objetivos, arquitetura, tratamento de dados, modelo aplicado, resultados obtidos e conclusões. Deve conter capturas de tela do dashboard e gráficos que descrevem os resultados adequadamente.	2.0
Modelagem e Avaliação	Correção metodológica (baseada no paper original e sugestões de melhorias), ajuste de hiperparâmetros e interpretação dos resultados.	3.0

Visualizações e dashboards	Clareza, estética e relevância dos gráficos.	1.0
Organização e documentação (README, estrutura do repositório)	Clareza das instruções, padronização das pastas, uso de Docker Compose e versionamento no GitHub.	1.0

Importante! **O relatório deverá ser entregue em formato *.docx via Google Classroom com no mínimo 10 páginas. O relatório deve conter:**

1. Introdução e objetivos do trabalho.
2. Descrição da arquitetura e das ferramentas utilizadas.
3. Metodologia de tratamento e modelagem de dados.
4. Análises e resultados (incluindo gráficos e tabelas).
5. Dashboard e insights obtidos.
6. Conclusões e possíveis melhorias futuras.

Demais arquivos do projeto deverão ser entregues via repositório no GitHub.

4. Requisitos Técnicos

- As equipes (até **6 integrantes**) deverão escolher **um artigo** entre os sugeridos (seção 7) ou outro previamente aprovado pelo professor.
- O projeto deve implementar **ao menos um modelo preditivo** (classificação, regressão ou agrupamento) e apresentar **visualizações dos resultados**. Cada equipe poderá propor **melhorias**, como:
    - Alterações de algoritmo;
    - Ajuste de hiperparâmetros;
    - Inserção de novas métricas;
    - Ampliação de variáveis explicativas;
    - Estratégias de balanceamento ou *feature engineering*.
- A arquitetura mínima do projeto deverá conter os seguintes serviços (em Docker Compose):

Serviço	Função Principal
FastAPI	Interface de ingestão dos dados do dataset selecionado (CSV/JSON) e integração com S3.
MinIO ou AWS S3	Armazenamento de dados brutos e modelos

Snowflake	Estruturação de dados tratados
Jupyter Notebook	Ambiente de análise, limpeza e modelagem preditiva
MLFlow	Registro e versionamento dos modelos de ML
ThingsBoard/Trendz Analytics	Visualização dos dados e dashboards interativos

Fluxo geral:

1. O **FastAPI** recebe e armazena os dados no **S3/MinIO**.
2. Os dados são estruturados em Snowflake.
3. Jupyter Notebook lê da base estruturada, trata e treina um modelo.
4. O modelo é versionado no MLFlow e exportado novamente para o S3.
5. O dashboard (ThingsBoard/Trendz) consome os dados e mostra visualizações e insights.

5. Estrutura do Repositório

A estrutura de pastas para o repositório no Github deverá ser a seguinte:

```
/
├── docker-compose.yml     # Orquestração dos contêineres
├── jupyterlab/            # Ambiente de análise e exploração (Dockerfile e configs)
├── mlflow/                # Configuração e armazenamento de experimentos
├── fastapi/               # Camada de ingestão (API)
├── notebooks/             # Notebooks de tratamento, visualização e modelagem
├── trendz/                # Dashboards exportados
├── reports/               # Figuras com os plots dos resultados
├── README.md              # Descrição do projeto
└── LICENSE                # Licença
```

O arquivo de licença é opcional, mas caso seja definido, licenças como MIT e BSD são recomendadas. O arquivo `README.md` deverá conter as seguintes informações:

1. Nome e sobrenome dos membros do projeto e seus respectivos **usuários no GitHub (@fulano, @beltrano, @sicrano)**.
2. Nome da disciplina: **Aprendizado de Máquina - 2025.2**.
3. Nome da instituição de ensino: **CESAR School**.
4. Instruções detalhadas para levantar a infraestrutura, executar e visualizar o dashboard.

6. Itens Obrigatórios de Entrega

Cada grupo deverá entregar:

1. **Pipeline executável via Docker Compose**, com os serviços descritos na Seção 4.
2. **Relatório técnico em *.docx via Google Classroom.**
3. **Repositório público no GitHub** contendo:
    - Código fonte (FastAPI, Notebooks, etc.);
    - Docker Compose funcional;
    - README com instruções de execução;
    - O link do repositório público deverá constar no relatório final
4. **Dashboard online** (no ThingsBoard/Trendz).
5. **Apresentação oral** com demonstração do pipeline e discussão dos resultados.

7. Sugestões de Artigos para Reprodução

Cada grupo deverá escolher **um artigo para reprodução.** São apresentados alguns trabalhos como sugestão. No entanto, **as equipes poderão utilizar um artigo diferente da lista, desde que devidamente alinhado com o professor.**

Abaixo estão **propostas possíveis**, cobrindo técnicas de **agrupamento, classificação e regressão**:

7.1. Predicting congenital syphilis cases: A performance evaluation of different machine learning models (https://doi.org/10.1371/journal.pone.0276150)

7.2. Hybrid Machine Learning Models for Intrusion Detection in IoT: Leveraging a Real-World IoT (https://doi.org/10.48550/arXiv.2502.12382)

7.3. Comparative Effectiveness of Classification Algorithms in Predicting Diabetes (https://doi.org/10.1109/CICN63059.2024.10847398)

7.4. Soil Analysis for Crop Recommendation using Machine Learning Algorithms (https://doi.org/10.1109/ICSTSN61422.2024.10671119)

7.5. Enhancing Prognosis Accuracy for Ischemic Cardiovascular Disease Using K Nearest Neighbor Algorithm: A Robust Approach (https://doi.org/10.1109/ACCESS.2023.3312046)

7.6. Movie Recommender System Using K-Means Clustering and K-Nearest Neighbor (https://doi.org/10.1109/CONFLUENCE.2019.8776969)

Bom trabalho!