UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

VÍTOR ÂNGELO RUSSI

DETECÇÃO DE ANOMALIAS APLICADA EM INTERNET DAS COISAS INDUSTRIAIS

Joinville
2022

VÍTOR ÂNGELO RUSSI

DETECÇÃO DE ANOMALIAS APLICADA EM INTERNET DAS COISAS INDUSTRIAIS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Gian Ricardo Berkenbrock

Joinville
2022

RESUMO

Num ambiente com a Internet das Coisas é gerado cada vez mais dados provenientes de sensores, e esses dados podem ser analisados para gerar informações úteis. Dados anômalos trazem informação consigo, porém, caso sejam ignorados, eles podem atrapalhar as análises, alterando as informações geradas e assim tomando conclusões imprecisas. Esse trabalho visa investigar técnicas de detecção de anomalias em séries temporais, para trazer mais informações sobre a séries estudadas, e aplicar algumas dessas técnicas utilizando Python para avaliar os resultados, integrando o processo todo utilizando a plataforma *ThingsBoard*. O modelo aplicado, ARIMA, trouxe resultados satisfatórios e a plataforma atendeu as funcionalidades exigidas.

Palavras-chave: Internet das Coisas. Dados. Anomalia.

ABSTRACT

Within an Internet of Things environment, more and more data is generated from sensors, and this data can be analyzed to generate useful information. Anomalous data bring information with them, but if ignored, they can disrupt the analysis, altering the information generated and thus taking inaccurate conclusions. This work aims to investigate anomaly detection techniques for time series, to bring more information about the studied series, and apply some of these techniques using Python to evaluate the results, integrating the whole process using the ThingsBoard platform. The applied model, ARIMA, brought satisfactory results and the platform met the required functionalities.

Keywords: Internet of Things. Data. Anomaly.

AGRADECIMENTOS

A minha família, que me acompanhou nessa jornada da vida, permitindo eu ser quem sou hoje.
Aos meus colegas, que me acompanharam durante a graduação e que enfrentamos diversos desafios juntos.
Ao meu orientador, que me aturou e incentivou durante todo o trabalho, que compartilhou inúmeros conhecimentos durante a graduação completa e que não desistiu de mim.
A minha noiva, que me ajudou nos meus momentos difíceis e que ficou do meu lado durante todo esse tempo.
E, por fim, a mim mesmo, que consegui superar essa etapa.

Any sufficiently advanced technology is indistinguishable from magic

Arthur C. Clarke, 1968

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API Interface de programação de aplicação

AR Autorregressivo

ARIMA Modelo Autorregressivo Integrado de Média Móvel

ARMA Modelo Autorregressivo de Média Móvel

BIC *Bayesian Information Criterion*

BLE *Bluetooth Low Energy*

DFD Diagrama de fluxo de dados

HTTP *Hypertext Transfer Protocol*

IHM Interface Homem-Máquina

IoT Internet das Coisas

IP Protocolo de Internet

MA Média Móvel

MQTT *Message Queuing Telemetry Transport*

OPC-UA *Open Platform Communications United Architecture*

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia intensificou a geração de dados, tornando cada vez mais importante que a sua aquisição e análise sejam feitas de formas mais adequadas, visando velocidade, precisão, memória utilizada, entre outros parâmetros. Os dados estão se tornando cada vez mais valiosos, fazendo com que o bom uso dele gere cada vez mais resultados positivos.

Num contexto de Internet das Coisas (IoT), é comum ter a necessidade de identificar anomalias num sistema com sensores que monitoram o sistema que eles estão inseridos (COOK; MISIRLI; FAN, 2020). As anomalias podem ser indicadores de falha de um sensor, corrompendo os dados adquiridos e, segundo Gaddam et al. (2020), o processamento de um dado corrompido compromete a desempenho geral de um sistema IoT, fazendo os dados lidos serem imprecisos e não confiáveis.

Segundo Blázquez-García et al. (2020), a detecção de anomalias se tornou um campo de estudos para muitos pesquisadores e é um dos principais constituintes de mineração de dados, em especial a análise do comportamento das anomalias ao longo do tempo.

A partir da detecção podem ser utilizadas métricas de avaliação para analisar a qualidade e confiabilidade dos dados. Algumas métricas, como exemplificado em Klima et al. (2020), podem ser o número de anomalias no sistema em funcionamento, ou o tempo que o sistema passa sem ocorrer uma anomalia.

Diante disso, este trabalho visa mostrar técnicas conhecidas de detecção de anomalias em séries temporais, utilizando métricas para avaliar a qualidade das séries temporais em estudo. Para isso, será utilizado uma plataforma de dados voltada para IoT e algumas bibliotecas conhecidas em Python para aplicar os métodos de detecção de anomalias.

1.1 OBJETIVO

Para resolver a problemática apresentada propõem-se os seguintes objetivos:

1.1.1 Objetivo Geral

Investigar técnicas de detecção de anomalias em séries temporais, de modo a prover informações descritivas da série temporal em estudo.

1.1.2 Objetivos Específicos

Para alcançar os resultados desejados são definidos os objetivos específicos como:

a) Estudo e definição de anomalias e suas métricas;
b) Estudo de técnicas de detecção de anomalias na literatura;
c) Implementar e testar as técnicas escolhidas; e
d) Analisar os resultados obtidos.

2 REVISÃO TEÓRICA

Neste capitulo serão apresentados os principais conceitos utilizados para a realização deste trabalho. Primeiro será apresentado o conceito de série temporal e anomalia, com seus tipos. Em seguida serão mostrados conceitos sobre a detecção de anomalias, juntamente de algumas técnicas para a detecção. Por fim, será apresentado a definição de qualidade e confiabilidade, com algumas métricas para avaliação.

2.1 SÉRIE TEMPORAL

Segundo Braei e Wagner (2020), séries temporais são conjuntos de amostragens feitas continuamente ao longo do tempo. As séries são normalmente amostradas com um intervalo de tempo fixo, podendo ser representadas por $T = (t_0^d, t_1^d, ..., t_n^d)$, onde d é a dimensão da série, com $d \in \mathbb{N}^+$ e $t \in \mathbb{N}$.

A principal diferença entre uma série temporal e outros conjuntos de dados é que os dados da série temporal não dependem apenas do valor amostrado d, eles dependendo também do momento n em que foram amostrados.

2.2 ANOMALIA

Uma anomalia é entendida como um dado que varia suficientemente do resto dos dados ao ponto de ser considerada por algum outro fator além do sistema atuante, podendo ser um erro ou um comportamento irregular do sistema. “Não existe uma definição padronizada para uma anomalia de um sensor”, adaptado de Gaddam et al. (2020), porém algumas definições de anomalia encontradas na literatura são:

a) *“[...] is commonly known to be an irregularity or a divergence in sensor behaviour during the process of cataloguing particular parameters or events when compared to its previous behaviour or readings.”* (GADDAM et al., 2020).
b) *“[...] is a data point that varies substantially from the remaining data points, as though it was produced by another technique.”* (ELMENSHAWY; HELMY, 2018).
c) *“[...] are typically defined in terms of deviation from some expected behavior. In general, the definitions of anomalies as “patterns in data that do not conform to a well-defined notion of normal behavior.” ”* (RAFIQUL et al.,

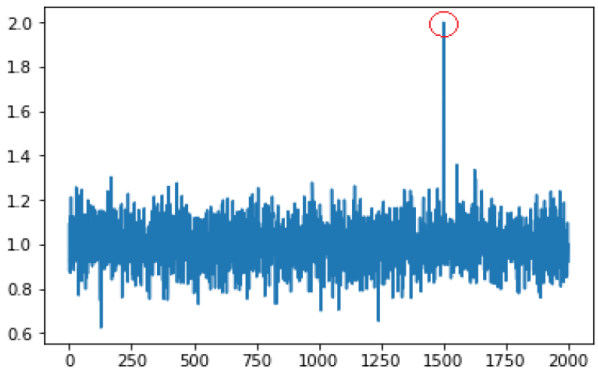
2017).

Uma anomalia podem ser caracterizada em três tipos: pontual, contextual e coletiva.

2.2.1 Anomalia Pontual

Segundo Braei e Wagner (2020), se o ponto varia significativamente do resto dos dados, ele pode ser considerado uma anomalia pontual. São as anomalias mais fáceis de detectar, por ter um comportamento muito distinto. A Figura 1 mostra um caso de anomalia pontual, onde o valor mostrado difere muito do restante.

Figura 1 – Uma anomalia global em um ruído Gaussiano aleatório.



Fonte: Cook, Misirli e Fan (2020).

2.2.2 Anomalia Contextual

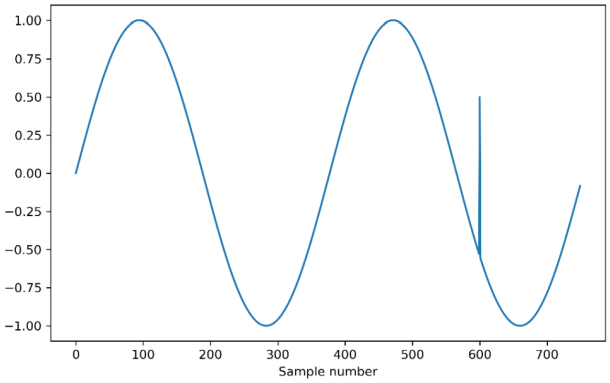
Um dado pode ser considerado uma anomalia contextual quando ele em um contexto não é anômalo, porém quando visto em outro contexto ele é, logo para poder saber se é uma anomalia ou não ele deve ser comparado com o contexto atual. A Figura 2 mostra um exemplo de anomalia contextual. O valor lido na amostra 600 por si só não é anômalo, visto que ele apareceu em outros momentos, porém dentro do contexto atual ele é.

Um exemplo prático seria o de um sensor medindo a temperatura ambiente e mostrando o valor de 30°C: para uma temperatura ambiente não é um valor estranho, porém se considerar que o sensor está no Canadá durante o inverno, o contexto mostra que o valor lido pode ser considerado uma anomalia.

2.2.3 Anomalia Coletiva

Alguns dados, quando analisados individualmente, podem parecer não ser anômalos, porém quando considerado o conjunto completo aonde ele se encontra,

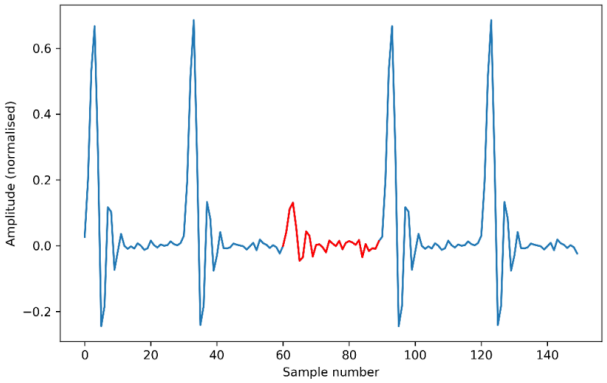
Figura 2 – Exemplo de uma anomalia contextual.



Fonte: Cook, Misirli e Fan (2020).

o conjunto todo pode ser considerado anômalo. A Figura 3 traz um exemplo de um conjunto de dados de um eletrocardiograma que, se vistos separadamente, são normais, porém o conjunto de dados aparecem no momento errado, se mostrando uma anomalia.

Figura 3 – Eletrocardiograma simulado com uma anomalia coletiva.



Fonte: Cook, Misirli e Fan (2020).

Um exemplo mostrado por ElMenshawy e Helmy (2018) é em transações do mercado de ações, onde uma transação entre duas partes é comum, porém muitas transações entre um pequeno grupo em um curto intervalo de tempo pode ser um indício de uma manipulação do mercado.

2.3 DETECÇÃO DE ANOMALIA

Segundo Rafiqul et al. (2017), a identificação de anormalidades é utilizado para identificar divergências do que é comum ou esperado. A detecção implica em diversas interpretações, como um erro na leitura, uma falha em um sensor, que geram leituras erradas, ou uma transação bancária de valor muito elevado, a qual não condiz com o padrão financeiro do cliente.

Existe uma grande variedade de algoritmos e abordagens diferentes que podem

ser utilizadas para detectar anomalias. As técnicas, embora possam estar em várias categorias, podem ser dividas nos seguintes grupos:

a) Estatístico/Probabilístico;
b) Distância e densidade;
c) Agrupamento (ou *clustering*);
d) Paramétrico e Reconstrução;
e) *Ensemble.*

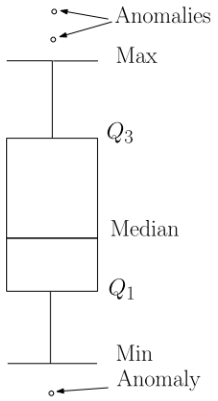
2.3.1 Estatístico/Probabilístico

Esses métodos utilizam dados históricos do comportamento usual do sistema. Quando um novo ponto é analisado, é calculado a probabilidade do ponto ser uma anomalia baseada num modelo previamente definido (SUN; MA, 2021).

