Projeto final – Bootcamp CDIA 2025

Eduardo Scafi

Curitiba, 01 de setembro de 2025.

Sumário

Contextualização

Este projeto foi desenvolvido como entrega final do Bootcamp de Ciência de Dados e IA, com foco em manutenção preditiva de máquinas industriais.

Problema proposto

Uma empresa do setor industrial contatou você para a criação de um sistema inteligente de manutenção preditiva das suas diferentes máquinas. Essa empresa forneceu um conjunto de dados contendo informações coletadas a partir de dispositivos IoT sensorizando atributos básicos de cada máquina.

O objetivo é criar um sistema capaz de identificar as falhas que venham a ocorrer, e se possível, qual foi o tipo da falha. Cada amostra no conjunto de dados é composta por 8 atributos que descrevem o comportamento de desgaste da máquina e do ambiente. Além dessas características, cada amostra é rotulada com uma das 5 possíveis classes de defeitos.

O sistema deverá ser capaz de, a partir de uma nova medição do dispositivo IoT (ou conjunto de medições), prever a classe do defeito e retornar a probabilidade associada. Além disso, a empresa espera que você extraia insights da operação e dos defeitos e gere visualizações de dados.

A correta identificação de padrões de falha pode reduzir custos e aumentar a confiabilidade, mas apresenta desafios devido à natureza dos dados, frequentemente desbalanceados.

- Objetivo: desenvolver modelos de machine learning para detecção de falhas.
- Importância: prever falhas reduz custos e aumenta a segurança operacional.
- Desafio principal: forte desbalanceamento, com predominância de registros sem falha.
- Estratégia: comparar abordagens de classificação (binária, multiclasse e multirrótulo).
- Foco: avaliar desempenho com métricas adequadas ao desbalanceamento.

Descrição dos Dados

Foram disponibilizados dois arquivos de dados:

- **Bootcamp_train.csv** - use o para explorar, treinar, avaliar seus modelos.

- **Bootcamp_test.csv** - esse arquivo não contém os labels, gere as predições usando seu modelo e use a seguinte API para ver o desempenho final do modelo.

Dicionário fornecido:

Campo	Descrição
id	Identificador das amostras do banco
id_produto	Identificador único do produto. Combinação da variável Tipo e um número de identificação
tipo	Tipo de produto/máquina (L/M/H)
temperatura_ar	Temperatura do ar no ambiente (K)
temperatura_processo	Temperatura do processo (K)
umidade_relativa	Umidade relativa do ar (%)
velocidade_rotacional	Velocidade rotacional da máquina em rotações por minutos (RPM)
torque	Torque da máquina em Nm
desgaste_da_ferramenta	Duração do uso da ferramenta em minutos
falha_maquina	Indica se houve falha na máquina (1) ou não (0)
FDF (Falha Desgaste Ferramenta)	Indica se houve falha por desgaste da ferramenta (1) ou não (0)
FDC (Falha Dissipacao Calor)	Indica se houve falha por dissipação de calor (1) ou não (0)
FP (Falha Potencia)	Indica se houve falha por potência (1) ou não (0)
FTE (Falha Tensao Excessiva)	Indica se houve falha por tensão excessiva (1) ou não (0)
FA (Falha Aleatoria)	Indica se houve falha aleatória (1) ou não (0)

O conjunto de dados fornecido é composto por variáveis numéricas, categóricas e lógicas, relacionadas a sensores, condições de funcionamento e características operacionais de máquinas. Além disso, o banco de dados de treino (**Bootcamp_train.csv**) inclui cinco colunas de indicadores de falhas.

A análise exploratória e tratamento prévio verificou a redundância ou irrelevância de colunas do dataset (id, id_produto), ao qual buscou-se manter somente variáveis informativas. Além disso, a análise exploratória mostrou grande predominância de registros/medições sem falhas, confirmando o desbalanceamento do problema. Esse aspecto do problema também fica claro na performance dos modelos implementados.

O conjunto de dados para teste fornecido (**Bootcamp_test.csv**), que não possui rótulos ou colunas de falhas, foi utilizado exclusivamente para inferência e observação do comportamento realista dos modelos.

Metodologia

Análise Exploratória de Dados (EDA)

Durante a EDA do problema, foram verificadas estatísticas descritivas das variáveis, distribuição de valores ausentes, identificação de colunas constantes e redundantes, além da avaliação do desbalanceamento entre classes de falhas.

Tópicos explorados:

- Identificação de outliers, valores ausentes e duplicados.
- Estudo de desbalanceamento entre variáveis.
- Análise univariada das variáveis de falha e controle.
- Padronização das variáveis de falha e controle.

