

# DEVOPSDAYS

## curitiba 2025



# Séries temporais



Aplicando estatística e machine learning em séries temporais

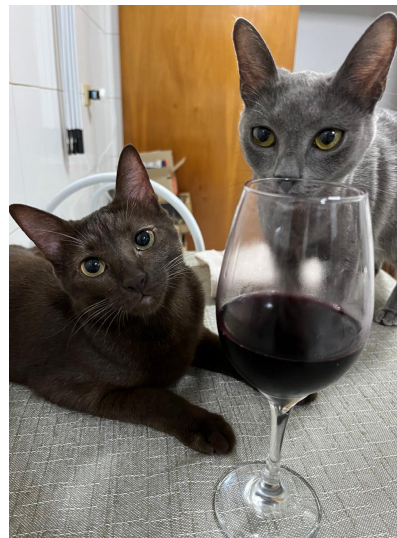
# Igor Estevan Jasinski

Engenhario de Observabilidade  **Sicredi**

SME - CNCF de observabilidade e OpenTelemetry



Pai de 5 gatos



# Agenda

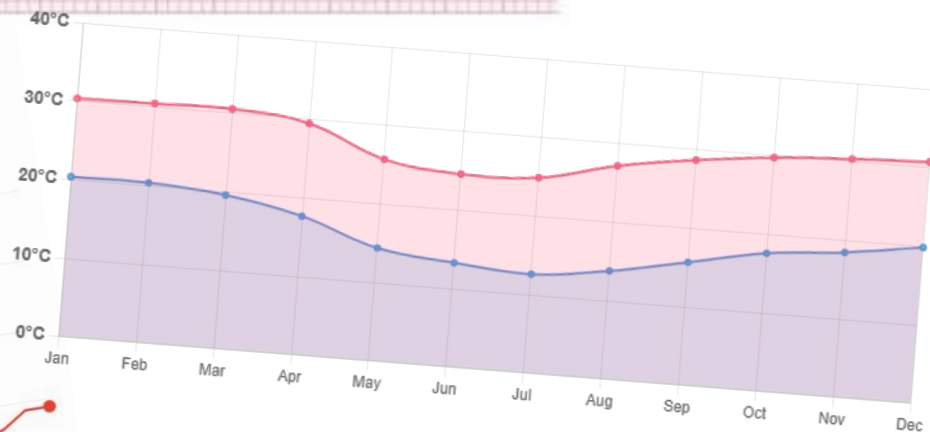
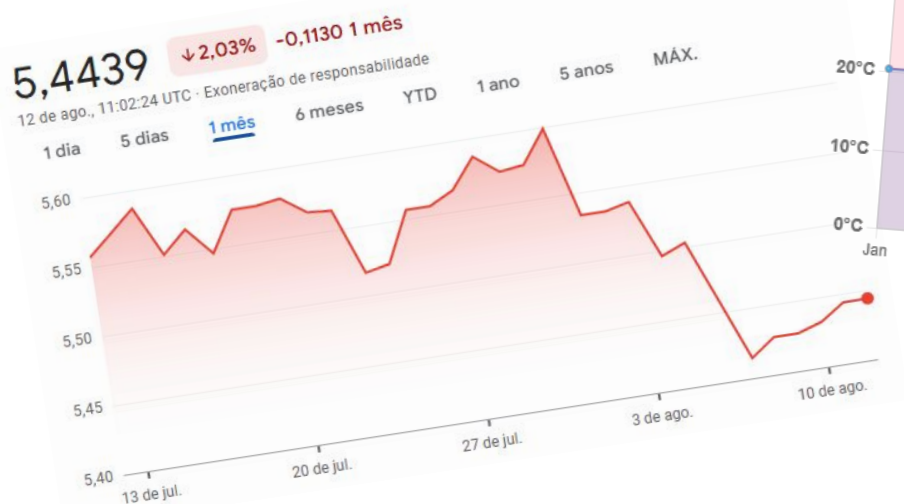
- Séries temporais
- Decomposição
- Forecast
- Detecção de anomalia
- Exemplo de Pipeline
- Conclusão

Latência



**O que é uma  
série temporal?**

**Séries temporais** referem-se a conjuntos de **dados** que são coletados, registrados ou observados ao longo do tempo e gravados em intervalos **regulares**.