Para isso ser verdade é necessário assumir uma condição: “instâncias normais de dados ocorrem em regiões de alta probabilidade de um modelo estocástico, enquanto anomalias ocorrem em regiões de baixa probabilidade do modelo estocástico”[1](CHANDOLA; BANERJEE; KUMAR, 2009, tradução nossa).

As técnicas estatísticas geram um modelo e então calculam se um novo dado pertence ou não a ele, onde que um dado com baixa probabilidade pode ser considerado uma anomalia. A Figura 4 mostra um modelo estatístico com alguns dados que se distanciam muito dos valores extremos, logo a probabilidade deles não serem gerados por ocasiões comuns é alta o suficiente para serem considerados anômalos.

Figura 4 – Um diagrama de caixa com dados anômalos.



Fonte: Chandola, Banerjee e Kumar (2009)

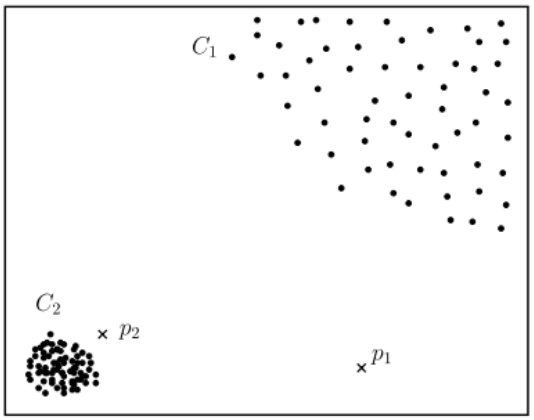
[1] No original: *“Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model.”*

2.3.2 Distância/densidade

Quando não se tem um parâmetro pré-definido, pode-se definir como um dado normal quando existem muitos outros dados próximos dele. Enquanto um dado, que está suficientemente distante do resto dos outros, indica que pode ter sido gerado por um mecanismo diferente e assim considerado anômalo (COOK; MISIRLI; FAN, 2020).

As técnicas baseadas na distância/densidade assumem, segundo Chandola, Banerjee e Kumar (2009), que dados normais ocorrem em uma vizinhança densa, enquanto anomalias estão longe dos seus vizinhos mais próximos. Um exemplo disso está na Figura 5, onde as vizinhanças C_1 e C_2 tem dados próximos mutualmente, enquanto os pontos p_1 e p_2 estão longes o bastante para serem considerados anômalos, mesmo sem ter um modelo conhecido previamente, apenas pela semelhança.

Figura 5 – Exemplo de anomalias detectáveis por análise da distância/densidade.



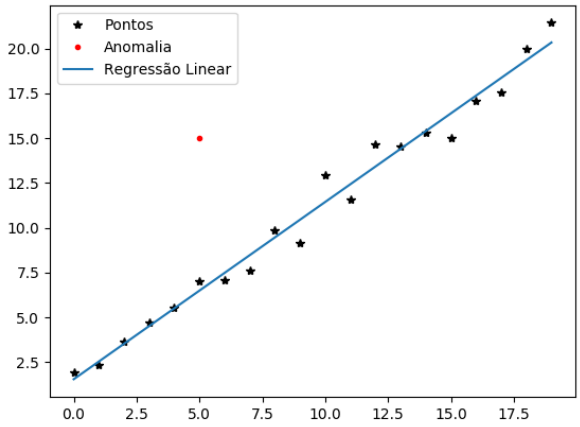
Fonte: Chandola, Banerjee e Kumar (2009)

2.3.3 Paramétrico e Reconstrução

Os métodos paramétricos e reconstrutivos procuram encontrar padrões nos dados para prever o comportamento futuro do sistema. Conforme Sebestyen et al. (2018), um modelo é construído a partir de uma técnica de identificação de sistemas e então é feita uma previsão dos próximos valores: se o erro em relação aos valores reais for significativo, a conclusão é o valor ser uma anomalia.

Um exemplo de identificação de sistemas é a regressão linear, que tenta estimar uma reta que melhor caracterize os dados fornecidos. A Figura 6 mostra uma regressão linear feita com os dados fornecidos, onde a maioria deles se encontram próximos da reta estimada, enquanto um valor, por estar muito distante, é considerado anômalo.

Figura 6 – Regressão linear.



Fonte: O autor

2.3.4 *Ensemble*

Aqui são utilizadas diferentes técnicas para observar o ponto, onde cada técnica tem um peso ou, segundo Cook, Misirli e Fan (2020), uma forma de votação é empregada sobre as saídas para cada método.

O *ensemble* pode ser feito por técnicas distintas para os mesmos dados, ou gerando os dados de múltiplas formas, sejam elas homogêneas ou heterogêneas. Na detecção de falhas em sensores, como descrito em Gaddam et al. (2020), uma abordagem homogênea utiliza um mesmo sensor ou sistema para fazer a leitura no mesmo ponto, enquanto uma abordagem heterogênea utiliza sensores distintos para ler o mesmo valor.

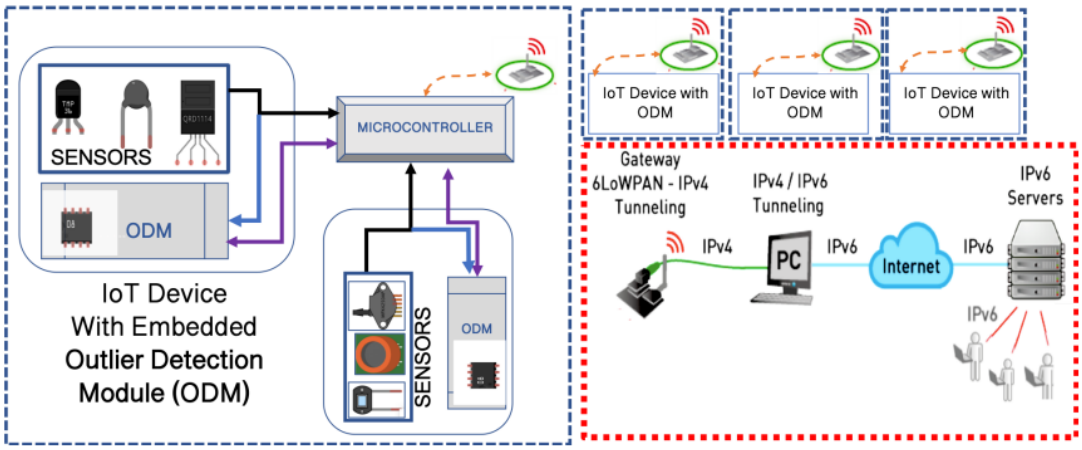
A Figura 7 mostra um exemplo de uma mistura de técnica homogênea com heterogênea:

a) Homogênea: o microcontrolador utiliza sensores distintos, que se baseiam em princípios físicos distintos uns dos outros para medir a temperatura;
b) Heterogênea: vários dispositivos idênticos estão ligados na mesma rede, fornecendo os dados para o servidor.

2.4 ALGORITMOS DE DETECÇÃO DE ANOMALIAS

A seguir serão apresentados alguns algoritmos de detecção de anomalia, classificados como modelos ARIMA. Os algoritmos fazem a detecção da anomalia tentando reconstruir o sinal e medindo o erro do sinal reconstruído com os dados originais.

Figura 7 – Modelo de detecção de anomalias baseado em *ensemble*.



Fonte: Gaddam et al. (2020)

2.4.1 Modelos ARIMA

Os modelos ARIMA se baseiam na autocorreção dos dados, isso é, como o valor de um dado influencia no valor de outro dado futuro (HYNDMAN; ATHANASOPOULOS, 2018). Eles podem ser separados em alguns modelos distintos, onde alguns são a combinação de outros, e são as características da série temporal que vão influenciar na eficiência do algoritmo.

2.4.1.1 Modelo Autorregressivo

Um modelo Autorregressivo(AR) se baseia na influência direta dos pontos passados no ponto atual, utilizando p dados passados para prever o dado atual. Segundo Hyndman e Athanasopoulos (2018), a equação geral de um modelo regressivo de ordem p pode ser escrito como:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t, \tag{1}$$

Na Equação 1, y_t é o próximo termo a ser calculado, c é uma constante, os termos $\phi_1, \ldots, \phi_n$ são os parâmetros que dão a forma ao modelo autorregressivo e ϵ_t é um ruído branco. O modelo é referenciado como $AR(p)$, onde p indica a ordem, ou seja, o número de dados passados a serem utilizados para definir o modelo autorregressivo.

2.4.2 Média Móvel

O modelo de Média Móvel(MA) não utiliza valores passados para prever o próximo valor, ele utiliza o erro das previsões passadas para tentar prever o próximo valor (HYNDMAN; ATHANASOPOULOS, 2018). Sua equação de ordem q pode ser escrita como:

$$y_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}, \tag{2}$$

Semelhante à Equação 1, a Equação 2 também conta com uma constante c e um conjunto de parâmetros que definem o modelo, com a diferença que as variáveis não são mais os valores passados e sim os erros passados. O modelo é referenciado como $MA(q)$, onde q indica a ordem, ou seja, o número de erros passados a serem utilizados para definir o modelo de média móvel.

2.4.3 Modelo Autorregressivo Integrado de Média Móvel (ARIMA)

O Modelo Autorregressivo Integrado de Média Móvel(ARIMA) é um modelo generalizado, utilizando uma AR e uma MA para a previsão dos dados, formando o Modelo Autorregressivo de Média Móvel (ARMA), com uma diferenciação. A diferenciação serve para séries não-estacionárias, uma vez que o modelo ARMA serve apenas para séries estacionárias (LEE; KIM, 2018). O modelo $ARMA(p, q)$ é dado por:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}, \quad (3)$$

Note que a Equação 3 é apenas uma combinação das Equações 1 e 2, apenas unindo as constantes em uma única variável. Esse modelo é referenciado por $ARMA(p, q)$, onde o p é a ordem do termo autorregressivo e o q é do termo de média móvel.

O modelo ARIMA adiciona uma diferenciação de grau d no modelo ARMA, para permitir a aplicação do modelo em uma série temporal não-estacionária, referenciado como $ARIMA(p, d, q)$. A diferenciação é aplicada da seguinte forma:

$$y'_t = y_t - y_{t-1} \quad (4)$$

A Equação 4 mostra uma diferenciação de grau um, eliminando uma tendência de grau um da série temporal. Na necessidade de um grau maior, pode-se diferenciar mais uma vez, como, por exemplo, uma diferenciação de segunda ordem:

$$y''_t = y'_t - y'_{t-1} \quad (5)$$

O modelo ARIMA, então, pode ser escrito como o modelo ARMA com a diferenciação, por exemplo, do primeiro grau:

$$y'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \cdots + \phi_p y'_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}, \quad (6)$$

2.5 QUALIDADE E CONFIABILIDADE

A qualidade e confiabilidade é dedicada ao estudo de falhas, como elas foram causadas, sua classificação e como ser prevenida. Moore et al. (2020 apud FRIES, 2006), define os requerimentos de confiabilidade afirmando que os dispositivos devem conseguir realizar a sua função:

a) que lhe foi atribuída;
b) sem falha;
c) sobre certas condições pré-definidas e;
d) durante um intervalo de tempo específico.

Os termos qualidade e confiabilidade têm significados semelhantes, tanto que são utilizados muitas vezes com o mesmo propósito, porém eles são diferentes. Qualidade é, como definido pela ISO 9000 (ISO, 2015 apud MOORE et al., 2020), é a habilidade de consistentemente prover produtos e serviços conforme os seus requisitos. Já a confiabilidade verifica o desempenho do produto ou serviço durante um intervalo de tempo, utilizando métricas como Falhas ao Longo do Tempo, Tempo Médio entre Falhas, entre outros.

2.5.1 Métricas de avaliação

A avaliação da qualidade e a confiabilidade pode ser medida utilizando certas métricas:

2.5.1.1 Falhas ao Longo do Tempo

Para medir a confiabilidade de um sistema pode-se medir a proporção entre falhas e o número total de operações numa janela de tempo (KLIMA et al., 2020):

$$FLT = \frac{N_{falhas}}{N_{total}}, \tag{7}$$

onde N_{falhas} é o número total de falhas e N_{total} é o número total de operações durante a janela de tempo escolhida. Quanto mais próximo de 1, mais falhas ocorreram, indicando um sistema menos confiável.

2.5.1.2 Tempo Médio entre Falhas

Fora de apenas uma janela de tempo, o Tempo Médio entre Falhas avalia a confiabilidade do sistema durante todo o seu funcionamento (KLIMA et al., 2020):

$$TMEF = \frac{\sum_{i=1}^{n} t_i - t_{i-1}}{n} \tag{8}$$

O cálculo consiste em medir o tempo entre duas falhas consecutivas ao longo de todos os dados e dividir pelo número de dados totais até o momento. Para obter a frequência média entre as falhas faz-se a inversa do Tempo Médio entre Falhas.

