Camilla Carvalho Alves
Matrícula: 2312695

Aplicação para Detecção de Anomalias Univariadas em Séries Temporais

Rio de Janeiro - RJ

Junho - 2024

Sumário

1 Introdução

A detecção de anomalias, também conhecida como detecção de valores discrepantes, tem sido uma área de pesquisa relevante e ativa por várias décadas (PANG et al., 2021). Esta tarefa é aplicável em diversas áreas devido à sua capacidade de identificar padrões anômalos que podem indicar problemas ou eventos significativos.

Na manufatura, por exemplo, anomalias nos dados de sensores podem indicar falhas em máquinas, permitindo a manutenção preditiva e reduzindo o tempo de inatividade (ERHAN et al., 2021). No setor financeiro, a detecção de anomalias em transações pode sinalizar atividades fraudulentas, ajudando a proteger instituições e clientes (HILAL; GADSDEN; YAWNEY, 2021).

Na área da saúde, a análise de sinais vitais é essencial para detectar problemas emergentes, sendo crucial em unidades de terapia intensiva e no monitoramento remoto de pacientes (SALEM; LIU; MEHAOUA, 2013). Anomalias no consumo de energia podem indicar ineficiências ou mau funcionamento de equipamentos, promovendo tanto a economia de recursos quanto a sustentabilidade ambiental (LEI et al., 2023). Em segurança cibernética, monitorar o tráfego de rede é fundamental para identificar e mitigar ataques e invasões (ALABADI; CELIK, 2020).

Assim, a necessidade de ferramentas acessíveis e fáceis de usar para a detecção de anomalias é crescente, especialmente à medida que grandes volumes de dados se tornam mais comuns em diversas indústrias. Essas ferramentas permitem que usuários, mesmo sem um conhecimento técnico profundo, possam identificar rapidamente padrões anômalos em suas bases de dados. Facilitar a detecção de anomalias pode levar a ações preventivas mais eficazes, minimizando prejuízos e otimizando processos.

1.1 Objetivo

O objetivo deste trabalho é desenvolver uma aplicação para a detecção de anomalias univariadas em séries temporais. Esta aplicação utiliza uma biblioteca própria que implementa métodos estatísticos e computacionais conhecidos para identificar padrões anômalos nos dados importados pelo usuário.

1.2 Organização do Documento

Este relatório está organizado da seguinte forma: No Capítulo 2, é apresentada a especificação de requisitos, dividida em requisitos funcionais e não-funcionais, que detalham as necessidades do sistema. O Capítulo 3 apresenta o diagrama de caso de uso, descrevendo as interações do usuário com o sistema. No Capítulo 4, é detalhado o projeto do programa, incluindo as tecnologias utilizadas, a arquitetura do projeto e os métodos empregados para a detecção de anomalias. O Capítulo 5 consiste no manual do usuário, fornecendo instruções sobre execução e funcionalidades da aplicação. Por fim, o Capítulo 6 apresenta um exemplo de uso com uma série temporal de tráfego de táxi em Nova Iorque.

2 Especificação de Requisitos

Este capítulo apresenta os requisitos funcionais e não-funcionais do projeto desenvolvido.

2.1 Requisitos Funcionais

- Ajustes dos Métodos Estatísticos e Computacionais
  - Descrição: A aplicação deve permitir a implementação de métodos estatísticos e computacionais tradicionais para detectar anomalias univariadas em séries termporais.
  - Funcionalidades: Utilizar métodos como Intervalo Interquartil (IQR), Z-Score, Média Móvel, Fator de Outlier Local (LOF), Decomposição Sazonal e K-Means para essa finalidade.
- Comparação dos Métodos Implementados
  - Descrição: A aplicação deve possibilitar a comparação dos resultados dos métodos implementados.
  - Funcionalidades: Apresentar uma tabela comparativa com os métodos implementados, destacando a quantidade de anomalias encontradas e a proporção de anomalias em relação à base de dados original do usuário para cada método utilizado.
- Visualização de Resultados
  - Descrição: A aplicação deve apresentar visualizações gráficas e exportações para facilitar a compreensão dos resultados obtidos.
  - Funcionalidades: Gerar gráficos da série temporal original do usuário, destacando as anomalias encontradas por cada método, e exportar os pontos anômalos identificados para o usuário.

2.2 Requisitos Não-Funcionais

- Eficiência Computacional
  - Descrição: A aplicação deve ser eficiente computacionalmente para lidar com conjuntos de dados de tamanho moderado.
  - Critérios de Avaliação: O tempo de execução e o uso de recursos devem ser razoáveis para conjuntos de dados típicos.
- Facilidade de Manutenção
  - Descrição: O código deve ser organizado e comentado de forma a facilitar a manutenção futura.
  - Critérios de Avaliação: Estrutura de código clara, uso de funções e comentários explicativos devem ser adotados.
- Compatibilidade com Dados de Entrada
  - Descrição: A aplicação deve ser flexível o suficiente para lidar com diferentes conjuntos de dados relacionados à detecção de anomalias.
  - Critérios de Avaliação: Capacidade de adaptar-se a diferentes tipos de conjuntos de dados com uma série temporal.
- Documentação Adequada
  - Descrição: A aplicação deve possuir documentação clara e abrangente, incluindo informações sobre o propósito de cada funcionalidade e instruções para execução.
  - Critérios de Avaliação: A documentação deve ser completa, fácil de entender e acessível aos usuários.

3 Diagrama de Caso de Uso

Neste capítulo é apresentado o diagrama de casos de uso criado para o sistema. A Figura 1 mostra a criação dele com base nos requisitos do sistema, assim como a especificação de cada um.