Pré-processamento

Essa etapa foi dedicada a preparação os dados de forma adequada à modelagem. Essa etapa envolveu a remoção de colunas não informativas, tratamento das variáveis categóricas e divisão do conjunto treino e teste.

- Exclusão de colunas de identificação e alvos (ex.: id, id_produto, falhas).
- Codificação de variáveis categóricas por one-hot encoding.
- Divisão treino/teste estratificada tentando preservar proporção de falhas.
- Padronização de variáveis numéricas quando necessária (ex.: SVM).

Modelagem

A modelagem consistiu na construção de diferentes classificadores para lidar com o problema sob três abordagens: binária, multiclasse e multirrótulo. Essa escolha teve como objetivo comparar abordagens simplificadas e mais detalhadas, verificando suas limitações em dados desbalanceados.

- Binária → Support Vector Machine (SVM), para distinguir falha vs. sem falha.
- Multiclasse → Decision Tree, para classificar em “sem falha”, “1 falha” ou “2+ falhas”.
- Multirrótulo → Random Forest MultiOutput, para prever tipos específicos de falhas.

Avaliação

A avaliação foi realizada com métricas adaptadas a cada abordagem, de modo a não depender exclusivamente da acurácia, que pode ser enganosa em cenários desbalanceados. Foram selecionadas medidas convercionais (precision, recall, f1-score) e outras mais robustas (Balanced Accuracy, MCC, Jaccard Index, Hamming Loss), buscando uma análise mais realista do desempenho dos modelos.

- Geral: *Accuracy* (acertos globais), *Precision*, *Recall*, *F1-score*.
- Binário: *Balanced Accuracy* (média dos recalls), *Recall* da classe falha.
- Multiclasse: *Balanced Accuracy* (equilíbrio entre classes) e *MCC* (correlação predito vs. real).
- Multirrótulo: *Subset Accuracy* (acerto total por amostra), *Jaccard Index* (sobreposição de rótulos) e *Hamming Loss* (fração média de rótulos errados).

Resultados

A etapa de resultados teve como objetivo avaliar o desempenho dos três modelos propostos (binário, multiclasse e multirrótulo), frente ao desbalanceamento dos dados. Cada relatório de classificação foi analisado em conjunto com métricas complementares.

A análise mostrou que métricas tradicionais, como acurácia, podem ser enganosas, reforçando a importância de utilizar indicadores mais robustos, como *Balanced Accuracy*, *Recall* e *Jaccard Index*. Além disso, os modelos foram aplicados ao conjunto de teste (df_test), sem rótulos, para verificar seu comportamento em dados inéditos, sendo reportadas as distribuições de previsões como métrica prática de aplicação.

- **Binário (SVM)**: bom recall da classe falha (~0,72); baixa precision (~0,05); útil como filtro inicial.
- **Multiclasse (Decision Tree)**: acurácia global alta (~0,98), mas enganosa; Balanced Accuracy baixa (~0,40); ignora classes raras.
- **Multirrótulo (Random Forest)**: desempenho aceitável em falhas frequentes (FDC, FP), ruim em falhas raras (FDF, FA); Jaccard Index muito baixo (~0,01).
- **Aplicação no df_test**: previsões apresentadas como distribuições e salvas em arquivos .csv:

- Modelo binário gerou previsões dispostas no arquivo “**predicoes_modelo_binario.csv**”.
- Modelo Multiclasse gerou previsões dispostas no arquivo **“predicoes_modelo_multiclasse.csv”**.
- Modelo Multirrótolo gerou previsões dispostas no arquivo **“predicoes_modelo_multirrotolo.csv”.**

Tecnologias Utilizadas

O desenvolvimento foi conduzido em ambiente Python. O uso de pacotes como pandas, scikit-learn e seaborn garantiu praticidade na manipulação, modelagem e visualização. O Jupyter Notebook foi escolhido como ambiente principal.

- Linguagem: Python.
- Bibliotecas de dados: pandas (tratamento e organização de dados).
- Bibliotecas de modelagem: scikit-learn (classificadores, métricas e pré-processamento).
- Visualizações: matplotlib e seaborn (gráficos e matrizes de confusão).
- Ambiente: Jupyter Notebook, adequado para exploração iterativa.
- Modelos aplicados: Decision Tree, Random Forest MultiOutput, SVM.
- Pré-processamento: one-hot encoding, padronização de variáveis, train-test split estratificado.
- Aplicação prática: previsões no df_test, possibilitando visualizar o comportamento real dos modelos em novos dados.