2.5.1.3 Disponibilidade

Disponibilidade pode ser definido como a probabilidade do sistema estar funcionando corretamente na hora que ele for requisitado (POKORNI, 2019), ou apenas

a relação entre o tempo que o sistema está funcionando em relação ao tempo total (KLIMA et al., 2020):

$$D = \frac{t_{online}}{t_{total}} = \frac{TMEF}{TMEF + TMPR}, \tag{9}$$

onde TMPR é o Tempo Médio para Reparo, ou seja, o tempo que leva para corrigir a falha. Se considerar falhas críticas, onde o sistema para de funcionar quando a falha ocorre, o TMPR é crucial para a disponibilidade.

3 MATERIAIS E MÉTODO

Esse capítulo descreve a metodologia empregada no estudo e apresenta as ferramentas utilizadas para a implementação das técnicas descritas na Seção 2.4. Além das implementações, os ajustes de integração também serão discutidos e apresentados. Na sequência, os conjuntos de dados utilizados e os parâmetros definidos para testar as técnicas escolhidas serão apresentados e comentados.

3.1 MÉTODO

Nessa Seção descrevem-se os passos para encontrar as anomalias nas séries em estudo utilizando os algoritmos probabilísticos. Para o estudo foram gerados 6 cenários e um cenário amostrado de um processo de solda por arco elétrico.

Para classificar o valor da série temporal como anomalia, os seguintes passos foram seguidos, ilustrado no Pseudocódigo 1:

1. Foi aplicado o método autoarima nas séries para determinar os termos p, d, q utilizados para reconstruir o sinal original. As condições de contorno utilizadas para a execução do método nos algoritmos foram os limites máximos nos termos p, d, q e o critério de informação utilizado para convergir o método.
2. Os sinais esperados foram gerados a partir dos termos determinados pelo autoarima.
3. Foram calculados os resíduos dos sinais originais e o gerados.
4. A partir dos resíduos foram calculadas as médias e o desvios padrão.
5. Se o desvio padrão do resíduo for maior que o limite definido, é classificado como uma anomalia.

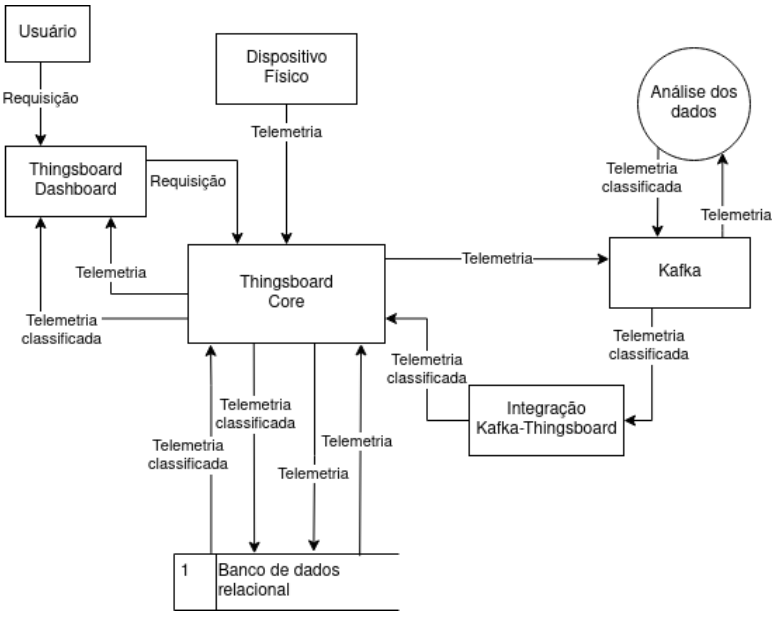
Algoritmo 1 Passos do método

```
1: while cenario.existe() do
2:     serie ← cenario.next()
3:     p, d, q ← auto_arima(serie, condicao_contorno)
4:     serie_esperada ← arima(p, d, q)
5:     residuos ← calc_residuos(serie, serie_esperada)
6:     media, sd ← calc_estatistica_desc(residuos)
7:     while temdadonasrie do
8:         if dadoForaDoIntervaloDe1Sd then
9:             DadoClassificadoComoAnomalia
10:        end if
11:    end while
12: end while
```

3.2 INTEGRAÇÃO DOS DADOS

A Figura 8 mostra o Diagrama de Fluxo de Dados (DFD) que descreve como foi feita a integração dos dados.

Figura 8 – Diagrama de Fluxo de Dados da integração de dados



Fonte: O autor

Do dispositivo físico, que vai fazer a aquisição do dado, o dado vai para o *Thingsboard Core* (ou só *Core*), a aplicação principal do sistema. Para fazer a classificação do dado, ele é enviado para o *Kafka*, que envia então para o processo que classifica o dado, que envia novamente para o *Core*. No processo que classifica, as seguintes bibliotecas são utilizadas: pdmarima e statsmodel.

O *Core* armazena os dados em um banco de dados relacional, tanto os dados que foram adquiridos quanto os que foram classificados. Para mostrar os dados para o

usuário, o *Core* interage com o *Thingsboard Dashboard* (ou só *Dashboard*), que recebe os dados crus e os classificados e mostra pro usuário, enquanto o mesmo pode fazer requisições como apagar os dados ou alterar parâmetros, que passam pelo *Dashboard* chegando no *Core*.

As versões dos principais softwares utilizados nesse trabalho foram:

- Thingsboard Community Edition 3.3.0;
- Zookeeper 3.4.13;
- Kafka 2.6.0;
- Python 3.8.10;
- pmdarima 1.8.5; e
- Statsmodels 0.13.0;

3.2.1 Exemplo de aplicação do DFD

Para exemplificar o fluxo e a visualização dos dados amostrados e classificados, será aplicada a lógica do DFD mostrado na Figura 8.

O sistema utilizado será o sistema de purificação de água *CHRIST Osmotron*®, utilizado em indústrias farmacêuticas (GARMAROODI et al., 2020). A Figura 9 mostra o sistema utilizado:

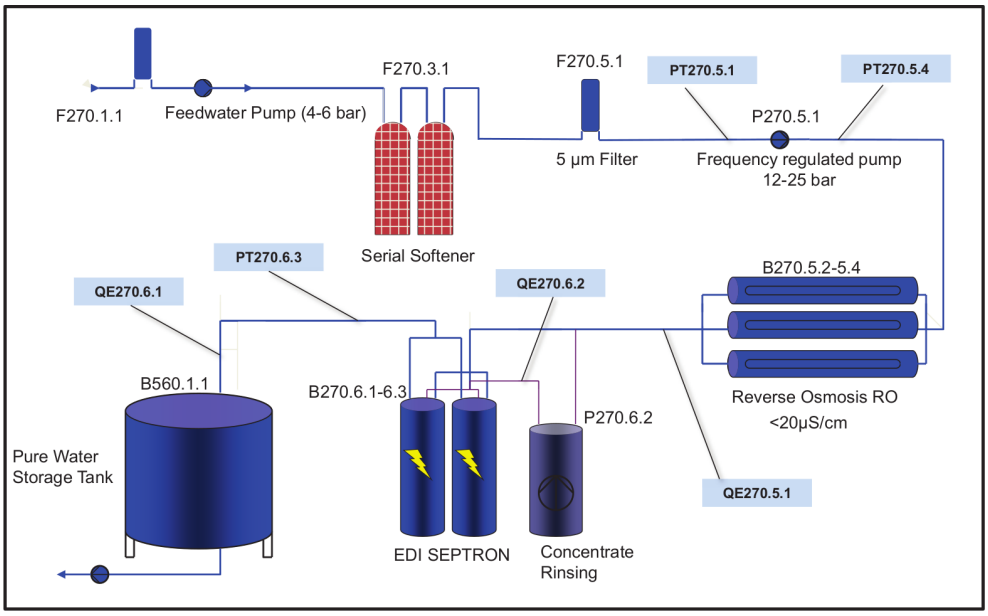
Figura 9 – O sistema de purificação de água exemplificado

Fonte: Garmaroodi et al. (2020)

Segundo Garmaroodi et al. (2020), o objetivo desse sistema é remover partículas, orgânicas e inorgânicas, e prevenir o crescimento de microrganismos. Cada etapa é muito importante para o controle de qualidade da água, então diversos sensores são utilizados ao longo do processo. A Figura 10 mostra o posicionamento

dos sensores utilizados na verificação do processo:

Figura 10 – Posicionamento dos sensores no interior do sistema



Fonte: Garmaroodi et al. (2020)

Os sensores são os itens destacados, iniciados em PT e QE, onde os iniciados em PT são sensores de pressão e os iniciados em QE são sensores de condutividade. Valores anômalos desses sensores, caso não tratados, podem trazer resultados incorretos sobre a qualidade da água, assim permitindo que água imprópria seja utilizada nos processos farmacêuticos. Nesse caso uma anomalia poderia ser um valor de pressão muito alto, indicando assim uma possível falha na bomba.

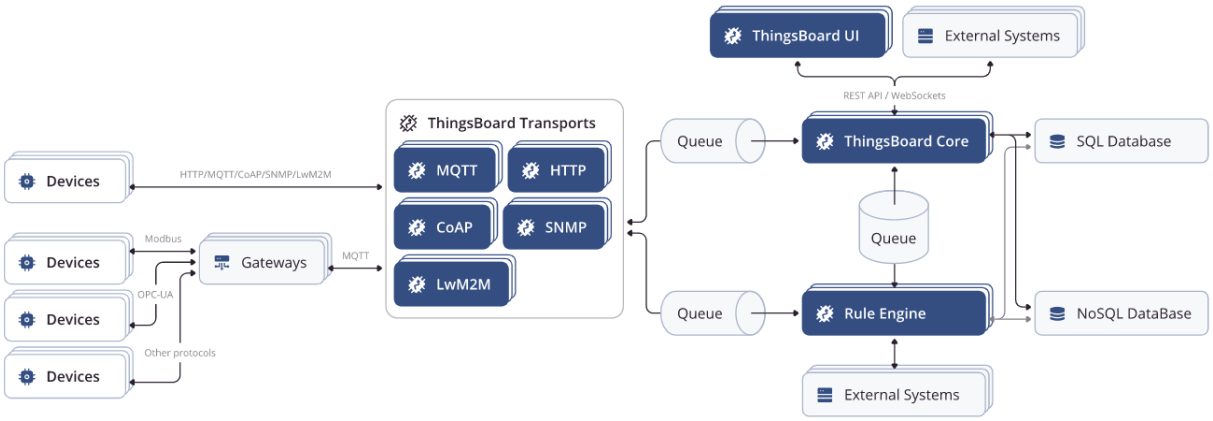
As amostragens dos sensores, então, podem ser enviadas para o *Thingsboard Core*, que irá armazenar e mostrar os dados lidos para o usuário. Os dados poderão ser enviados para análise, que irá retornar para o usuário a série classificada, evidenciando os dados anômalos caso existentes. A partir disso poderão ser feitas análises sobre a natureza do dado, se é uma imprecisão do sensor, um erro, ou uma falha do sistema completo, entre outras possíveis naturezas.

3.2.2 *Thingsboard*

O *Thingsboard* é uma plataforma para aquisição, visualização, processamento e gerenciamento de dados e dispositivos. Ele permite uma conexão entre diversos dispositivos físicos através de protocolos de rede industriais da camada de aplicação, como o *Hypertext Transfer Protocol* (HTTP), *Message Queuing Telemetry Transport* (MQTT) e o *Constrained Application Protocol* (CoAP) (THINGSBOARD, 2021).