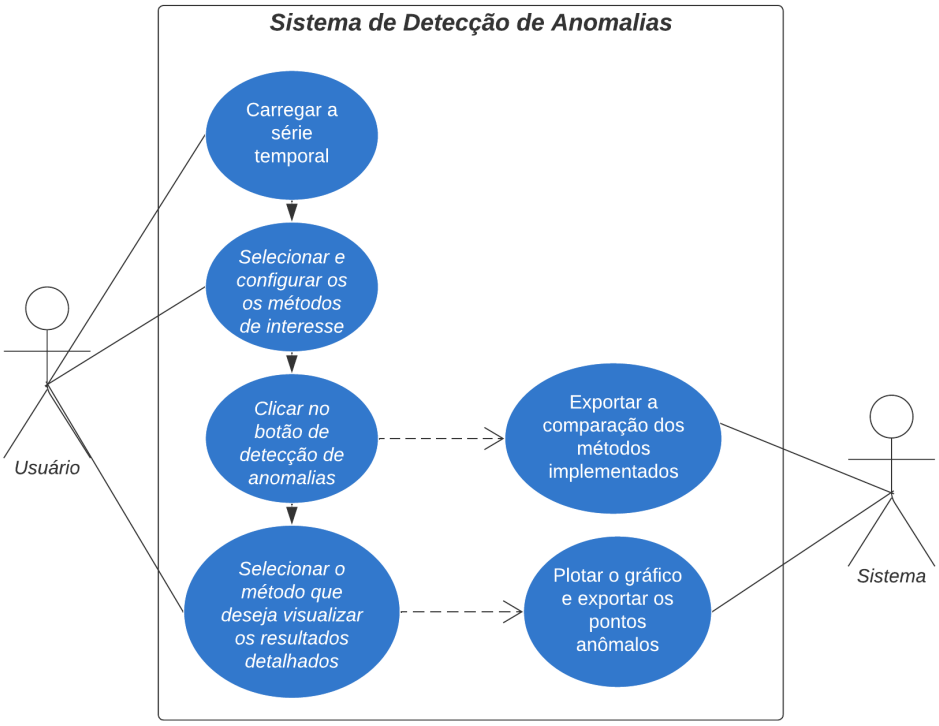


Figura 1 – Diagrama de casos de uso.

1. Carregar a série temporal.

Descrição	
Ator:	O usuário.
Descrição sucinta:	Abrir e carregar uma série temporal.
Pré-condições:	Estar com a aplicação aberta.
Cenário principal:	1. O usuário abre a aplicação; 2. O usuário clica no botão de carregar série temporal; 3. O usuário seleciona e carrega um csv com a primeira coluna sendo os valores da série temporal; 4. O sistema informa que a série temporal foi carregada; 5. O sistema exibe os métodos disponíveis para configuração.

2. Detectar as anomalias.

Descrição	
Ator:	O usuário.
Descrição sucinta:	Detectar anomalias de forma consolidada.
Pré-condições:	Estar com a aplicação aberta e com a base carregada.
Cenário principal:	1. O usuário seleciona os métodos de interesse; 2. O usuário configura os métodos de interesse; 3. O usuário clica no botão de detectar anomalias; 4. O sistema exporta os resultados consolidados dos métodos; 5. O sistema habilita a visualização detalhada de cada método.
Cenário alternativo:	Base carregada não estar no formato adequado para funcionalidade de detecção de anomalias proposta.

3. Visualizar os resultados dos métodos.

Descrição	
Ator:	O usuário.
Descrição sucinta:	Visualizar os resultados de cada método.
Pré-condições:	Estar com a aplicação aberta com os resultados consolidados.
Cenário principal:	1. O usuário seleciona o método de interesse; 2. O usuário clica na opção de plotar anomalias; 3. O sistema plota o gráfico com as anomalias destacadas; 4. O sistema exporta os dados identificados como anômalos.

4 Projeto do Programa

Este capítulo apresenta informações sobre odesenvolvimento da aplicação, como a biblioteca e métodos utilizados.

4.1 Tecnologias Utilizadas

A linguagem de programação escolhida para este projeto foi Python, versão 3.12.4. Além disso, diversas bibliotecas também foram utilizadas, conforme detalhado na Tabela 1.

Tabela 1 – Lista das dependências necessárias para o projeto.

Biblioteca	**Versão**
numpy	2.0.0
pandas	2.2.2
matplotlib	3.9.0
scikit-learn	1.5.0
statsmodels	0.14.2
pandastable	0.13.1
tkinter	8.6

4.2 Arquitetura do Projeto

A aplicação foi projetada com uma arquitetura modular para facilitar a manutenção e a extensibilidade, conforme ilustrado na Figura 2:

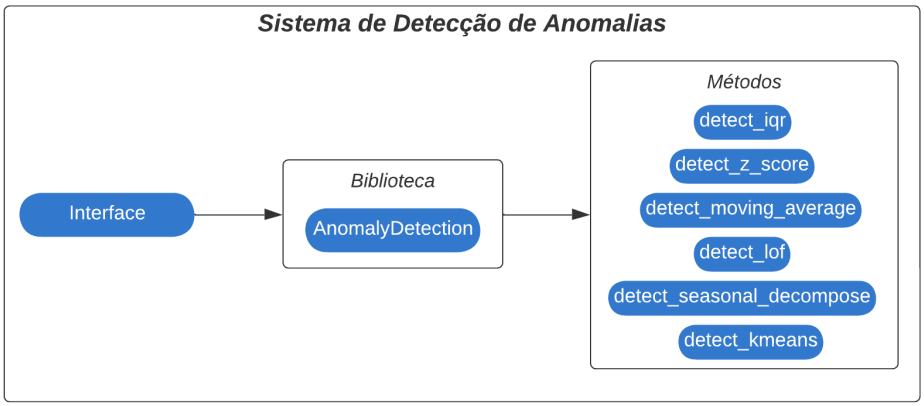


Figura 2 – Diagrama dos componentes

- **Interface da Aplicação**: A interface gráfica, construída com a biblioteca `tkinter`, permite o usuário carregar uma série temporal, selecionar métodos de detecção de anomalias, e visualizar os resultados em gráficos e tabelas. A interface consome a biblioteca `AnomalyDetection`.
- **Biblioteca AnomalyDetection**: A biblioteca `AnomalyDetection` é responsável por implementar os métodos de detecção de anomalias. Ela fornece uma classe com funções específicas para cada método de detecção.
- **Métodos de Detecção**: Os métodos de detecção são implementados na classe `AnomalyDetection` e são chamados pela interface da aplicação conforme a seleção do usuário. Estes métodos incluem técnicas estatísticas e computacionais para identificar pontos anômalos.