# Séries temporais no Prometheus

# Séries temporais no Prometheus

Series	Value
go_goroutines{instance="cadvisor:8080", job="cadvisor"}	125 @1649828553.559

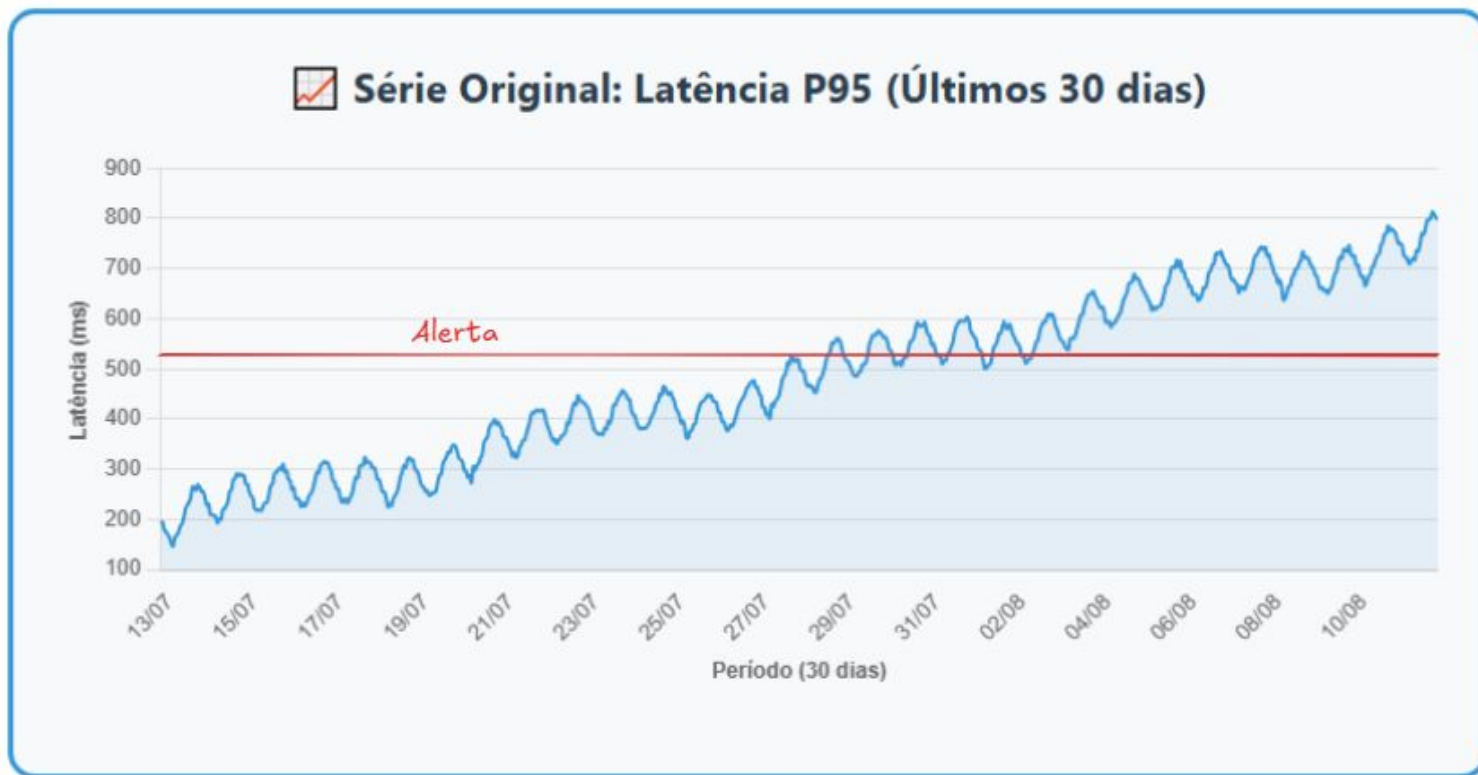
Labels: { \_\_name\_\_="go\_goroutines", instance="cadvisor:8080", job="cadvisor"}

Value: 125

Timestamp: 1649828553.559

```
# HELP http_request_duration_seconds Request duration histogram
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{le="0.1"} 2      # 2 requests ≤ 0.1s
http_request_duration_seconds_bucket{le="0.5"} 5      # 5 requests ≤ 0.5s
http_request_duration_seconds_bucket{le="1"} 8        # 8 requests ≤ 1s
http_request_duration_seconds_bucket{le="5"} 9        # 9 requests ≤ 5s
http_request_duration_seconds_bucket{le="+Inf"} 10     # All 10 requests
http_request_duration_seconds_sum 23.8                # Total seconds
http_request_duration_seconds_count 10                # Total requests
```