A Figura 11 mostra os principais componentes e as suas respectivas interfaces:

Figura 11 – Diagrama mostrando as funcionalidades do *Thingsboard*

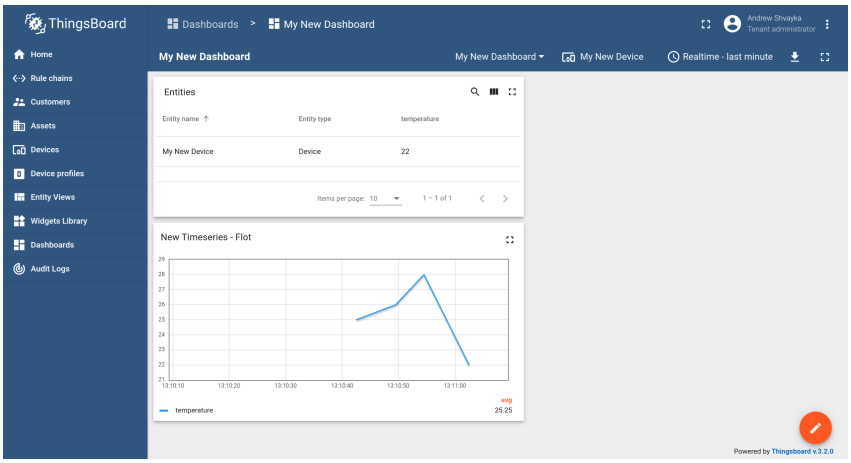


Fonte: ThingsBoard (2021)

3.2.2.1 Dashboards

O *Dashboard* é a parte do Thingsboard que faz a Interface Homem-Máquina (IHM), contando com gráficos, botões, cartões, entrada de dados e links para outros *Dashboards*. Os componentes do *Dashboard*, os *Widgets*, podem ser personalizados, permitindo uma interface mais dinâmica e intuitiva. A Figura 12 mostra um exemplo de *Dashboard*:

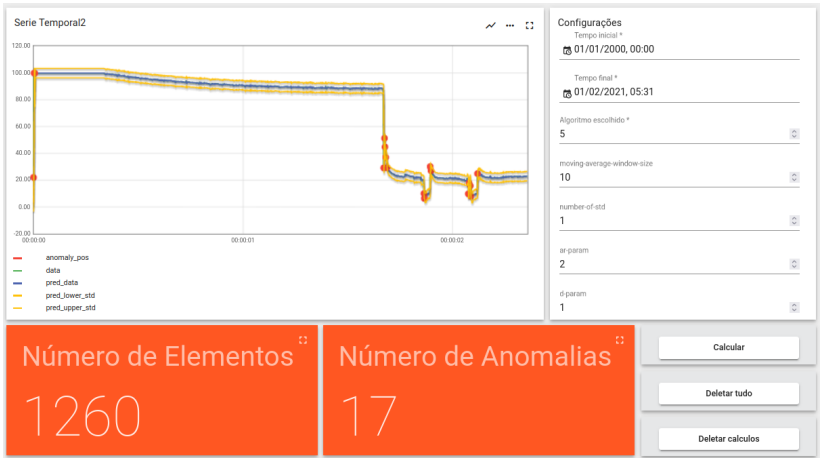
Figura 12 – Diagrama mostrando as opções de conexão com o Thingsboard



Fonte: ThingsBoard (2021)

O *Dashboard* desenvolvido para a realização desse trabalho está representado na Figura 13. Nele é possível ver a série temporal a ser analisada, permite a escolha do algoritmo e a personalização dos parâmetros. Após o cálculo, é mostrado o número de elementos e o número de anomalias encontradas, mostrando graficamente onde estão as anomalias.

Figura 13 – Dashboard desenvolvido

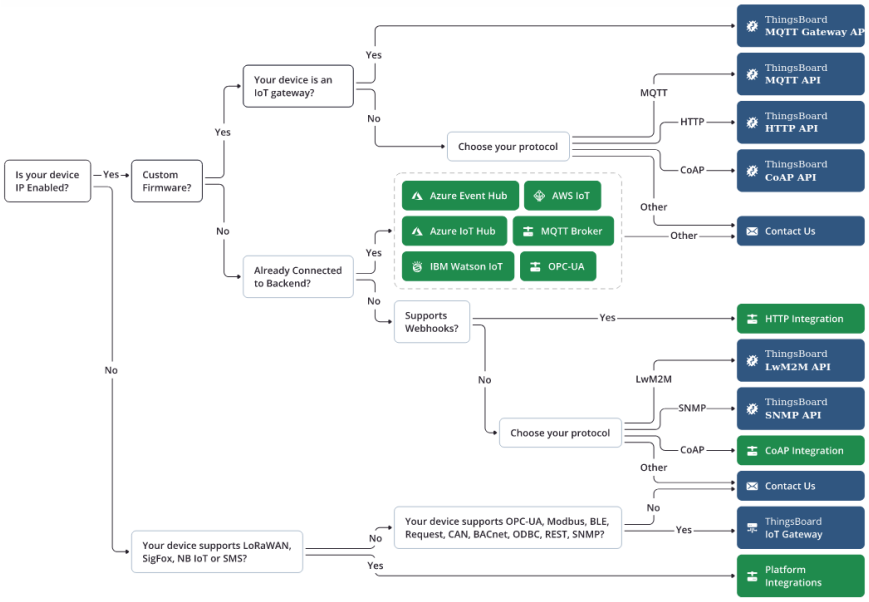


Fonte: Autor

3.2.2.2 Devices

Os *Devices* são a representação no *Thingsboard* dos dispositivos reais, que fazem o papel de sensores, atuadores, *gateways*, entre outros. A Figura 14 mostra quais as possibilidades de integração, para fazer o recebimento, processamento e armazenamento dos dados.

Figura 14 – Diagrama mostrando as opções de conexão com o Thingsboard



Fonte: ThingsBoard (2021)

Na Figura 14 estão ilustradas as formas de integração dos dados, podendo utilizar a Interface de Programação de Aplicação (API) do *Thingsboard* diretamente se tiver suporte para o protocolo desejado ou fazer uma integração caso não tenha suporte.

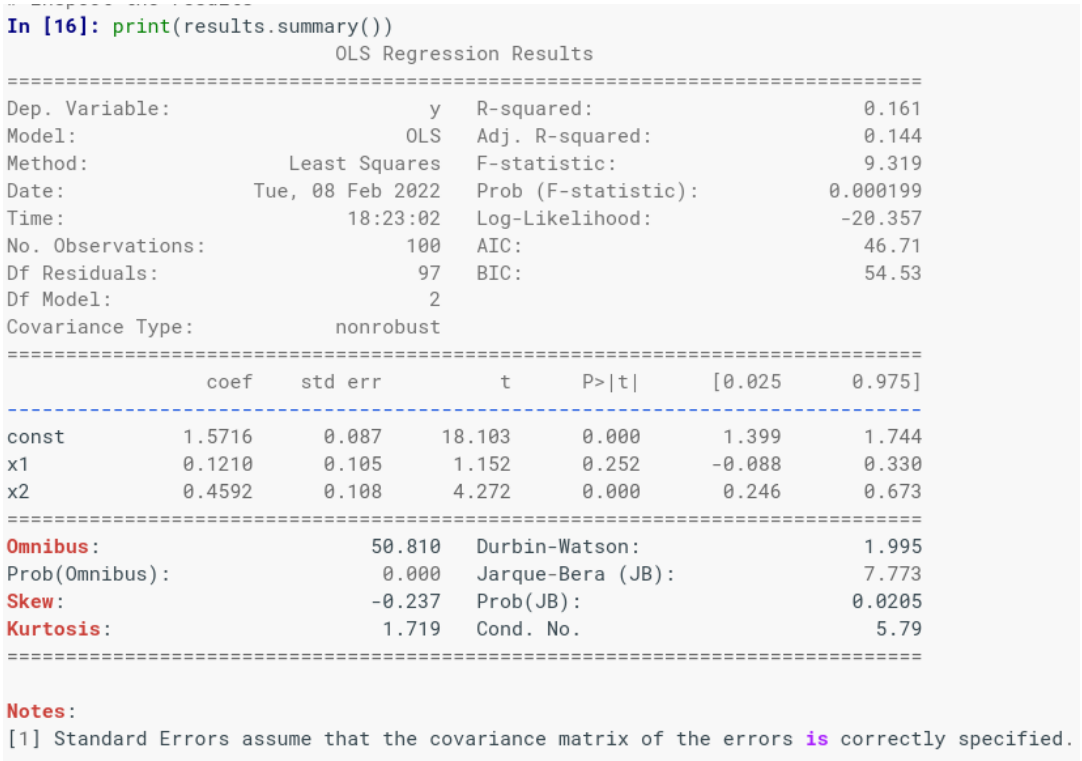
3.2.3 *Apache Kafka*

Esse sistema é um ambiente para transmissão de eventos, que contém uma alta taxa de transferência, escalável e que armazena os dados permanentemente em um sistema distribuído, durável e tolerante a falhas. Ele tem uma variação de APIs e bibliotecas para realizar a integração. A sua configuração é gerenciada pelo Zookeeper (GARG, 2013).

3.2.4 *Statsmodel*

Essa é uma biblioteca que contém uma extensiva lista de modelos estatísticos e testes estatísticos. Ela contém uma lista de resultado extensa, e os resultados são testados em outros pacotes estatísticos existentes, garantindo assim a veracidade dos resultados (SEABOLD; PERKTOLD, 2010). Um exemplo resumido de um resultado está descrito na Figura 15.

Figura 15 – Sumário do resultado de um modelo de regressão



```
In [16]: print(results.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.161
Model:                            OLS   Adj. R-squared:                  0.144
Method:                 Least Squares   F-statistic:                     9.319
Date:                Tue, 08 Feb 2022   Prob (F-statistic):           0.000199
Time:                        18:23:02   Log-Likelihood:                -20.357
No. Observations:                 100   AIC:                             46.71
Df Residuals:                      97   BIC:                             54.53
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.5716      0.087     18.103      0.000       1.399       1.744
x1             0.1210      0.105      1.152      0.252      -0.088       0.330
x2             0.4592      0.108      4.272      0.000       0.246       0.673
==============================================================================
Omnibus:                       50.810   Durbin-Watson:                   1.995
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                7.773
Skew:                          -0.237   Prob(JB):                       0.0205
Kurtosis:                       1.719   Cond. No.                         5.79
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Fonte: Seabold e Perktold (2010)

A Figura 15 mostra um sumário de qual modelo foi utilizado, alguns parâmetros de como foi executado o modelo (que no caso foi uma regressão) e alguns medidores.

No sumário apresenta-se o R-quadrado, sendo ele um medidor estatístico do quão próximo um conjunto de dados está da curva de regressão gerada.

3.2.5 *pmdarima*

A biblioteca *pmdarima*, (AL., 2017–) através do método *auto_arima*, é a responsável pela determinação do modelo ARIMA. Esse método testa vários valores de termos p, q e d (explicados na Seção 2.4.1), criando diversos modelos visando minimizar o BIC. Um exemplo dessa criação de modelos está descrita a seguir:

```
>>> pmdarima.auto_arima(timeserie, seasonal=False)
    Performing stepwise search to minimize bic
 ARIMA(0,1,0)(0,0,0)[0]             : BIC=-254.354, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0]             : BIC=-546.347, Time=0.04 sec
 ARIMA(0,1,1)(0,0,0)[0]             : BIC=-788.512, Time=0.05 sec
 ARIMA(1,1,1)(0,0,0)[0]             : BIC=-783.933, Time=0.06 sec
 ARIMA(0,1,2)(0,0,0)[0]             : BIC=-784.055, Time=0.06 sec
 ARIMA(1,1,2)(0,0,0)[0]             : BIC=-776.596, Time=0.07 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : BIC=-781.605, Time=0.19 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 0.510 seconds
```

3.2.6 Bibliotecas auxiliares

Segundo Reback et al. (2020), a biblioteca Pandas é de código aberto construída para análise e manipulação de dados, desenvolvida em Python. Algumas de suas características são:

- Otimizado para desempenho, com partes em C ou Cython;
- Ferramentas para ler e escrever em diversos formatos, como arquivos de texto, bancos de dados relacionais, entre outros;
- Funcionalidades para séries temporais, com geração de dados, conversão para frequência, médias móveis, atrasos, e outras funcionalidades.

A biblioteca NumPy é de código aberto provê objetos e operações de execução rápida para trabalhar com vetores e matrizes (HARRIS et al., 2020), desenvolvida em Python. Ela tem diversas implementações em C, conseguindo assim trazer a velocidade de execução do C para o Python, assim como integrações com múltiplas bibliotecas existentes.

3.3 INTEGRAÇÃO THINGSBOARD E KAFKA

A versão utilizada do *Thingsboard* não tem suporte ao envio de mensagem de um cliente *Kafka* para um *Device*, então foi necessário fazer a integração por outro meio. Foi criado um conector que consume as mensagens de um tópico Kafka e envia

as mensagens para um *Device* via *HTTP*, como mostra o Algoritmo 2:

Algoritmo 2 Lógica de integração Thingsboard/Kafka