4.2.1 Métodos Utilizados

A seguir, detalhamos os métodos implementados na aplicação:

- `detect_iqr`: Implementa a detecção de anomalias baseada no intervalo interquartil. O método IQR é baseado na análise da distribuição dos dados. Ele calcula os quartis $Q1$ *e* $Q3$ da série temporal e a diferença interquartil $IQR = Q3 - Q1$. Pontos são considerados anomalias se estiverem abaixo de $Q1 - 1.5 \times IQR$ ou acima de $Q3 + 1.5 \times IQR$.
- `detect_z_score`: Implementa a detecção de anomalias baseada no Z-score. Este método utiliza a padronização dos dados, calculando o Z-score para cada ponto da série temporal. O Z-score indica quantos desvios padrão um ponto está afastado da média. Valores absolutos de Z-score acima de um limite (geralmente 3) são considerados anomalias.
- `detect_moving_average`: Implementa a detecção de anomalias usando média móvel. A média móvel calcula a média de uma janela de pontos na série temporal. Um ponto é considerado uma anomalia se a diferença entre o ponto e a média móvel for maior que um múltiplo do desvio padrão da série temporal.
- `detect_lof`: Implementa a detecção de anomalias usando o fator de outlier local. O LOF é um método que mede a densidade local de cada ponto em relação aos seus vizinhos. Pontos com densidade significativamente menor que a de seus vizinhos são considerados anomalias.
- `detect_seasonal_decompose`: Implementa a decomposição sazonal para detecção de anomalias. A decomposição sazonal separa a série temporal em três componentes: tendência, sazonalidade e resíduo. O método de decomposição utilizado pode ser

aditivo ou multiplicativo. Os resíduos são analisados para detectar anomalias, onde valores fora de um limite de desvios padrão são considerados atípicos.

- `detect_kmeans`: Implementa a detecção de anomalias usando K-Means. K-Means é um algoritmo de clusterização que particiona os dados em K grupos (clusters). Após o agrupamento, calcula-se a distância de cada ponto ao centro do seu cluster. Pontos com distância significativamente maior que a média são considerados anomalias.

4.3 Código Fonte

O código deste projeto está disponível no GitHub[1], ferramenta usada tanto para acesso dos usuários quanto para versionamento. O código foi escrito em Python, conforme descrito anteriormente.

[1] https://github.com/camillalves/INF2102

5 Manual do Usuário

Neste capítulo serão detalhados os aspectos essenciais sobre a utilização do sistema.

5.1 Público-Alvo

A aplicação é direcionada a analistas de dados, cientistas de dados, desenvolvedores e qualquer pessoa interessada em detectar anomalias em séries temporais utilizando métodos estatísticos e computacionais.

5.2 Instalação e Execução

A instalação e execução do projeto pode ser realizada através do clone do repositório do GitHub para o ambiente local. Após clonar o repositório, é necessário instalar todas as dependências necessárias para a aplicação utilizando o arquivo `requirements.txt`.

Por fim, deve-se navegar até o diretório onde o arquivo principal da aplicação está localizado e executá-lo. A interface gráfica da aplicação será iniciada, permitindo que o usuário carregue uma série temporal, selecione e configure os métodos de detecção de anomalias e visualize os resultados.

5.3 Funcionalidades

Para carregar uma série temporal, o usuário deve selecionar a opção "Carregar série temporal". Em seguida, a janela de seleção de arquivo será exibida, permitindo ao usuário escolher um arquivo CSV contendo os dados da série temporal. Após a seleção do arquivo, a série temporal será carregada na aplicação.

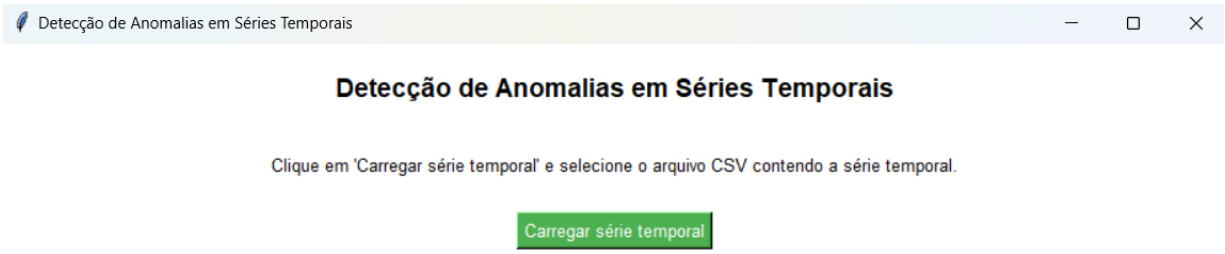


Figura 3 – Carregamento da Série Temporal

Após carregar a série temporal, o usuário deve selecionar os métodos de detecção desejados, marcando as caixas de seleção correspondentes. Cada método possui parâmetros configuráveis que podem ser ajustados conforme necessário.