# Séries temporais no Prometheus



# Decomposição

# Decomposição

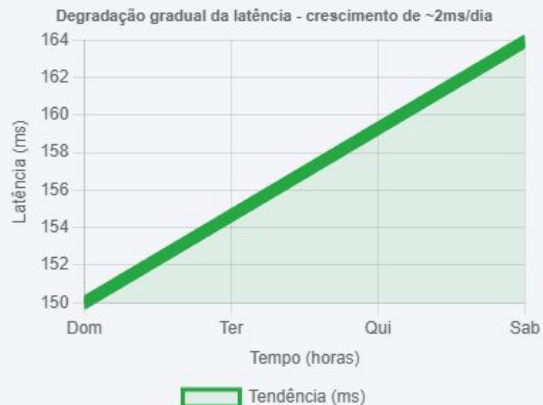


Latência Original da API



# Decomposição

## Tendência



**Tendência:** Movimento de longo prazo da série temporal. Representa a direção geral (crescimento, declínio ou estabilidade) da métrica ao longo do tempo, ignorando flutuações de curto prazo.

## Sazonalidade



**Sazonalidade:** Padrões regulares e previsíveis que se repetem em intervalos fixos (hora, dia, semana, mês). Reflete comportamento natural e esperado dos usuários e sistemas.

## Resíduo



**Resíduo:** Variações que não são explicadas nem por tendência nem por sazonalidade. Representa o componente aleatório onde encontramos anomalias verdadeiras.

# Algoritmos de decomposição

## STL (Seasonal-Trend decomposition using LoESS)

**Paradigma:** Non-parametric, LOESS-based decomposition

**Fundamento:** Aplicação iterativa de suavização LOESS para extração robusta de componentes temporais com handling automático de outliers.

**Aplicação Ótima:** Séries com sazonalidade única e estável, alta tolerância a outliers, infraestrutura geral

**Complexidade Computacional:**  $O(n \log n)$

Robusto a Outliers

$$Y(t) = T(t) + S(t) + R(t)$$

Iterative LOESS smoothing com robustness weights

## X13 X13 ARIMA-SEATS

**Paradigma:** Model-based, statistically optimal decomposition

**Fundamento:** Combinação de métodos X-11 com modelagem ARIMA para ajuste sazonal estatisticamente ótimo.

**Aplicação Ótima:** Séries econômicas/financeiras, análises oficiais, máxima precisão estatística

**Complexidade Computacional:**  $O(n^2)$  devido à estimação ARIMA

Precisão Estatística

ARIMA(p,d,q)(P,D,Q)<sub>s</sub> modeling + X-11 filters

Model-based seasonal adjustment

## BAT TBATS (Trigonometric, Box-Cox, ARMA, Trend, Seasonal)

**Paradigma:** State-space model com componentes trigonométricas

**Fundamento:** Modelagem de sazonalidade complexa através de funções trigonométricas com state-space representation para handling de periodicidades não-inteiras.

**Aplicação Ótima:** Sazonalidade não-inteira, múltiplas periodicidades, séries de alta frequência

**Complexidade Computacional:**  $O(n^3)$  para estimação de parâmetros

Flexibilidade Máxima

$$Y(t) = \ell_t + b_t + \sum s_{j,t} + \varepsilon_t$$

State-space com componentes trigonométricas

# Algoritmos de decomposição

```
from statsmodels.tsa.seasonal import STL  
  
decomp = STL(data, period=m).fit()
```

```
from statsmodels.tsa.x13 import x13_arma_analysis  
  
result = x13_arma_analysis(data)  
components = {  
    'trend': result.trend,  
    'seasonal': result.seasonal,  
    'cycle': result.cycle,  
    'irregular': result.irregular  
}
```



# Decomposição na observabilidade

## Análise de Tendência (Trend Component)

### **Deteção precoce de drift sistêmico e degradação gradual**

Permite deteção precoce de degradação gradual de performance, crescimento insustentável de recursos, ou mudanças estruturais no comportamento do sistema. Isolamento da componente de tendência revela patterns de longo prazo invisíveis em dados brutos, possibilitando intervenção proativa 6-12h antes de impacto crítico.

## Modelagem de Sazonalidade (Seasonal Component)

### **Thresholds adaptativos baseados em padrões cíclicos**

Fundamental para implementação de thresholds adaptativos que se ajustam dinamicamente aos padrões esperados, reduzindo drasticamente falsos positivos. Extração de padrões sazonais permite baselines que variam conforme contexto temporal, eliminando 85% dos alertas desnecessários durante variações normais.

## Análise de Resíduos (Residual Component)

### **Isolamento de variações não explicadas para deteção pura**

O resíduo isola variações não explicadas pelos padrões conhecidos, permitindo deteção de anomalias através de métodos estatísticos puros. Componente residual oferece signal-to-noise ratio 10x superior, pois variações previsíveis foram removidas, concentrando análise apenas em eventos genuinamente anômalos.