```
1: while True do
2:     if kafkaTopic.length > 0 then
         msg = kafkaTopic.consumeMsg()
         http.post(msg, URL)
3:     end if
         delay(INTERVAL)
4: end while
```

4 RESULTADOS E DISCUSSÕES

Esse capítulo descreve a metodologia empregada no estudo e apresenta as ferramentas utilizadas para a implementação das técnicas descritas na seção 2.4. Além das implementações, os ajustes de integração também serão discutidos e apresentados. Na sequência, os conjuntos de dados utilizados e os parâmetros definidos para testar as técnicas escolhidas serão apresentados e comentados.

4.1 CENÁRIOS DE ESTUDO

As séries temporais utilizadas no estudo são divididas em dois grupos: dados simulados e dados medidos. Para os dados simulados foram escolhidos algumas funções conhecidas, que serão explicadas em 4.3.

4.2 PARÂMETROS UTILIZADOS

Esta seção descreve os parâmetros utilizados nos algoritmos probabilísticos: AR, MA, ARMA e ARIMA e as limitações.

Na documentação da biblioteca *pmdarima* tem algumas boas práticas para melhorar a geração de modelos. Para evitar o excesso de treinamento, foi definido como 10 o valor máximo de p e q e 3 para d. O critério de informação utilizado para convergir a geração dos modelos foi o BIC. O desvio padrão utilizado para definir como anomalia foi de 1.

A série temporal medida foi separada em várias faixas para facilitar a visualização da aplicação do algoritmo. O motivo da escolha de cada faixa será explicado em 4.3.7.

4.3 CENÁRIOS

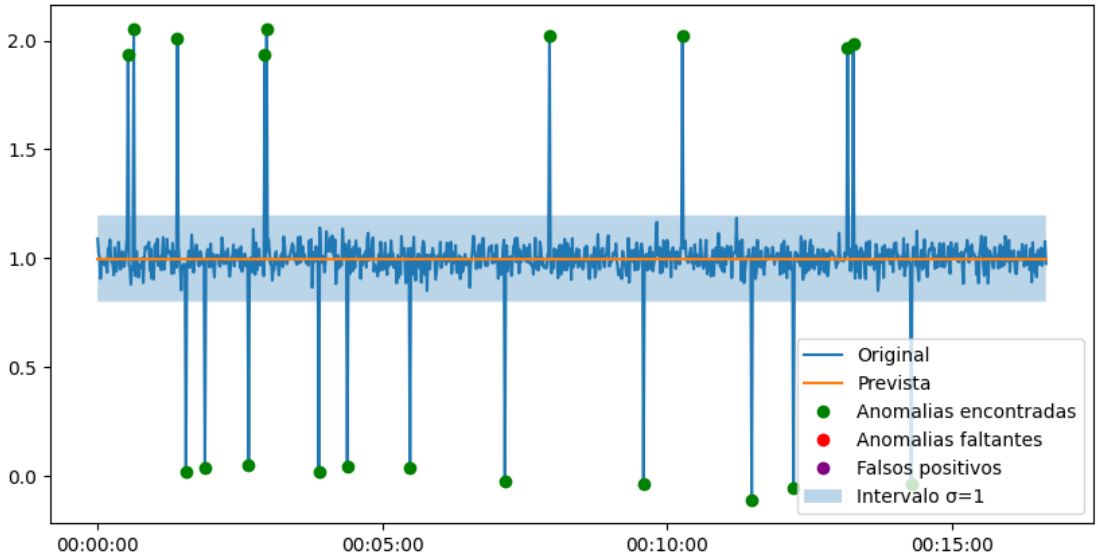
Nessa seção são descritos os cenários utilizados para avaliar os algoritmos. Todas as séries geradas têm 1000 pontos espaçados em 1 segundo, totalizando uma variação de tempo de 16 minutos e 40 segundos, contendo 20 anomalias pontuais de amplitude 1 e ruido gaussiano branco de amplitude 0,05.

A série temporal medida foi amostrada durante aproximadamente 26 segundos com uma taxa de amostragem de 5000 Hz, gerando um total de 118000 pontos espaçados em 20 microssegundos.

4.3.1 Cenário 1

Neste cenário (Figura 16), gera-se uma série temporal com valor constante em 1, com as características comuns. Esse comportamento é comum em casos onde o valor amostrado não deveria variar com o tempo, como a temperatura de uma sala de metrologia que deve ser constante, ou a parte constante da curva de aquecimento de um forno.

Figura 16 – Resultados do Cenário 1



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(0,0,0). Todas as vinte anomalias foram classificadas corretamente, sem falsos positivo.

4.3.2 Cenário 2

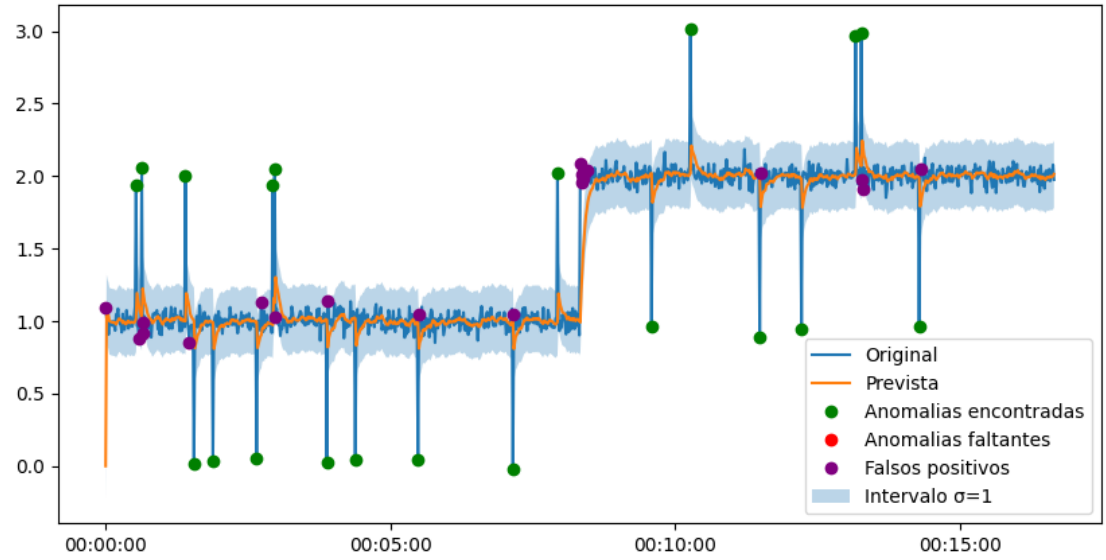
Neste cenário gera-se uma série temporal com valor constante em um e uma mudança discreta de amplitude um, na metade do tempo simulado, conforme mostrado na Figura 16. A mudança discreta acontece em casos onde tem uma variação brusca do valor médio como, por exemplo, o valor lido por um sensor de luminosidade quando uma luminária é acesa ou apagada.

O modelo ARIMA gerado foi o ARIMA(0,1,1). Todas as vinte anomalias foram detectadas. No entanto, o sistema classificou dezesseis pontos como falsos positivos.

4.3.3 Cenário 3

A Figura 18 mostra o cenário 3, onde foi gerado uma série temporal com uma rampa começando em 0 e terminando em 2. Uma rampa aparece em situações onde o valor medido sobe uniformemente, como a velocidade de um veículo com aceleração

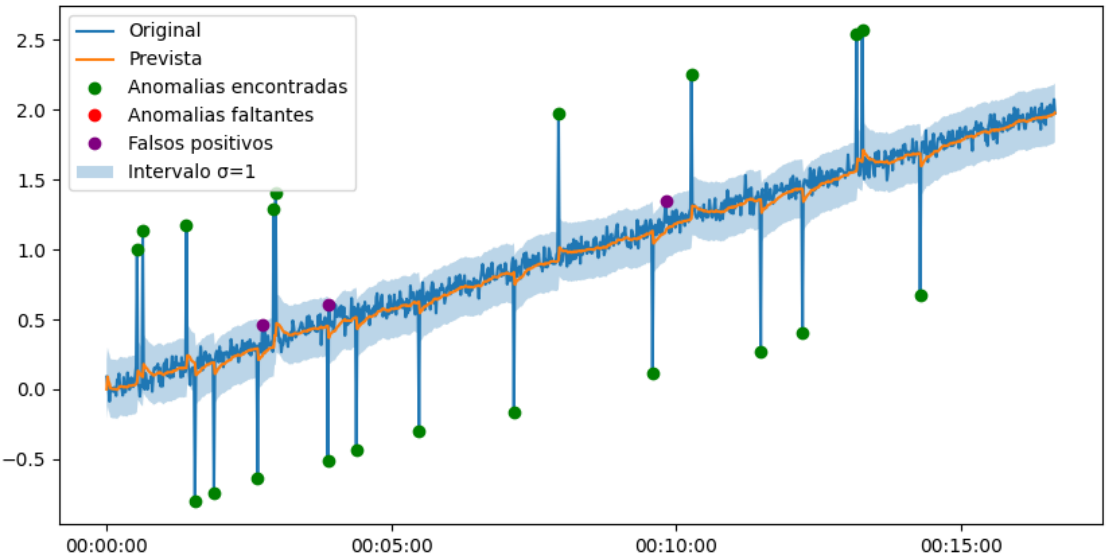
Figura 17 – Resultados do Cenário 2



Fonte: Autor

constante, ou na etapa da rampa na curva de aquecimento de um forno elétrico.

Figura 18 – Resultados do Cenário 3



Fonte: Autor

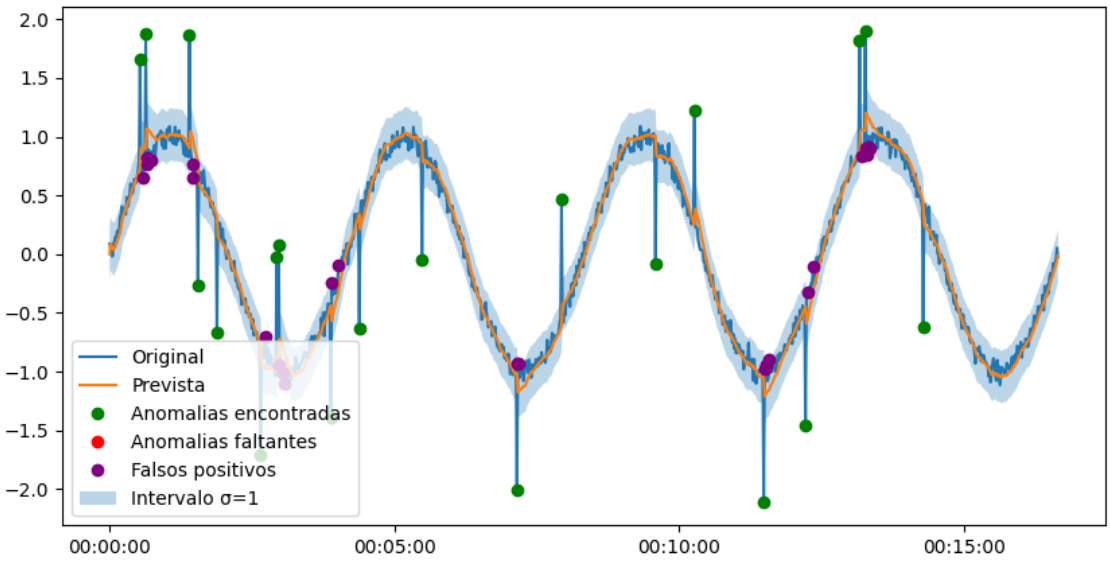
O modelo ARIMA gerado foi o ARIMA(0,1,1). Todas as vinte anomalias foram detectadas. No entanto, o sistema classificou dois pontos como falsos positivos.

4.3.4 Cenário 4