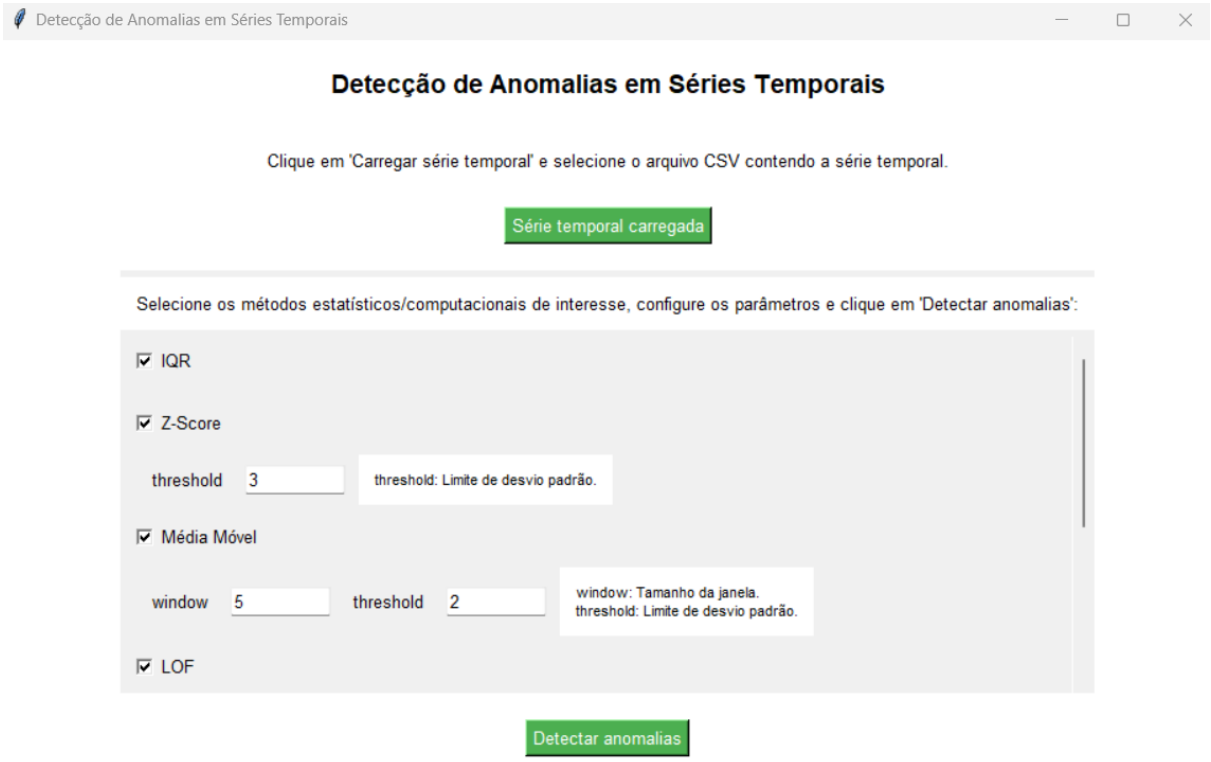


Figura 4 – Seleção e configuração dos métodos

Após selecionar os métodos e configurar os parâmetros, o usuário deve clicar no botão "Detectar anomalias". A aplicação processará os dados utilizando os métodos de detecção de anomalias selecionados e apresentará uma comparação dos resultados. Esta comparação incluirá uma tabela com o número de anomalias detectadas por cada método, além da proporção de anomalias na base de dados. A tabela será exibida em uma nova janela, facilitando a visualização dos resultados.

Além disso, a aplicação permite visualizar graficamente as anomalias detectadas. Para isso, o usuário pode selecionar o método desejado na caixa de seleção indicada e, em seguida, clicar no botão "Plotar Anomalias".

Figura 5 – Visualização dos resultados

Após isso, a aplicação gerará um gráfico de linha da série temporal, destacando os pontos considerados anômalos pelo método selecionado. Este gráfico será exibido em uma nova janela, proporcionando uma visualização clara das anomalias na série temporal. Para cada método de detecção de anomalias, a aplicação também exibirá uma lista com os índices e valores das anomalias detectadas. Esta funcionalidade auxilia o usuário a identificar e analisar detalhadamente os pontos anômalos na série temporal.

6 Teste

Este capítulo apresenta um exemplo de uso com uma série temporal de teste para garantir a funcionalidade da aplicação.

6.1 Conjunto de dados

Para testar a aplicação, utilizou-se um conjunto de dados não rotulados fornecido pela NYC Taxi and Limousine Commission, extraído da plataforma Kaggle[1]. Esta plataforma é amplamente reconhecida no campo da ciência de dados por suas competições, tutoriais e uma variedade de datasets de alta qualidade.

O conjunto de dados selecionado registra o número de passageiros de táxi na cidade de Nova Iorque, com dados capturados em intervalos de 30 minutos. Cada registro contém a data, o horário e o número total de passageiros de táxi naquele intervalo específico. Este conjunto de dados cobre o período de 1º de julho de 2014 a 25 de janeiro de 2015, apresentando um registro contínuo e sem valores faltantes.

O objetivo ao utilizar esses dados é identificar padrões de comportamento atípicos no tráfego de táxis, que possam indicar anomalias. Para focar na análise de anomalias na série temporal, foi mantida apenas a coluna correspondente ao volume de tráfego. Esta simplificação permite que a aplicação se concentre exclusivamente na variável de interesse.