# Decomposição na observabilidade

Tendência

```
deriv(api_latency_trend[2h]) > 5
```

Sazonalidade

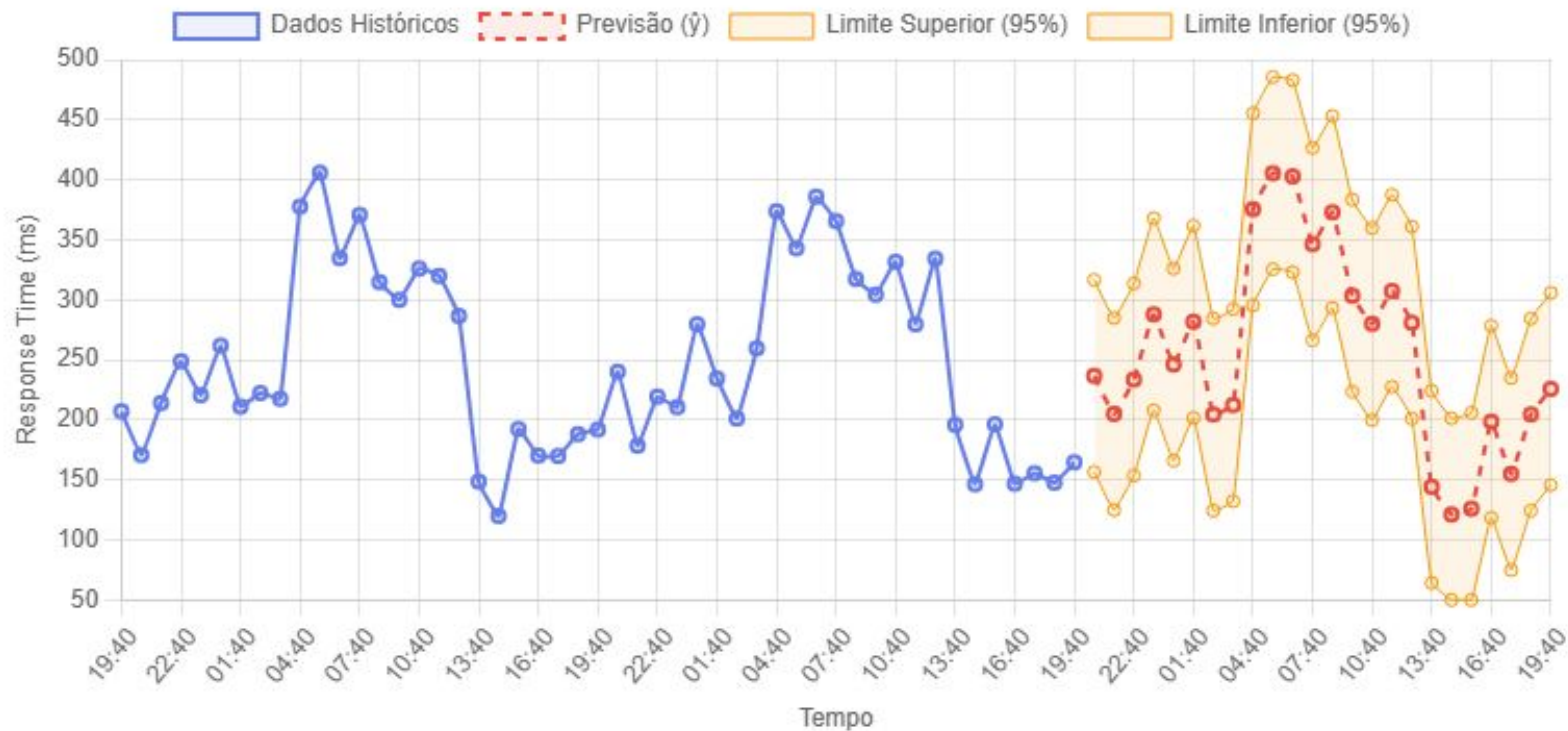
```
abs(api_latency - api_latency_seasonal) > api_latency_seasonal * 0.3
```

Resíduo

```
abs(api_latency_residual) > 2 * stddev_over_time(api_latency_residual[24h])
```

# Forecast

# Forecast



# Algoritmos de Forecast

## **ARIMA (AutoRegressive Integrated Moving Average)**

**Melhor para:** Séries estacionárias com padrões lineares

**Vantagens:** Rápido, interpretável, boa para tendências

**Desvantagens:** Dificuldade com sazonalidade complexa

$$\text{ARIMA}(p,d,q): (1-\varphi_1L-\dots-\varphi_pL^p)(1-L)^dX_t = (1+\theta_1L+\dots+\theta_qL^q)\varepsilon_t$$

