UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Davi Rocha Cardoso

Técnicas de análise em séries temporais para detectar outliers em aplicações de monitoramento ambiental

Florianópolis
2024

Davi Rocha Cardoso

Técnicas de análise em séries temporais para detectar outliers em aplicações de monitoramento ambiental

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Carlos Barros Montez, Dr.

Florianópolis
2024

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.
Orientações em:
http://portalbu.ufsc.br/ficha

Davi Rocha Cardoso

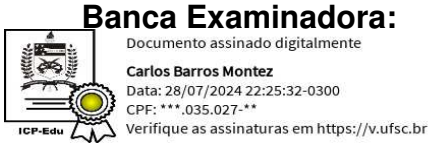
Técnicas de análise em séries temporais para detectar outliers em aplicações de monitoramento ambiental

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 9 de julho de 2024.

Prof. Marcelo de Lellis, Dr.
Coordenador do Curso

Banca Examinadora:



Prof. Carlos Barros Montez, Dr.
Orientador
UFSC/CTC/DAS

Prof. Carlos Barros Montez, Dr.
Supervisor
UFSC/CTC/DAS

Prof. João Paulo Zomer Machado,
Avaliador
UFSC/CTC/DAS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus familiares, à minha namorada, aos meus colegas de classe e ao meu orientador, cada um desempenhando um papel fundamental ao fornecer apoio e orientação essenciais para alcançar este objetivo.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão ao professor Carlos Montez, que, além de sugerir e definir o tema, esteve presente em todas as discussões ao longo do semestre. Sua participação foi fundamental para aumentar ainda mais meu interesse pela área de IoT e análise de dados. Graças ao seu incentivo e orientação, descobri um tema extremamente interessante, onde a análise de outliers se revelou uma área fascinante. A dedicação do professor Montez não apenas ampliou meus conhecimentos, mas também despertou uma paixão pela pesquisa e exploração de dados, elementos essenciais para minha formação acadêmica e futura carreira.

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 01 de julho de 2024.

Na condição de representante da UFSC na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que Davi Rocha Cardoso, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

Documento assinado digitalmente
Carlos Barros Montez
Data: 28/07/2024 22:41:37-0300
CPF: ***.035.027-**
Verifique as assinaturas em https://v.ufsc.br

Carlos Barros Montez
UFSC

RESUMO

O trabalho visa desenvolver uma solução abrangente para a detecção e tratamento de outliers em dados de redes de sensores sem fio (RSSF). Utilizando técnicas de aprendizado de máquina e estatísticas, como K-Nearest Neighbors (KNN), K-means, DBSCAN e Autoencoders, o estudo aplica essas metodologias a base de dados de estações metearológicas para avaliar a eficácia de cada técnica na identificação de anomalias. A pesquisa detalha a implementação e os resultados obtidos, demonstrando que os métodos propostos são eficazes na detecção de outliers, contribuindo para a melhoria da qualidade dos dados coletados. O trabalho destaca a importância da detecção e tratamento dos outliers para garantir a confiabilidade das medições e a precisão das análises em sistemas baseados em RSSF, além disso, entender o contexto em que os dados anômalos ocorrem, verificando se são erros ou eventos, locais ou globais.

Palavras-chave: Redes de Sensores Sem Fio, Detecção de Outliers, Dados.

ABSTRACT

The study aims to develop a comprehensive solution for the detection and treatment of outliers in wireless sensor network (WSN) data. Utilizing machine learning and statistical techniques, such as K-Nearest Neighbors (KNN), K-means, DBSCAN, and Autoencoders, the methodologies are applied to meteorological station databases to evaluate the effectiveness of each technique in identifying anomalies. The research details the implementation and results obtained, demonstrating that the proposed methods are effective in detecting outliers, contributing to the improvement of the quality of the collected data. The study highlights the importance of outlier detection and treatment to ensure the reliability of measurements and the accuracy of analyses in WSN-based systems, as well as understanding the context in which anomalous data occur, verifying whether they are errors or events, local or global.

Keywords: Wireless Sensor Networks, Outlier Detection, Data.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO DO PROBLEMA

Com o crescimento exponencial da Internet das Coisas (IoT), sensores sem fio têm sido amplamente adotados em diversas áreas, tais como monitoramento ambiental, saúde, segurança e automação industrial. Esses sensores são responsáveis pela coleta de uma vasta quantidade de dados em tempo real, proporcionando insights valiosos para a tomada de decisões. No entanto, a qualidade dos dados coletados pode ser comprometida pela presença de outliers, ou seja, leituras que se desviam significativamente do comportamento esperado dos dados.

A ocorrência de dados anômalos pode ser atribuída a vários fatores. Primeiramente, falhas nos sensores podem gerar leituras incorretas. Sensores podem apresentar defeitos ou desgaste ao longo do tempo, resultando em dados imprecisos. Problemas de calibração e desgaste físico são também fontes potenciais de erros nas medições coletadas.

Adicionalmente, interferências ambientais representam outra causa significativa de outliers. Condições ambientais adversas, como temperaturas extremas, umidade excessiva ou poluição, podem afetar o desempenho dos sensores. Mudanças súbitas no ambiente, como variações climáticas, também podem introduzir ruídos nas medições, comprometendo a precisão dos dados.

Erros de comunicação também são uma fonte de valores atípicos, redes de sensores sem fio dependem de uma comunicação estável para a transmissão de dados. Interferências eletromagnéticas e congestionamento da rede podem causar perdas de pacotes ou dados corrompidos. Problemas de sincronização e falhas na transmissão de dados também podem resultar em leituras anômalas.