6.2 Resultados

Para o teste da aplicação, foram utilizados os parâmetros padrão para cada método de detecção de anomalias. Os limites de desvio padrão foram configurados como 3 para o método Z-Score, 2 para o método da Média Móvel, e 1.5 para os métodos LOF e K-Means. No método da Média Móvel, foi utilizada uma janela de tamanho 5. O método LOF foi configurado para considerar 20 vizinhos. A decomposição sazonal foi realizada utilizando o modelo aditivo, e o K-Means foi configurado para agrupar os dados em 2 clusters.

A Figura 6 apresenta uma tabela comparativa que resume os resultados obtidos com cada método. Através desta tabela, é possível visualizar que o método K-Means foi o que mais identificou anomalias na série temporal testada, enquanto os métodos Z-Score e Média Móvel identificaram apenas uma anomalia cada, demonstrando uma menor sensibilidade em comparação aos outros métodos.

[1] https://www.kaggle.com/datasets/julienjta/nyc-taxi-traffic/code

Resultados da Comparação

	Método	Número de anomalias	Proporção de anomalias na base
1	IQR	2	0.02%
2	Z-Score	1	0.01%
3	Média Móvel	1	0.01%
4	LOF	19	0.18%
5	Decomposição Sazonal	7	0.07%
6	K-Means	994	9.63%

Figura 6 – Resultados comparativos dos métodos

As Figuras 7 a 18 fornecem uma visualização detalhada dos resultados de cada método. Cada figura inclui um gráfico de linhas da série temporal original com as anomalias identificadas destacadas em vermelho. Além dos gráficos, cada método apresenta uma tabela listando os índices e os valores das anomalias detectadas.

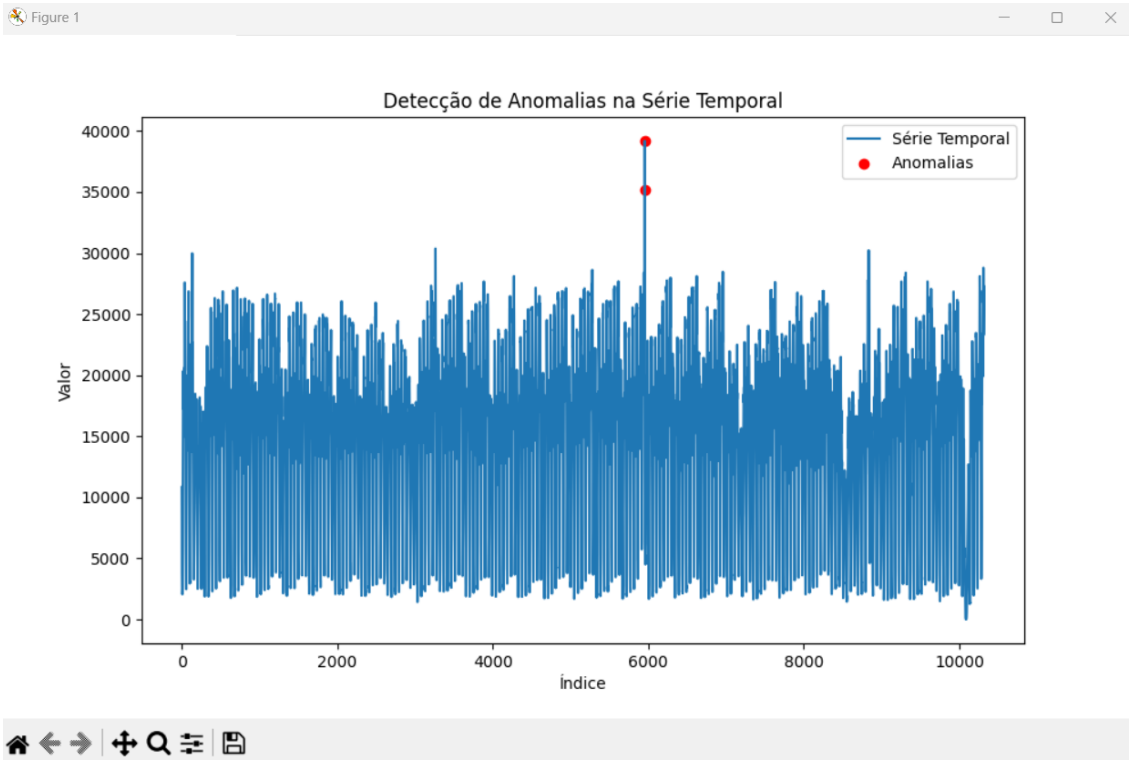


Figura 7 – Resultados do método IQR

Índice e Valores...