## **Prophet (Facebook)**

**Melhor para:** Dados com forte sazonalidade e holidays

**Vantagens:** Robustos a outliers, missing data

**Desvantagens:** Maior complexidade computacional

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

trend + seasonality + holidays + noise

## **LSTM (Long Short-Term Memory)**

**Melhor para:** Padrões complexos e não-lineares

**Vantagens:** Captura dependências de longo prazo

**Desvantagens:** Requer mais dados e recursos

$$h_t = \text{LSTM}(x_t, h_{t-1})$$

Hidden state com memory gates

# Algoritmos de Forecast

```
my_model = Prophet(interval_width=0.95)

my_model.fit(df)

future_dates = my_model.make_future_dataframe(periods=horizon, freq='MS')
future_dates.head()
```

```
model = Sequential()
model.add(LSTM(units=64, input_shape=(sequence_length, 1)))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_train, y_train, epochs=10, batch_size=32)

predictions = model.predict(X_test)
```

# Forecast na observabilidade



## Observabilidade Proativa

Antecipe problemas antes que afetem usuários. Identifique tendências de degradação com até 24-48h de antecedência.



## Planejamento de Capacidade

Preveja necessidades de recursos baseado em padrões históricos e sazonalidade, otimizando custos de infraestrutura.



## Alertas Inteligentes

Reduza falsos positivos usando intervalos de confiança dinâmicos ao invés de thresholds estáticos.



## Manutenção Preditiva

Agende manutenções baseado em previsões de degradação de performance.

# Forecast na observabilidade

Alert com Intervalos de Confiança

```
cpu_usage_percent > cpu_forecast_upper_95 or cpu_usage_percent < cpu_forecast_lower_95
```

Requests acima do predito

```
requests_current > requests_forecast_yhat * 3
```

Vendas baixas vs predição

```
sales_current < sales_forecast_yhat * 0.8
```

Anomaly Score

```
kafka_lag - kafka_lag_yhat / kafka_lag_yhat_upper - kafka_lag_yhat_lower
```



# **Deteccção de anomalia**

# Detecção de anomalias - Pontuais

## ● Anomalias Pontuais (Point Anomalies)

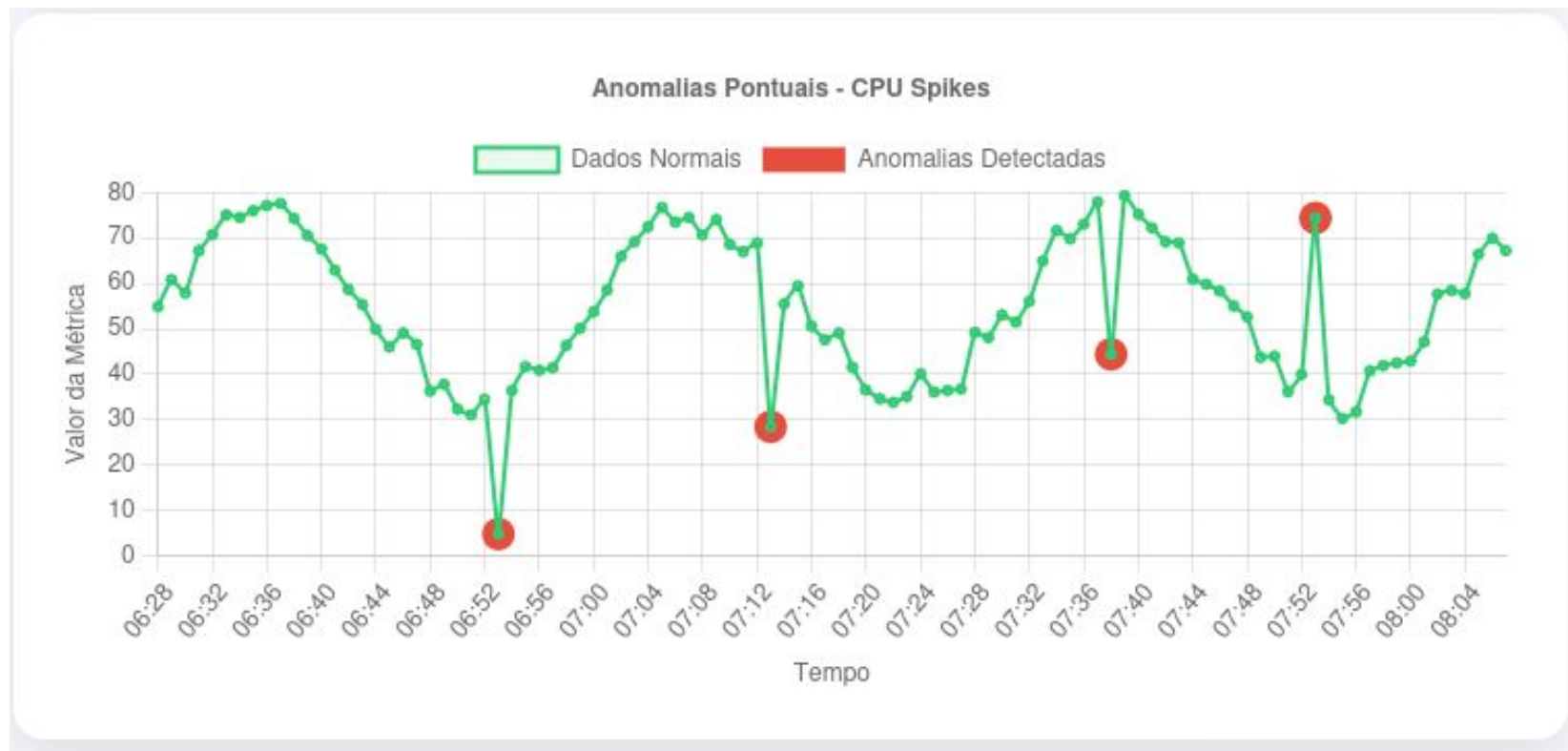
**Definição:** Observações individuais que apresentam desvio estatisticamente significativo em relação à distribuição esperada.

