Desafio de Forecasting Industrial

Pipeline End-to-End com Dados de Sensores

Curso de IA Avançado - Digiboard P&D

30 de novembro de 2025

1 Contexto do Problema

O objetivo deste desafio é desenvolver um pipeline de Machine Learning capaz de prever o consumo de energia para a próxima janela de operação, permitindo um planejamento mais eficiente e detecção de anomalias.

2 Requisitos do Pipeline de Dados

Antes da modelagem, é necessário transformar o dado bruto em um formato consumível pelos algoritmos.

2.1 Pré-processamento

- **Resampling:** A frequência de 30s possui muito ruído branco. Realize a reamostragem para médias de **15 minutos** ou **1 hora**.
- **Tratamento de Gaps:** Identifique falhas no sensor e utilize técnicas de imputação (interpolação linear ou repetição do último valor válido).

3 Análise Exploratória de Dados (EDA)

Antes de iniciar a modelagem, é fundamental entender o comportamento físico da máquina e os padrões da fábrica. Realize as seguintes atividades no seu Notebook:

3.1 Inspeção Visual e Estacionariedade

- Plote a série temporal completa para identificar *outliers* óbvios (ex: picos de tensão irreais ou zeros que indicam falha de sensor).
- Faça um "zoom"em uma semana específica para entender o padrão diário.
- Verifique a estacionariedade da série (média e variância constantes no tempo) visualmente ou via teste estatístico (ADF Test).

3.2 Decomposição da Série (STL)

Utilize a decomposição sazonal (biblioteca `statsmodels`) para separar a série em três componentes:

1. **Tendência:** A máquina está consumindo mais energia ao longo dos meses? (Desgaste?)
2. **Sazonalidade:** Existe um padrão claro que se repete a cada 24h ou a cada semana?
3. **Resíduo:** O que sobra é ruído aleatório ou contém informação?

3.3 Padrões de Calendário (Heatmaps)

Gere visualizações que cruzem variáveis temporais para entender a rotina da fábrica:

- **Heatmap (Dia da Semana × Hora do Dia):** Coloque as horas no eixo Y e os dias da semana no eixo X. As cores devem representar a intensidade do consumo.
- *Objetivo:* Identificar visualmente horários de almoço, trocas de turno e se a fábrica opera em regime reduzido nos fins de semana.

3.4 Autocorrelação (ACF e PACF)

Plote os gráficos de Autocorrelação (ACF) e Autocorrelação Parcial (PACF).

- Identifique até quantos "lags"(atrasos) o consumo atual tem forte correlação com o passado. Isso ajudará a definir o tamanho da janela de entrada dos modelos (Lookback window).

3.5 Feature Engineering (Obrigatório)

Para capturar a natureza temporal dos dados, implemente:

1. **Transformação Cíclica:** Converta as variáveis de tempo (Hora do dia, Dia da semana) para coordenadas de um círculo, garantindo a continuidade temporal (ex: 23h próxima de 00h).
$$x_{sin} = \sin\left(\frac{2\pi x}{\max(x)}\right), \quad x_{cos} = \cos\left(\frac{2\pi x}{\max(x)}\right) \tag{1}$$
2. **Lags (Defasagens):** Crie colunas representando o passado ($t-1$, $t-2$, $t-24h$).
3. **Rolling Statistics:** Média móvel e desvio padrão móvel para capturar a volatilidade recente.

4 Modelagem e Algoritmos Avançados

Nesta etapa, você deve comparar diferentes paradigmas de previsão. O objetivo é superar o *baseline* ingênuo.

4.1 Modelos Estatísticos Automatizados

Utilize o **AutoARIMA** (biblioteca `pmdarima`) como referência estatística robusta.

- O AutoARIMA busca automaticamente os melhores parâmetros (p, d, q) e sazonalidade.
- Lembrar que utiliza somente a serie temporal, então não é preciso adicionar nenhuma feature adicional, somente uma feature, por exemplo, potência, como um série temporal.
- *Nota:* Este modelo serve de "teto"para métodos clássicos e "chão"para métodos de Deep Learning.

4.2 Machine Learning Clássico

Implemente algoritmos baseados em árvores de gradiente (Gradient Boosting). Estes costumam ter excelente performance em dados tabulares com features bem construídas.

- **XGBoost** ou **LightGBM**.
- **CatBoost** (ótimo para lidar com features categóricas de turnos sem necessidade de One-Hot Encoding).

4.3 Deep Learning para Séries Temporais (SOTA)

Experimente arquiteturas de redes neurais modernas projetadas especificamente para sequências temporais. Escolha **pelo menos uma** das arquiteturas abaixo (sugestão: usar bibliotecas como *PyTorch Forecasting* ou *Darts*):

- **TFT (Temporal Fusion Transformer):** Modelo baseado em atenção que oferece interpretabilidade (mostra quais variáveis impactaram a previsão) e lida bem com múltiplos horizontes de tempo.
- **N-BEATS:** Arquitetura de Deep Learning pura baseada em blocos de tendência e sazonalidade, que frequentemente supera métodos estatísticos tradicionais.
- **PatchTST:** Modelo recente baseado em Transformers que segmenta a série temporal em "patches", obtendo resultados de estado da arte em long-term forecasting.
- **LSTM/GRU:** Redes recorrentes clássicas (baseline de Deep Learning).

5 Validação e Entregáveis

- **Validação:** Utilize **Time Series Split** (janela deslizante). Nunca utilize *Shuffle* ou *K-Fold* aleatório.
- **Métricas:** RMSE (Root Mean Squared Error) e MAE (Mean Absolute Error).
- **Entrega:** Um Jupyter Notebook contendo o pipeline, análise comparativa dos modelos (Tabela de Métricas) e a conclusão sobre qual modelo iria para produção na fábrica.

Anexo: Starter Kit (Python)

Utilize o código abaixo como ponto de partida para a engenharia de features.

```python
import pandas as pd
import numpy as np

def processar_features(df_raw, freq_resample='15min'):
    # 1. Resampling e Agregacao (Media e Maximo de pico)
    df_resampled = df_raw.resample(freq_resample).agg({
        'potencia_watts': ['mean', 'max']
    })
    # Ajustar nome das colunas
    df_resampled.columns = ['_'.join(col).strip() for col in df_resampled.columns.values]
    df_resampled.rename(columns={'potencia_watts_mean': 'y_target'}, inplace=True)

    # 2. Features Ciclicas (Hora do Dia)
    # Transforma 0-23h em representacao circular
    df_resampled['hora'] = df_resampled.index.hour
    df_resampled['hora_sin'] = np.sin(2 * np.pi * df_resampled['hora'] / 24)
    df_resampled['hora_cos'] = np.cos(2 * np.pi * df_resampled['hora'] / 24)

    # 3. Criacao de Lags (Autocorrelacao)
    # Ex: Valor de 1 passo atras e valor de 24h atras
    steps_24h = 4 * 24 # Considerando freq de 15min

    df_resampled['lag_t1'] = df_resampled['y_target'].shift(1)
    df_resampled['lag_24h'] = df_resampled['y_target'].shift(steps_24h)

    return df_resampled.dropna()
```

Listing 1: Feature Engineering Base