Neste cenário gera-se uma série temporal senoidal de amplitude 1, começando em 0 e finalizando em 8π, como mostrado na Figura 19. Esse tipo de série pode ser

encontrado na amostragem da tensão de alimentação de redes elétricas, ou na posição do eixo de um motor em relação a uma referência axial.

Figura 19 – Resultados do Cenário 4



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(1,1,2). Todas as anomalias foram detectadas, com 11 falsos positivos.

4.3.5 Cenário 5

Neste cenário, como mostrado na Figura 20, gera-se uma série temporal senoidal de amplitude 1, começando em 0 e finalizando em 8π, somado a uma rampa que começa em -2 e finaliza em 3.

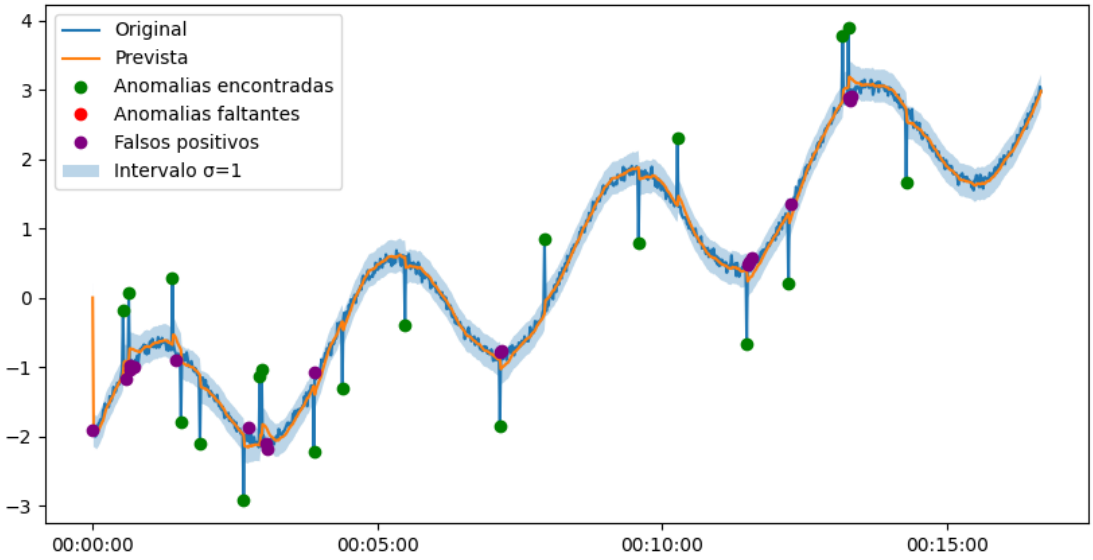
O modelo ARIMA gerado foi o ARIMA(1,1,2). Todas as anomalias foram detectadas, com 13 falsos positivos.

4.3.6 Cenário 6

Neste cenário gera-se uma série temporal utilizando uma equação do segundo grau $\frac{-x^2+1000x}{\frac{1000}{2}^2} * 3$ para utilizar os valores normalizados entre 0 e 3, conforme indica a Figura 21. Esse tipo de cenário é encontrado na amostragem da posição de um veículo com aceleração constante, ou na trajetória de um projétil em queda livre.

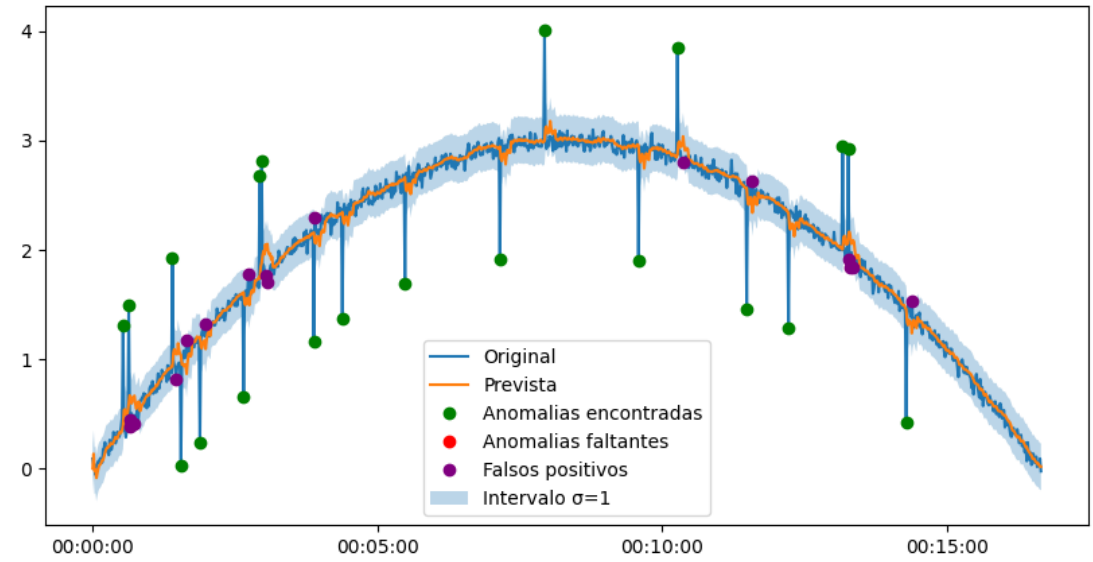
O modelo ARIMA gerado foi o ARIMA(10,2,2). Todas as anomalias foram detectadas, com 15 falsos positivos.

Figura 20 – Resultados do Cenário 5



Fonte: Autor

Figura 21 – Resultados do Cenário 6



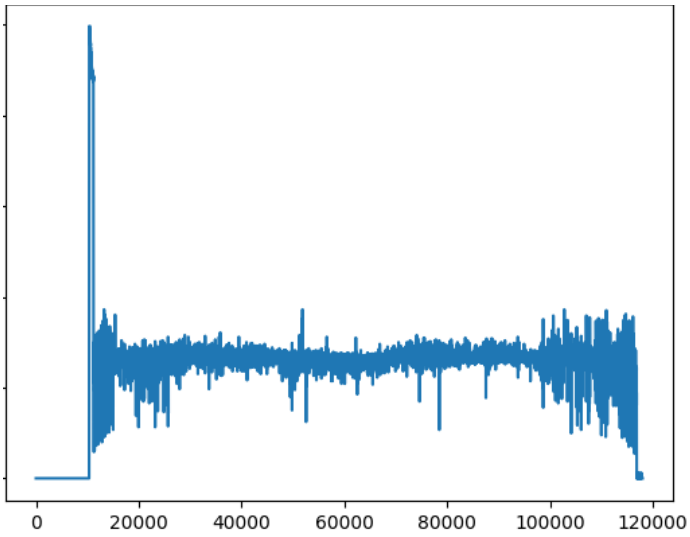
Fonte: Autor

4.3.7 Cenário 7

Neste cenário foi utilizado uma série temporal obtida em um processo de soldagem. Os dados são da corrente lida no processo de soldagem a arco. Foram definidas quatro faixas de dados para melhorar a visualização da aplicação do algoritmo. Uma visualização completa dos dados está mostrada em 22.

A faixa 1 tem uma parte da etapa limitada em 100 por conta do limite de leitura do sensor.

Figura 22 – Cenário 7 completo

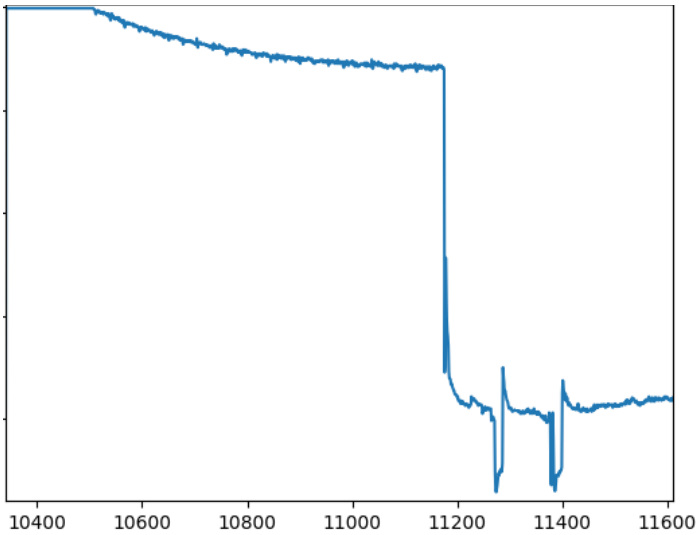


Fonte: Autor

4.3.7.1 Faixa 1

A faixa 1 tem dados entre os pontos 10340 e 11600, como mostra a Figura 23. Essa faixa foi escolhida pois tem uma mudança discreta bem acentuada. O resultado está descrito na Figura 24.

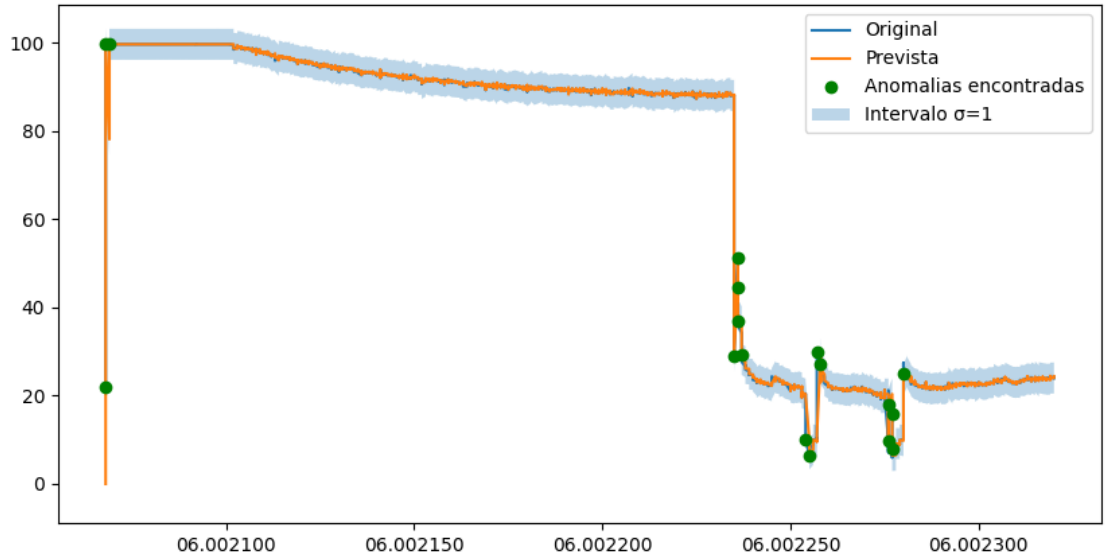
Figura 23 – Resultados do Cenário 7 (Faixa 1)



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(2,1,0). Dos 1260 pontos foram detectadas 17 anomalias.

Figura 24 – Resultados do Cenário 7 (Faixa 1 - saída)

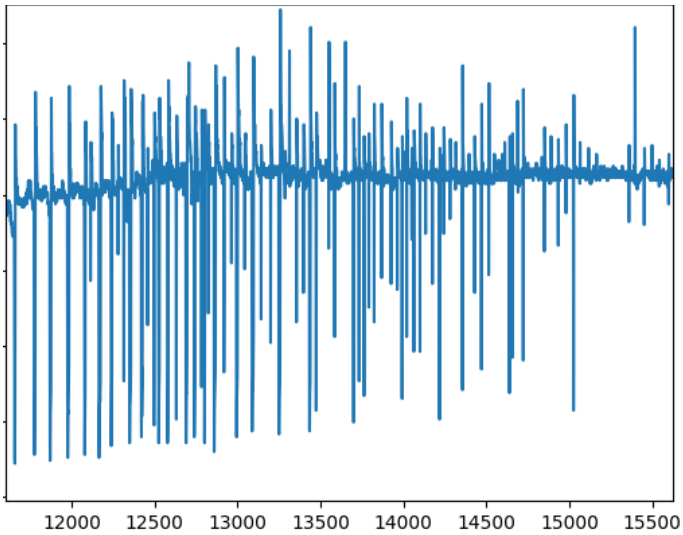


Fonte: Autor

4.3.7.2 Faixa 2

A faixa 2 tem dados entre os pontos 11600 e 15600, como mostra a Figura 25. Essa faixa foi escolhida por ter um valor médio com bastante ruído sobre toda a faixa de dados. O resultado está descrito na Figura 26.

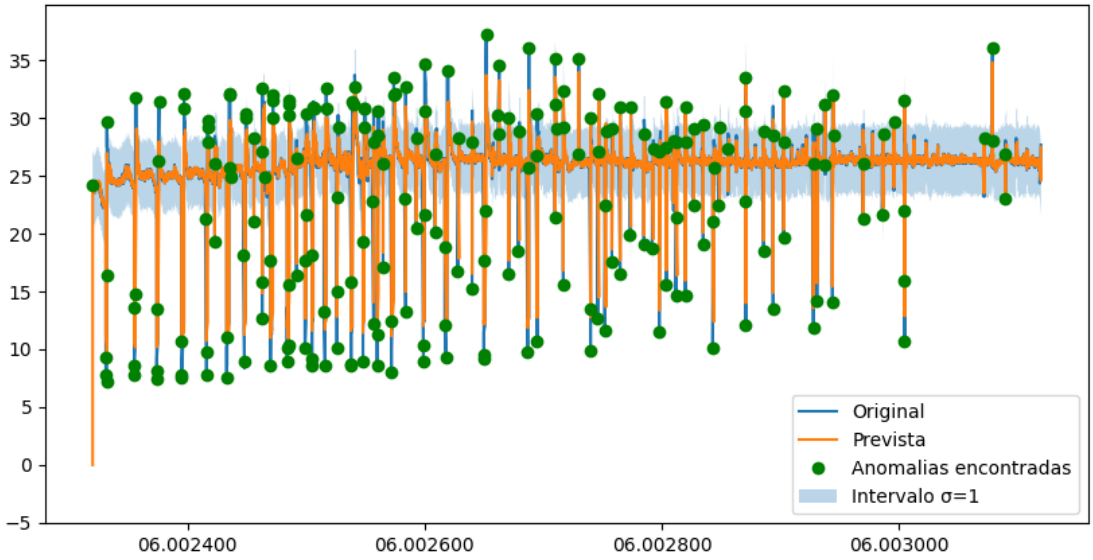
Figura 25 – Resultados do Cenário 7 (Faixa 2)



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(0,1,3). Dos 4000 pontos foram detectadas 224 anomalias.

Figura 26 – Resultados do Cenário 7 (Faixa 2 - saída)

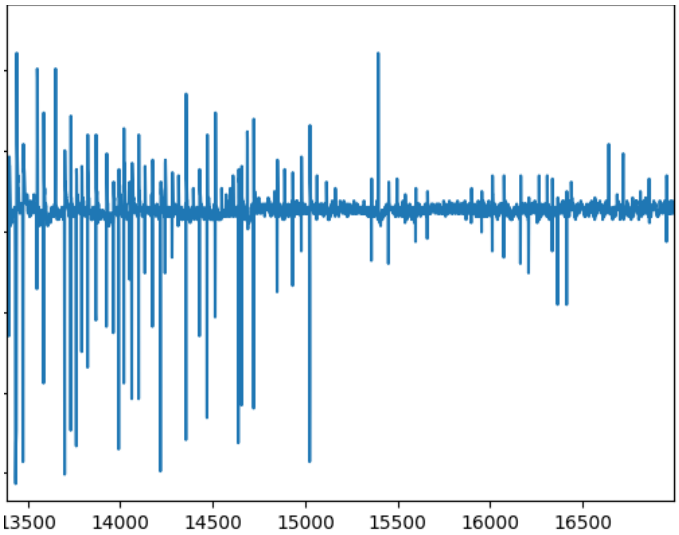


Fonte: Autor

4.3.7.3 Faixa 3

A faixa 3 tem dados entre os pontos 13500 e 16500, como mostra a Figura 27. Essa faixa foi escolhida por ter um valor médio e uma transição entre um ruído intenso e um ruído suave. O resultado está descrito na Figura 28.

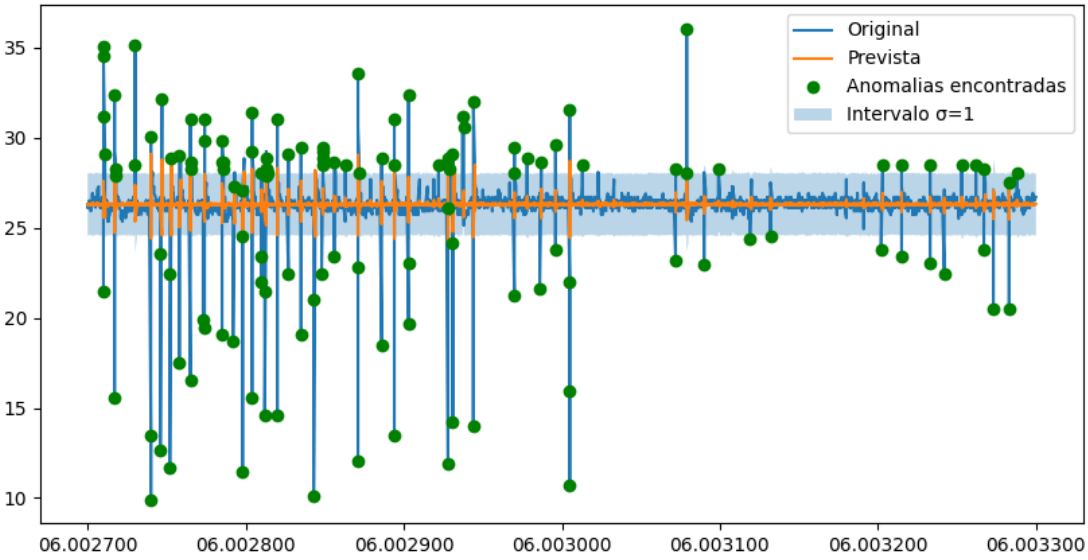
Figura 27 – Resultados do Cenário 7 (Faixa 3)



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(0,0,3). Dos 3000 pontos foram detectadas 129 anomalias.

Figura 28 – Resultados do Cenário 7 (Faixa 3 - saída)

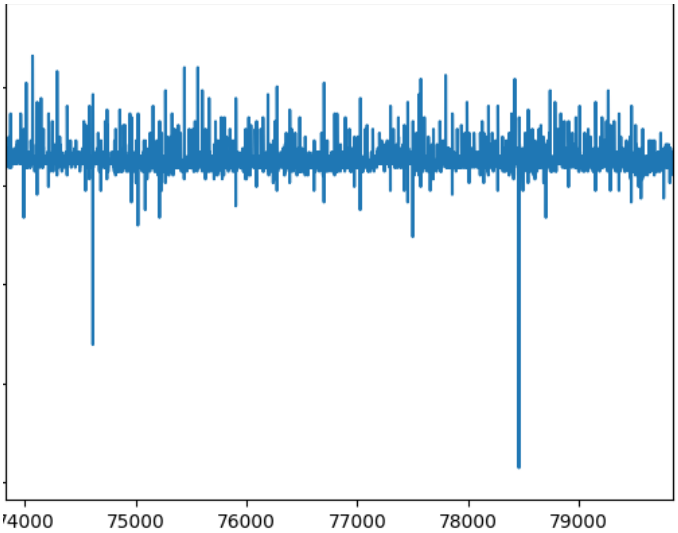


Fonte: Autor

4.3.7.4 Faixa 4

A faixa 4 tem dados entre os pontos 74000 e 79000, como mostra a Figura 29. Essa faixa foi escolhida por ter um valor médio e um ruído branco suave, porém com alguns pontos com um valor bem afastado da média. O resultado está descrito na Figura 30.