**Características:** Eventos isolados, tipicamente representando falhas instantâneas ou spikes de carga.

**Exemplos:** Spike abrupto de utilização de CPU, erro HTTP 500 isolado, latência de request anômala.

**Métodos de Detecção:** Z-score, Interquartile Range (IQR), Isolation Forest, One-Class SVM.

# Detecção de anomalias - Pontuais



# Detecção de anomalias - Coletivas

## ● Anomalias Coletivas (Collective Anomalies)

**Definição:** Sequências temporais onde observações individuais podem ser normais, mas o padrão coletivo apresenta comportamento anômalo.

**Características:** Cada ponto individual encontra-se dentro de parâmetros normais, mas a sequência temporal revela degradação ou instabilidade sistêmica.

**Exemplos:** Degradação gradual de performance, memory leak progressivo, oscilações de frequência anômala.

**Métodos de Detecção:** LSTM Autoencoders, Sequential Pattern Mining, Change Point Detection, Hidden Markov Models.

# Detecção de anomalias - Coletivas



# Detecção de anomalias - Contextuais

## ● Anomalias Contextuais (Contextual Anomalies)

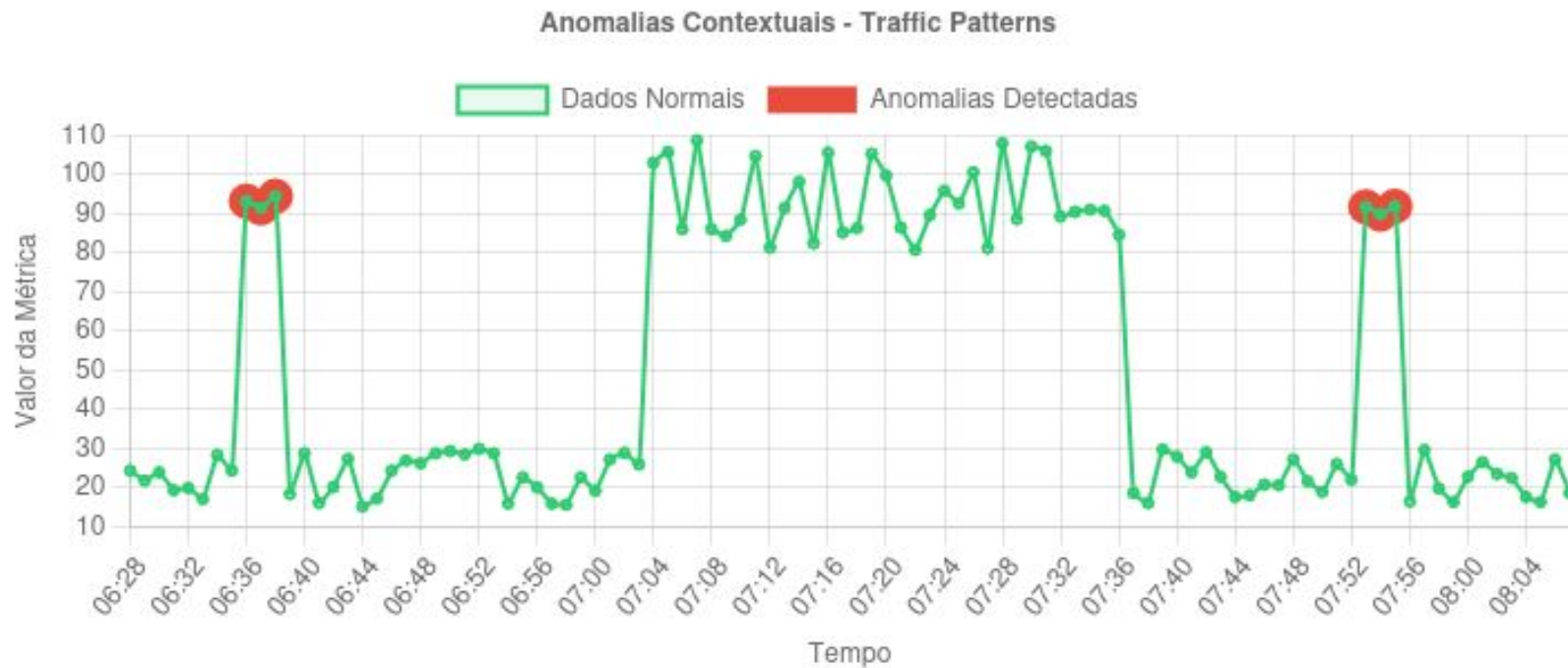
**Definição:** Observações que são anômalas apenas quando consideradas em contexto específico de features temporais, geográficas ou categóricas.