	Index	Valor
1	5954	39197.00
2	5955	35212.00

Figura 8 – Resultados do método IQR

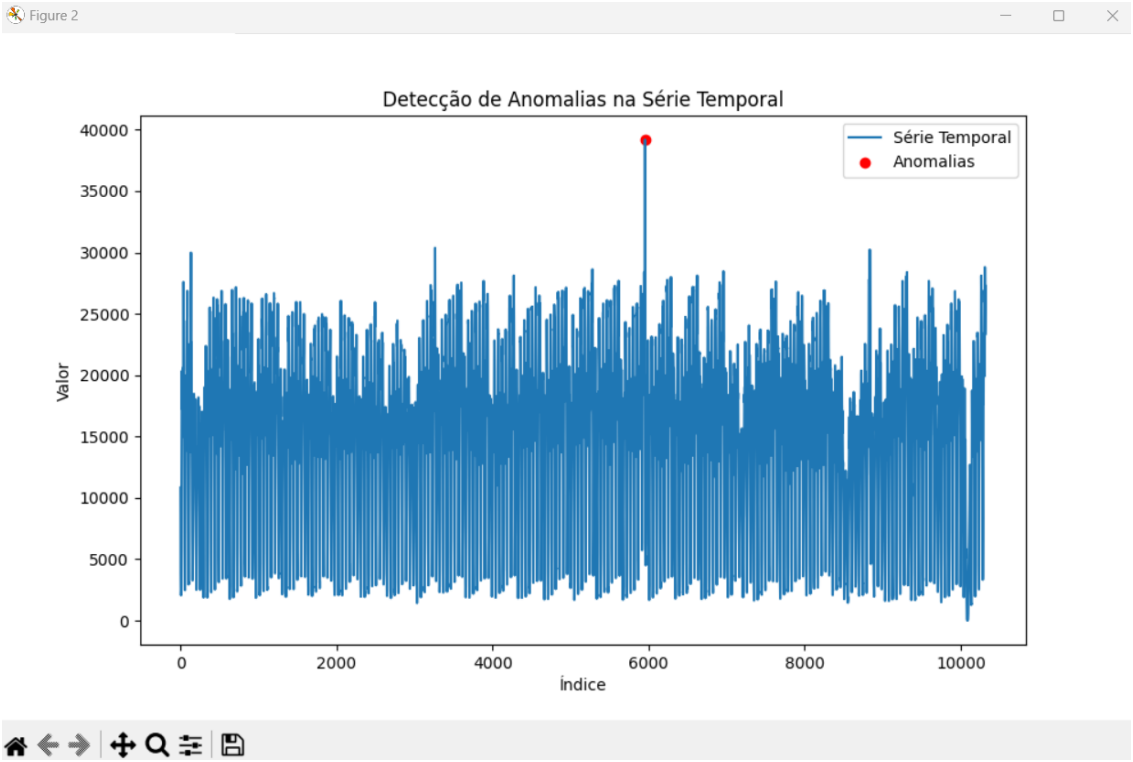


Figura 9 – Resultados do método Z-Score

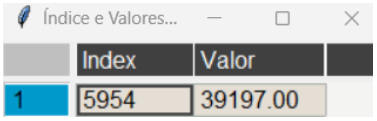


Figura 10 – Resultados do método Z-Score

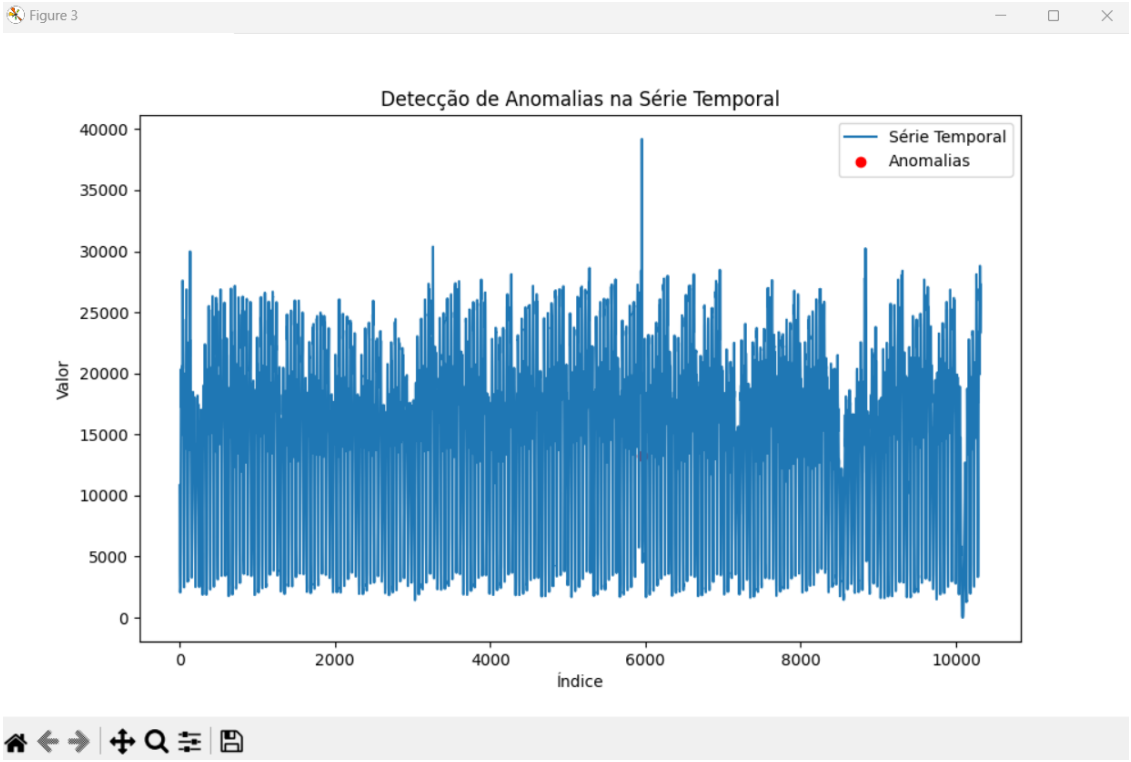


Figura 11 – Resultados do método Média Móvel

Figura 12 – Resultados do método Média Móvel

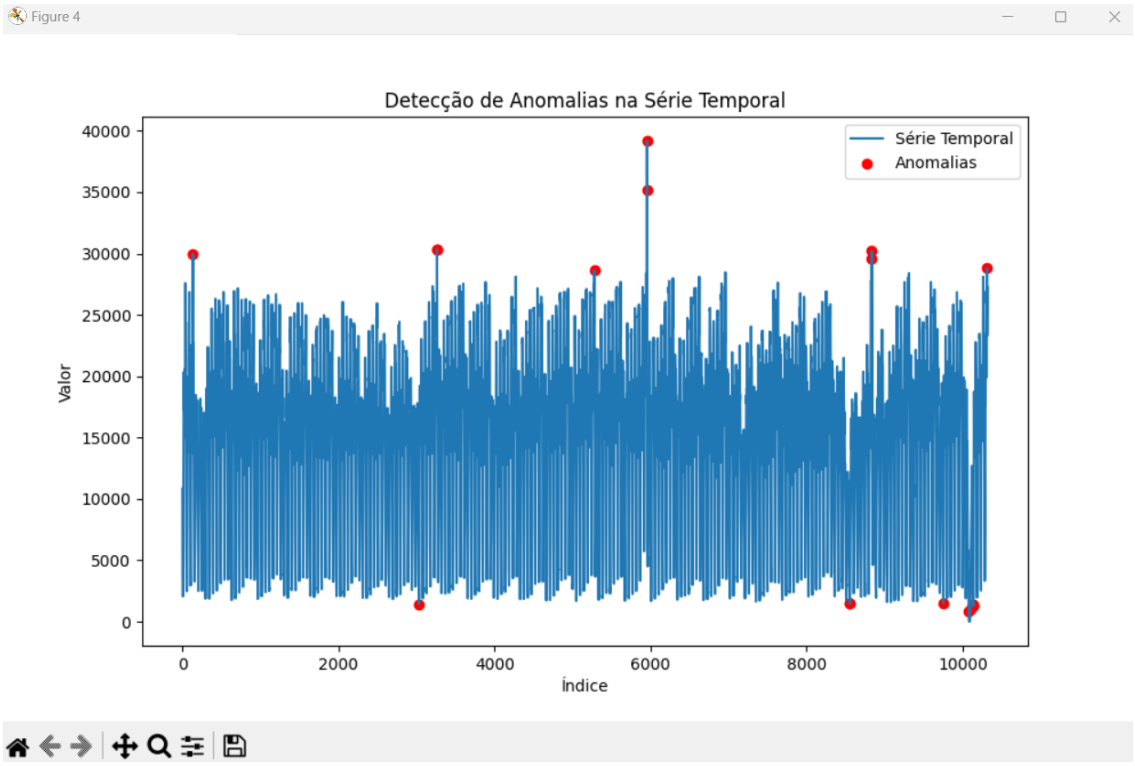
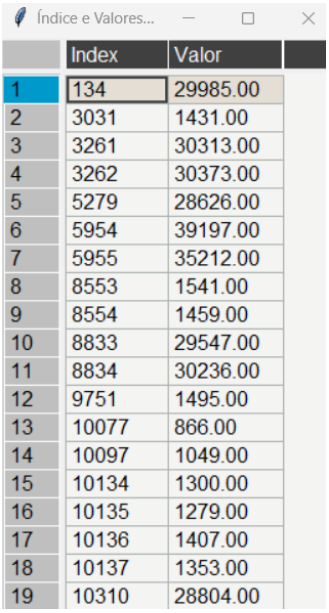


Figura 13 – Resultados do método LOF



	Index	Valor
1	134	29985.00
2	3031	1431.00
3	3261	30313.00
4	3262	30373.00
5	5279	28626.00
6	5954	39197.00
7	5955	35212.00
8	8553	1541.00
9	8554	1459.00
10	8833	29547.00
11	8834	30236.00
12	9751	1495.00
13	10077	866.00
14	10097	1049.00
15	10134	1300.00
16	10135	1279.00
17	10136	1407.00
18	10137	1353.00
19	10310	28804.00