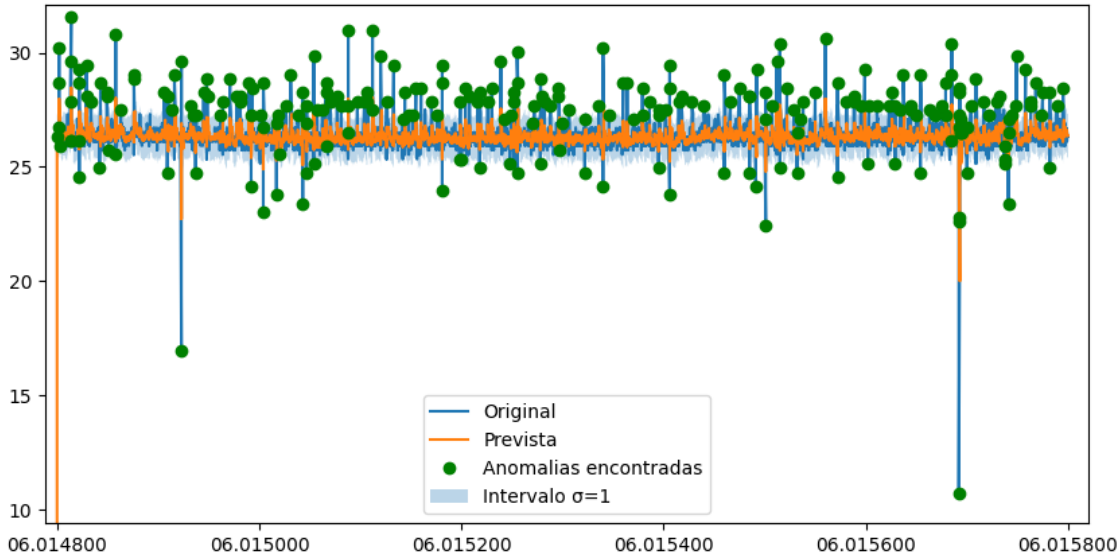
Figura 29 – Resultados do Cenário 7 (Faixa 4)



Fonte: Autor

O modelo ARIMA gerado foi o ARIMA(10,1,0). Dos 5000 pontos foram detectadas 246 anomalias.

Figura 30 – Resultados do Cenário 7 (Faixa 4 - saída)



Fonte: Autor

4.4 DISCUSSÕES

Com os resultados apresentados nas seções anteriores, nessa seção eles serão discutidos. A Tabela 1, apresenta-se o resumo com os resultados encontrados em todos os cenários simulados.

Tabela 1 – Resumo dos resultados obtidos

Cenário	Pontos	Anomalias	Falsos Positivos	Modelo
1	1000	20	0	(0,0,0)
2	1000	36	16	(0,1,1)
3	1000	22	2	(0,1,1)
4	1000	31	11	(1,1,2)
5	1000	33	13	(1,1,2)
6	1000	35	15	(10,2,2)

Fonte: Autor

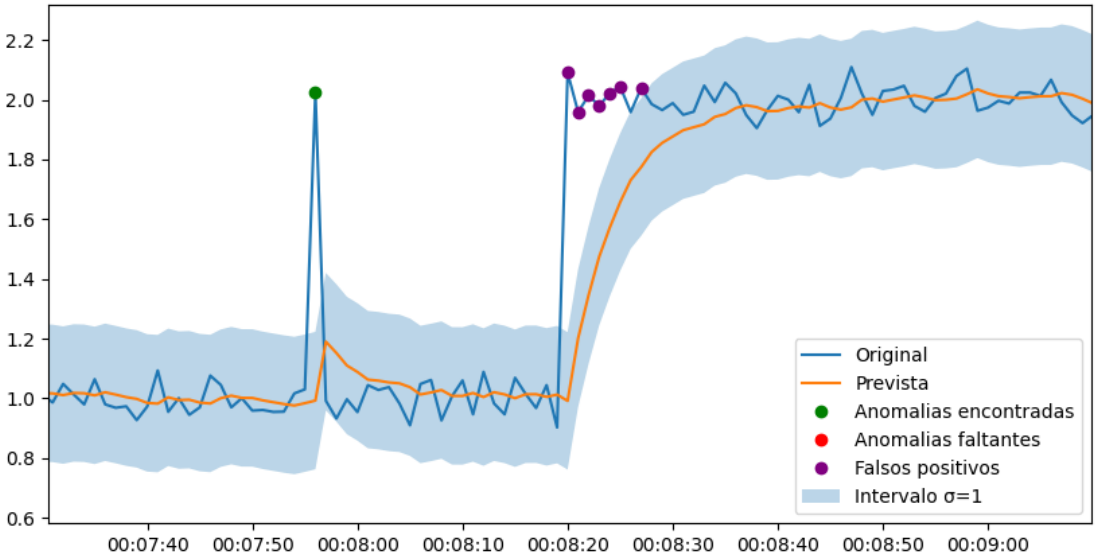
Diante dos resultados mostrados nas figuras e da Tabela 1, serão explicados alguns pontos percebidos nos resultados, trazendo mais informações principalmente sobre os falsos positivos.

4.4.1 Cenário 2

Foi possível notar, como mostra a Figura 31, que no momento da mudança discreta sete pontos foram considerados falsos positivos.

Esse acontecimento leva a entender que o modelo ARIMA gerado não traz

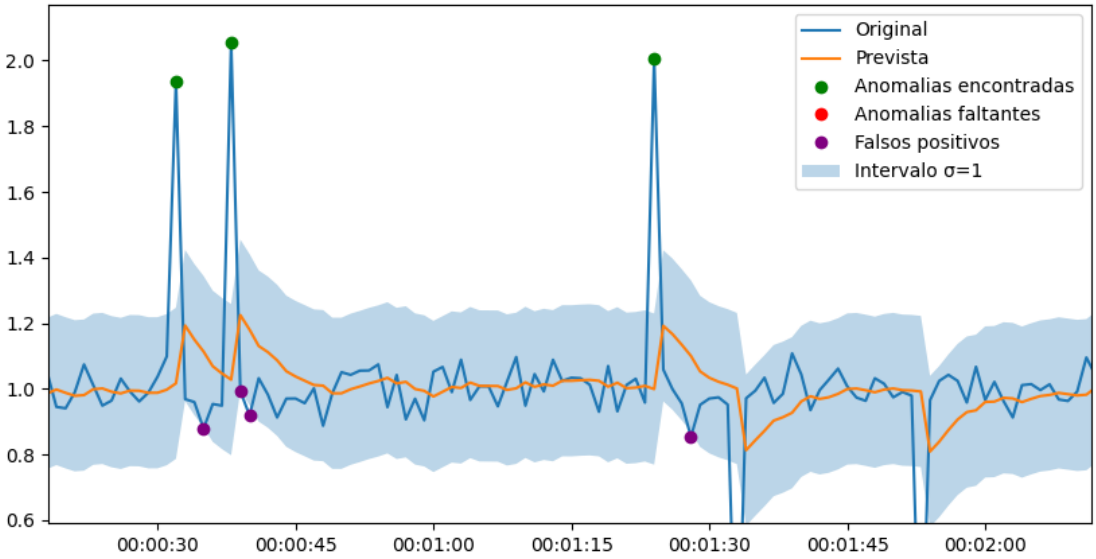
Figura 31 – Resultados do Cenário 2 (Detalhe 1)



Fonte: Autor

resultados confiáveis na região em torno da mudança discreta. Alguns falsos positivos também apareceram logo após a detecção das anomalias pontuais, quando o ruído branco está no lado oposto à anomalia, conforme mostrado na Figura 32.

Figura 32 – Resultados do Cenário 2 (Detalhe 2)

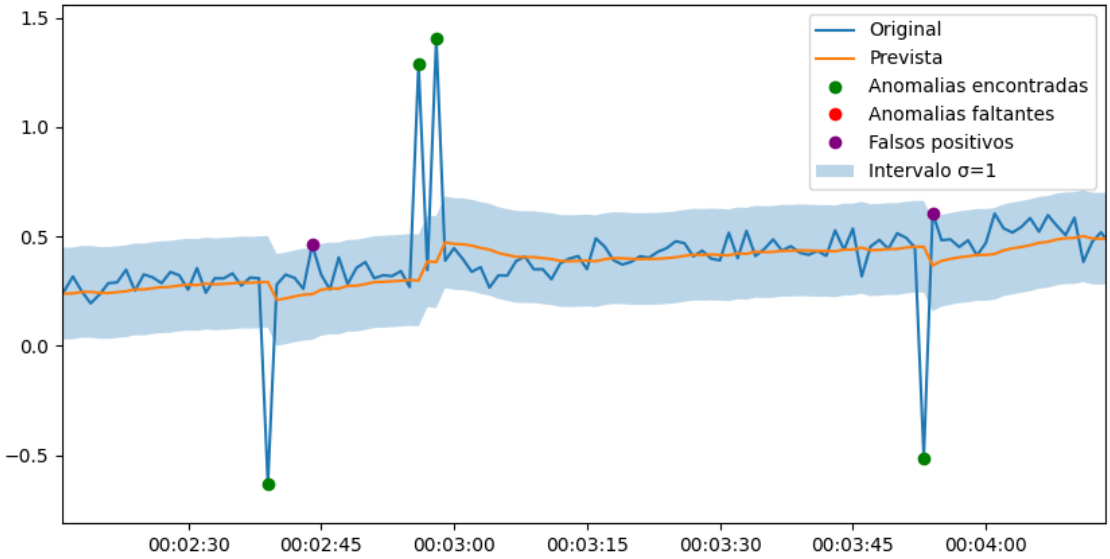


Fonte: Autor

4.4.2 Cenário 3

A Figura 33 mostra que logo após a anomalia alguns falsos positivos foram gerados, o mesmo do Cenário 2 explicado na Figura 32.

Figura 33 – Resultados do Cenário 3 (Detalhe 1)

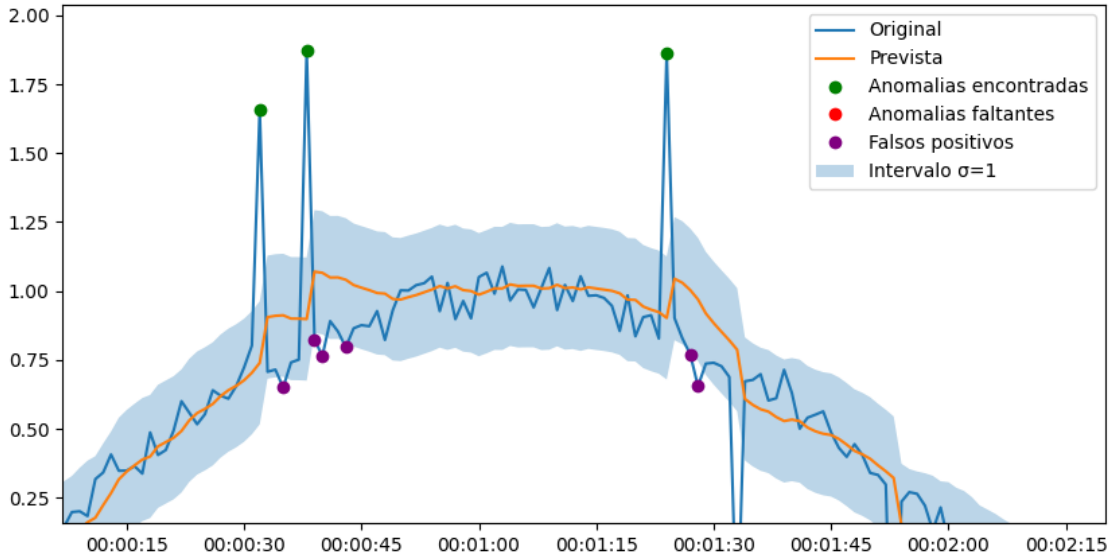


Fonte: Autor

4.4.3 Cenário 4

O mesmo evento do falso positivo gerado após uma anomalia pode ser percebido na Figura 34.

Figura 34 – Resultados do Cenário 4 (Detalhe 1)

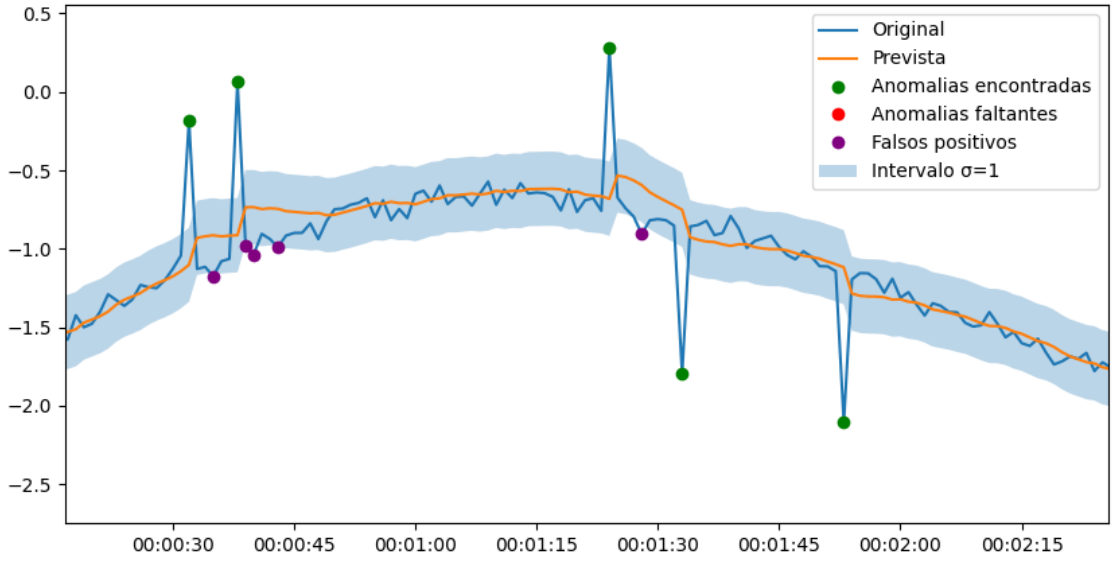


Fonte: Autor

4.4.4 Cenário 5

A Figura 35 mostra que nesse cenário também apareceram os falsos positivos após uma anomalia.

Figura 35 – Resultados do Cenário 5 (Detalhe 1)

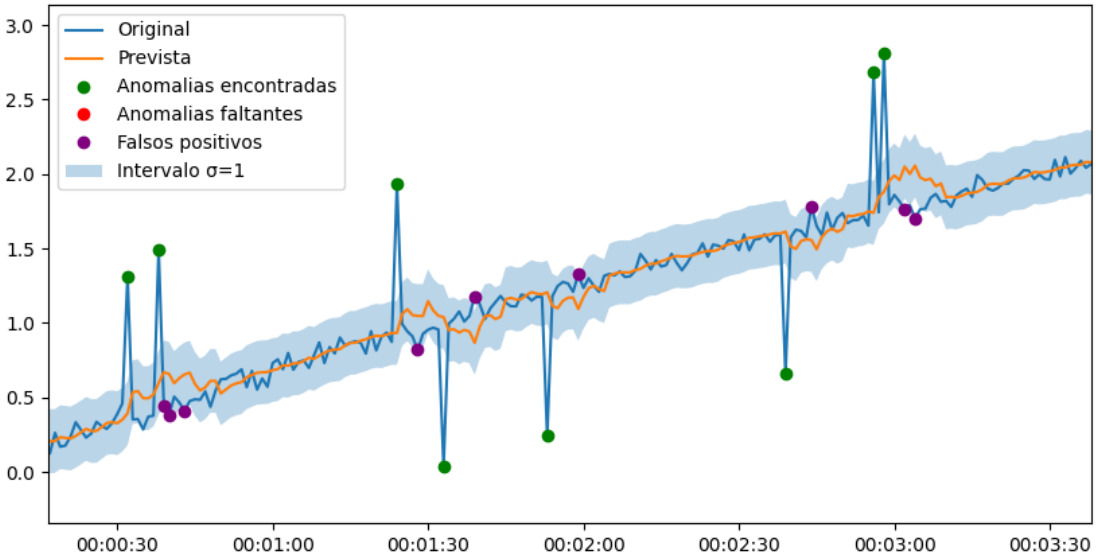


Fonte: Autor

4.4.5 Cenário 6

Finalizando os cenários simulados, a Figura 36 mostra que também apresenta o mesmo evento dos falsos positivos, indicando que esse evento pode ser considerado comum para o método utilizado.

Figura 36 – Resultados do Cenário 6 (Detalhe 1)



Fonte: Autor

Esse evento, no entanto, não ocorreu no Cenário 1. Isso aconteceu, pois, como o modelo ARIMA gerado foi o ARIMA(0,0,0), isso significa que a série prevista é uma função constante, logo a presença de uma anomalia não causa o deslocamento do sinal gerado, assim evitando o aparecimento de falsos positivos.

4.5 LIMITAÇÕES

Nesse trabalho teve seu escopo em dados históricos e anomalias pontuais. Assim, análise durante a geração dos dados e outras anomalias não são abordadas. Nos algoritmos, poucas variações foram exploradas nos parâmetros das abordagens probabilísticas, assim novas variações nos parâmetros podem apresentar resultados diferentes.

Foi escolhido um único valor de desvio padrão e um critério de informação para convergir o *pmdarima*, então outros valores poderão ser explorados.

5 CONCLUSÕES

Uma série temporal com dados errados geram informações erradas ou imprecisas. A análise correta das séries temporais se tornou fundamental para gerar informações corretas e precisas, justificando estudos e trabalhos na identificação e correção dessas séries. Detectar suas anomalias permite qualificar e corrigir o conjunto de dados, para poder utilizá-lo com sucesso.