**Características:** Valores numericamente normais que se tornam anômalos devido ao contexto temporal, espacial ou comportamental.

**Exemplos:** Tráfego elevado durante período de baixa atividade, CPU alta em horário de manutenção programada.

**Métodos de Detecção:** Conditional Anomaly Detection, Time-Series Decomposition, Feature-based Classification.

# Detecção de anomalias - Contextuais





# Algoritmos de Detecção de anomalias

## IF Isolation Forest

**Paradigma:** Unsupervised, Tree-based Ensemble

**Fundamento:** Anomalias requerem menor número de partições para isolamento em estruturas hierárquicas.

**Aplicação Ótima:** Datasets multidimensionais, detecção em tempo real, alta escalabilidade

**Complexidade Computacional:**  $O(n \log n)$

Alta Performance

$$s(x,n) = 2^{(-E(h(x))/c(n))}$$

onde  $E(h(x))$  representa a profundidade média de isolamento

## LF Local Outlier Factor

**Paradigma:** Density-based Outlier Detection

**Fundamento:** Comparação de densidade local de pontos com densidade de vizinhança k-nearest.

**Aplicação Ótima:** Detecção de outliers em regiões de diferentes densidades

**Complexidade Computacional:**  $O(n^2)$

Precisão Contextual

$$LOF(x) = \sum (lrd(o)/lrd(x)) / |N(x)|$$

Local Reachability Density comparativa

## Z Statistical Methods

**Paradigma:** Métodos Estatísticos Clássicos

**Fundamento:** Detecção de desvios estatisticamente significativos assumindo distribuição normal.

**Aplicação Ótima:** Séries univariadas, implementação de baixa latência, baseline systems

**Complexidade Computacional:**  $O(n)$

Latência Mínima

$$Z\text{-score} = (x - \mu) / \sigma$$

Anomalia quando  $|Z| > \text{threshold}$   
(tipicamente 2.5 ou 3)



# Algoritmos de Detecção de anomalias

```
IsolationForest  
model = IsolationForest(contamination=0.05)  
  
model.fit(data)  
#retorna -1 para anomalias e 1 para normais  
data['Anomaly'] = model.predict(data)
```

# Detecção de anomalias na observabilidade

## Contexto Inteligente

**Considera sazonalidade e correlações**

100 req/s às 3AM é anômalo, mas normal às 14h. ML entende contexto.

## Redução Drástica de Ruído

**Muito menos alertas desnecessários**

Algoritmos aprendem padrões normais, alertando apenas em verdadeiras anomalias.

## Redução Significativa de MTTR

**Menos tempo perdido com falsos positivos**

Equipes focam em problemas reais ao invés de investigar alarmes falsos.

## Escalabilidade Automática

**Monitora milhares de métricas**

Impossível configurar thresholds manualmente para todas as métricas.