Figura 14 – Resultados do método LOF

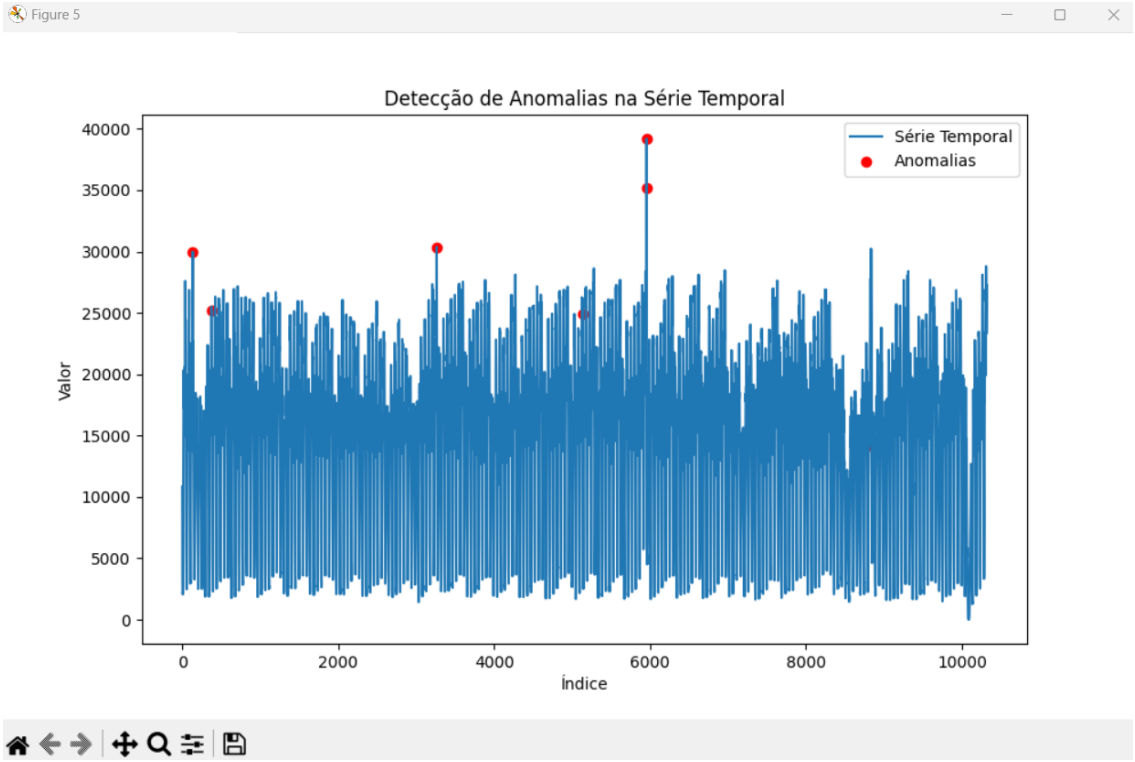


Figura 15 – Resultados do método de Decomposição Sazonal

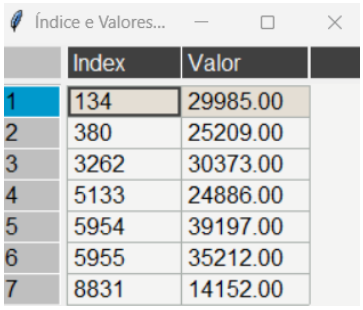


Figura 16 – Resultados do método de Decomposição Sazonal

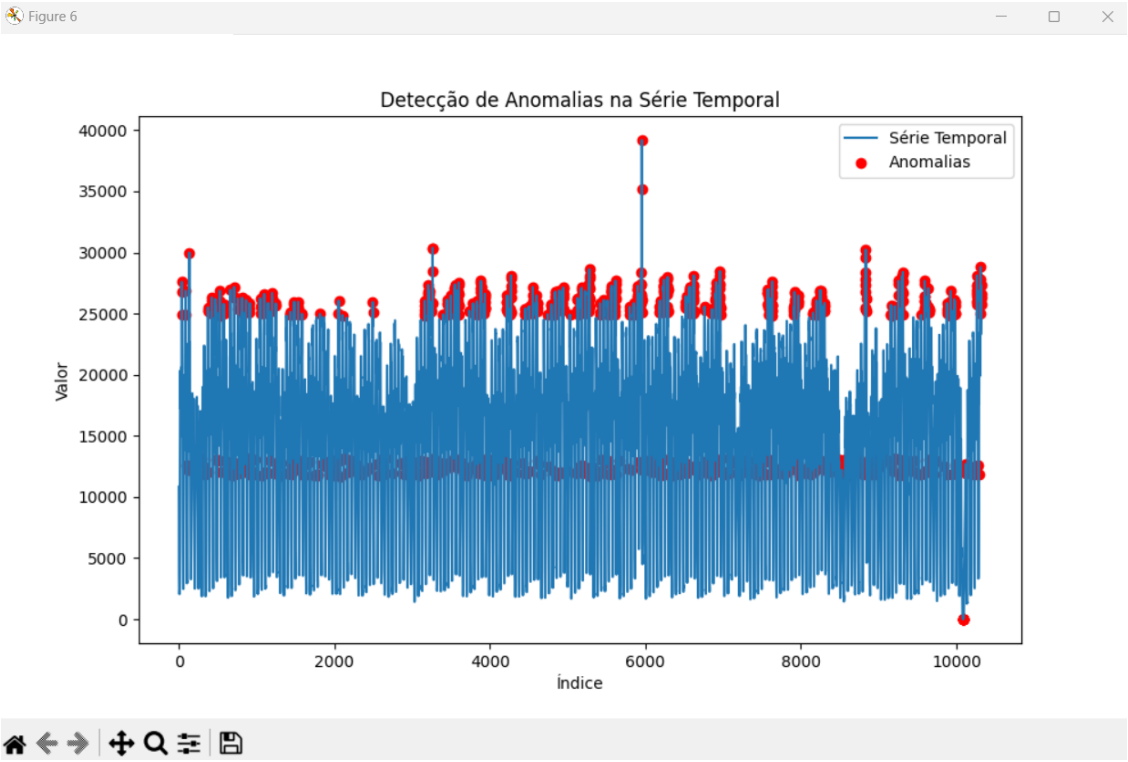


Figura 17 – Resultados do método K-Means

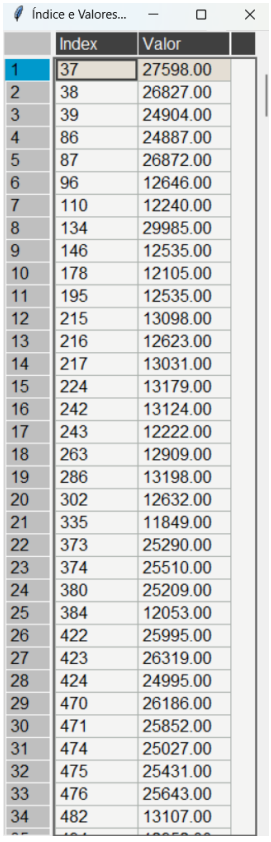


Figura 18 – Resultados do método K-Means

Referências

ALABADI, M.; CELIK, Y. Anomaly detection for cyber-security based on convolution neural network : A survey. In: . [S.l.: s.n.], 2020. p. 1–14. Citado na página 2.

ERHAN, L. et al. Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, v. 67, p. 64–79, 2021. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253520303717>. Citado na página 2.

HILAL, W.; GADSDEN, S.; YAWNEY, J. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, v. 193, p. 116429, 12 2021. Citado na página 2.

LEI, L. et al. A dynamic anomaly detection method of building energy consumption based on data mining technology. *Energy*, v. 263, p. 125575, 2023. ISSN 0360-5442. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360544222024616>. Citado na página 2.

PANG, G. et al. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 54, n. 2, p. 1–38, mar. 2021. ISSN 1557-7341. Disponível em: <http://dx.doi.org/10.1145/3439950>. Citado na página 2.

SALEM, O.; LIU, Y.; MEHAOUA, A. Anomaly detection in medical wireless sensor networks. *Journal of Computing Science and Engineering*, Korean Institute of Information Scientists and Engineers, v. 7, n. 4, p. 272–284, 2013. Citado na página 2.