Este trabalho utilizou um gerador de modelos ARIMA em algumas séries temporais simuladas e em faixas de uma série temporal real para reconstruir as séries e, a partir da média e desvio padrão dos resíduos, definir se os valores da série temporal são anomalias pontuais. Para receber, integrar e mostrar os dados, foi utilizada a plataforma *Thingsboard*, que permitiu escolher a faixa de dados e o algoritmo a ser utilizado.

Os resultados obtidos atingiram os objetivos esperados, o modelo ARIMA com a técnica dos resíduos conseguiram encontrar todas as anomalias pontuais simuladas. A plataforma atendeu todas as funcionalidades desejadas com sucesso, facilitando a utilização das séries temporais e está em constante evolução, então é esperado que melhore cada vez mais com o tempo.

Para trabalhos futuros poderão ser variados os parâmetros estatísticos, como o desvio-padrão, ou utilizar outras técnicas de detecção, estatísticos ou com inteligência artificial, podendo até buscar detectar outros tipos de anomalias, como a contextual e coletiva. Pode-se também complementar o método proposto para reduzir o número de falsos positivos, como mostrado nas Figuras 31 e 32.

REFERÊNCIAS

AL., T. G. S. et. **pmdarima: ARIMA estimators for Python**. 2017–. [Online; accessed <today>]. Disponível em: <http://www.alkaline-ml.com/pmdarima>.

BLáZQUEZ-GARCíA, A. et al. **A review on outlier/anomaly detection in time series data**. 2020.

BRAEI, M.; WAGNER, S. **Anomaly detection in univariate time-series: A survey on the state-of-the-art**. 2020.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. **Anomaly Detection : A Survey**. 2009.

COOK, A. A.; MISIRLI, G.; FAN, Z. **Anomaly Detection for IoT Time-Series Data: A Survey**. 2020.

ELMENSHAWY, D.; HELMY, W. Detection techniques of data anomalies in iot: A literature survey. **International Journal of Civil Engineering and Technology**, v. 9, 2018. ISSN 09766316.

GADDAM, A. et al. Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions. **Electronics (Switzerland)**, v. 9, 2020. ISSN 20799292.

GARG, N. **Apache Kafka**. [S.l.]: Packt Publishing, 2013. ISBN 1782167935.

GARMAROODI, M. S. S. et al. **Detection of anomalies and faults in industrial IoT systems by data mining: Study of CHRIST Osmotron water purification system**. 2020.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HYNDMAN, R.; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. 2nd. ed. Australia: OTexts, 2018.

KLIMA, M. et al. **Quality and reliability metrics for IoT systems: A consolidated view**. 2020.

LEE, S.; KIM, H. K. Adsas: Comprehensive real-time anomaly detection system. **CoRR**, abs/1811.12634, 2018. Disponível em: <http://arxiv.org/abs/1811.12634>.

MOORE, S. J. et al. Iot reliability: a review leading to 5 key research directions. **CCF Transactions on Pervasive Computing and Interaction**, v. 2, 2020. ISSN 2524-521X.

POKORNI, S. Reliability and availability of the internet of things. **Vojnotehnicki glasnik**, v. 67, 2019. ISSN 0042-8469.

RAFIQUL, M. et al. A comprehensive survey of time series anomaly detection in online social network data. **International Journal of Computer Applications**, v. 180, 2017.

REBACK, J. et al. **pandas-dev/pandas: Pandas**. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>.

SEABOLD, S.; PERKTOLD, J. statsmodels: Econometric and statistical modeling with python. In: **9th Python in Science Conference**. [S.l.: s.n.], 2010. Acesso em: 19 junho. 2022.

SEBESTYEN, G. et al. A taxonomy and platform for anomaly detection. In: . [S.l.: s.n.], 2018.

SUN, B.; MA, L. An overview of outliers and detection methods in general for time series from iot devices. In: . [S.l.: s.n.], 2021. v. 1274 AISC. ISSN 21945365.

THINGSBOARD. **ThingsBoard Documentation**. New York, 2021. Disponível em: <https://thingsboard.io/docs/>. Acesso em: 24 abril. 2021.