# Detecção de anomalias na observabilidade

Zscore

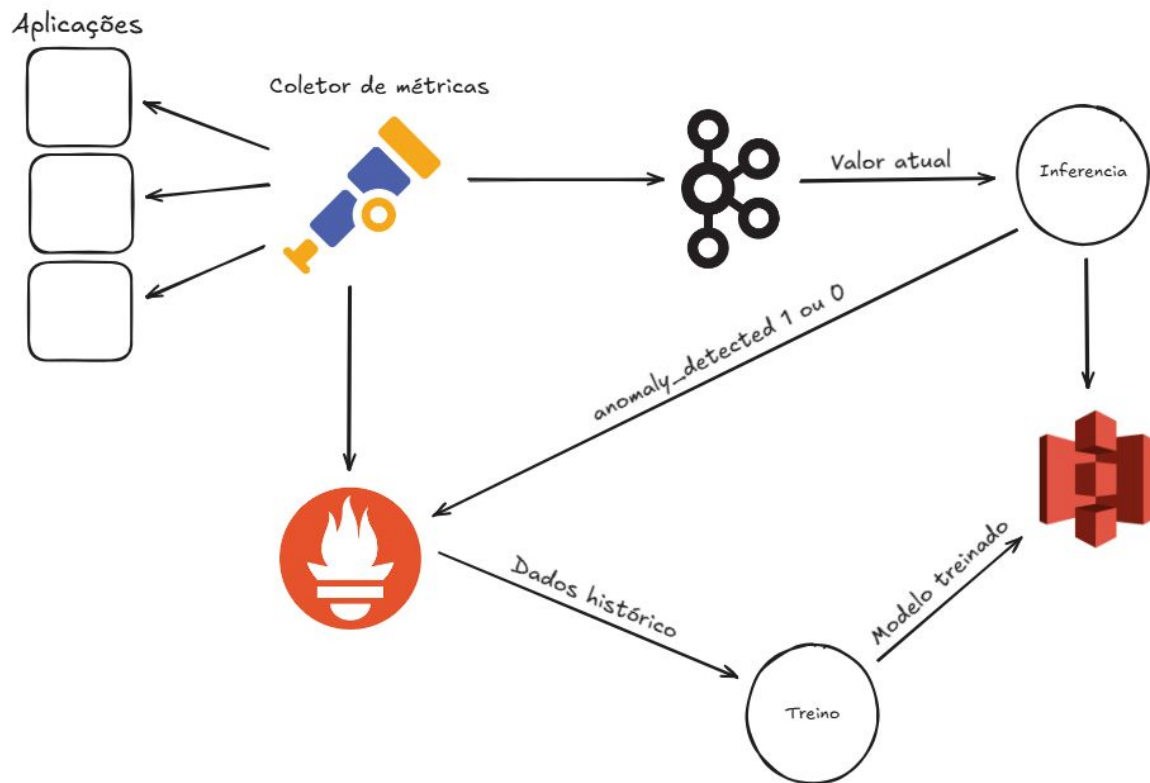
```
(rate(http_requests_total[1m]) - avg_over_time(rate(http_requests_total[1m])[15m:]))  
  / stddev_over_time(rate(http_requests_total[1m])[15m:]) > 3
```

Isolation anomaly score gauge

```
anomaly_score_metric = 1
```

# Exemplo

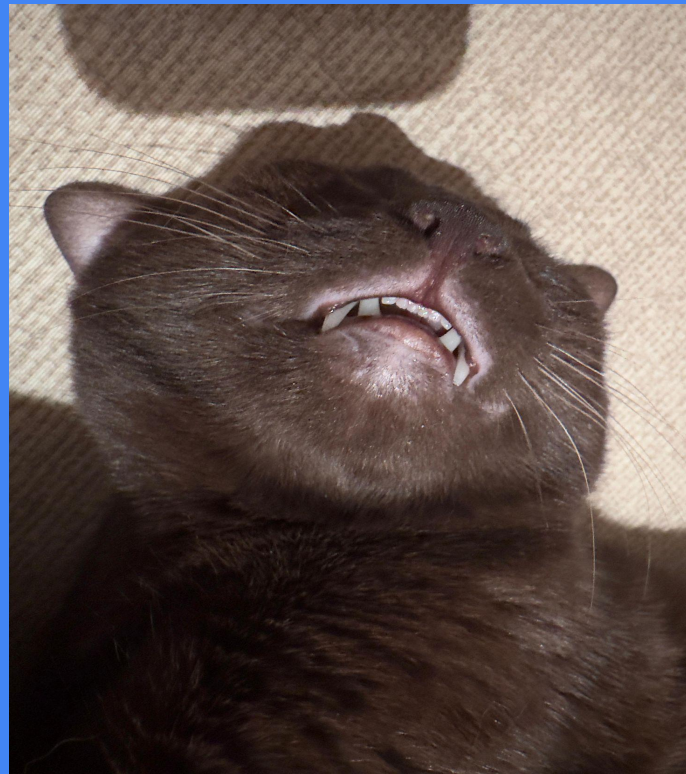
# Exemplo detecção de anomalia



# Conclusão



# Obrigado



# DEVOPSDAYS

## curitiba 2025