Ataques maliciosos representam outro fator de preocupação. Redes de sensores são vulneráveis a ataques cibernéticos, como a injeção de dados falsos, ataques de negação de serviço (DoS) e interceptação de comunicações. Ataques deliberados podem introduzir anomalias com o objetivo de comprometer a integridade dos dados, levando a decisões incorretas baseadas em informações falsas.

Dessa forma, a presença de outliers compromete a integridade e a precisão dos dados, dificultando a análise e a tomada de decisões. Em aplicações críticas, como monitoramento de saúde e segurança, a detecção inadequada de anomalias pode resultar em consequências graves, incluindo danos materiais e riscos à vida humana.

Diante disso, além de tratar os dados anômalos é de extrema importância classifica-los, com o intuito de identificar padrões e características similares em pontos fora da curva, que em situações normais seriam descartados e sua relevância não seria considerada. A detecção e o tratamento eficaz de outliers em leituras de sensores são cruciais para garantir a confiabilidade dos dados coletados, a eficiência

dos sistemas baseados nesses dados e a segurança das operações realizadas com base nessas informações. A capacidade de identificar e mitigar os efeitos dos outliers permite uma análise mais precisa e robusta, essencial para o sucesso das diversas aplicações da IoT.

1.2 OBJETIVO GERAL

O trabalho tem como objetivo geral encontrar e desenvolver uma solução abrangente para a detecção e tratamento de outliers em leituras de sensores sem fio, integrando diversas técnicas avançadas de análise de dados. A proposta consiste na aplicação de técnicas existentes, na avaliação de seus resultados e na busca pelo aprimoramento dessas soluções para o conjunto de dados específico.

A primeira abordagem envolve o uso de modelos estatísticos. A aplicação de modelos de distribuição probabilística, como distribuições normais e não-normais, permitirá a identificação de valores atípicos com base na probabilidade de ocorrência. Esta abordagem se baseia no comportamento esperado dos dados para detectar anomalias que se desviam significativamente do padrão estabelecido.

Em termos de aprendizado de máquina, serão exploradas tanto técnicas supervisionadas quanto não supervisionadas. Algoritmos como Support Vector Machines (SVM) e redes neurais artificiais serão utilizados para treinar modelos capazes de classificar dados normais e anômalos com alta precisão. Esses modelos são treinados com conjuntos de dados rotulados para aprender a distinguir entre padrões normais e outliers, garantindo uma maior acurácia na detecção. Por outro lado, técnicas de clustering, como K-means e DBSCAN, serão implementadas para detectar grupos de dados discrepantes sem a necessidade de rótulos pré-definidos. Esses métodos agrupam dados semelhantes e identificam aqueles que não pertencem a nenhum grupo como outliers, oferecendo uma abordagem flexível e adaptativa para a detecção de anomalias. A combinação de diferentes abordagens de aprendizado de máquina, ou algoritmos híbridos, também será explorada para melhorar a robustez e a precisão na detecção de anomalias.

O trabalho visa não apenas propor uma solução que auxilie na detecção e classificação dos outliers, mas também avaliar a eficácia dessas técnicas no contexto específico dos dados coletados por sensores. Através de testes rigorosos e validações com dados reais, o objetivo é identificar as metodologias mais eficazes e propor aprimoramentos que possam aumentar a confiabilidade e a precisão dos sistemas de monitoramento.

1.3 ESTRUTURA DO DOCUMENTO

Este documento está organizado em seis capítulos principais, cada um abordando um aspecto crucial do estudo sobre a detecção de outliers em redes de sensores sem fio (RSSF). O Capítulo 1 apresenta a introdução, a visão geral do problema e os objetivos do estudo. O Capítulo 2 foca na análise e revisão das principais técnicas de análise de dados, tais como abordagens estatísticas, técnicas baseadas em vizinhança, métodos de clusterização, técnicas de classificação e abordagens de decomposição espectral. O Capítulo 3 contextualiza a base de dados utilizada no trabalho. O Capítulo 4 descreve as técnicas aplicadas e os resultados obtidos, detalhando os parâmetros utilizados. O Capítulo 5 traz comparações entre as técnicas e as leituras de cada sensor. O Capítulo 6 expõe as conclusões e sugere trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 REDES DE SENSORES SEM FIO (RSSF)

As redes de sensores sem fio representam um avanço significativo em relação aos sensores tradicionais devido à sua capacidade de coletar, processar e transmitir dados de maneira eficiente e autônoma. Esses sensores são componentes multifuncionais e de baixo custo que podem ser implantados em uma variedade de ambientes e aplicações, como saúde, monitoramento industrial e vigilância.

2.1.1 Estrutura das Redes de Sensores Sem Fio

Uma rede de sensores é composta por um grande número de nós distribuídos de maneira densa em uma área de interesse. Esses nós são capazes de realizar tarefas de sensoriamento, processamento de dados e comunicação sem fio em distâncias curtas. A comunicação entre os nós é frequentemente organizada em uma arquitetura de múltiplos saltos, onde os dados são transmitidos de nó em nó até alcançar um ponto central de coleta, conhecido como *sink* (AKYILDIZ *et al.*, 2002).

Os nós sensores são equipados com unidades de sensoriamento, processamento, transmissão e uma unidade de energia. A unidade de sensoriamento geralmente consiste em sensores específicos e conversores analógico-digitais que transformam sinais analógicos em digitais para processamento. A unidade de processamento, frequentemente associada a uma pequena unidade de armazenamento, gerencia as tarefas do sensor e a comunicação com outros nós. A unidade de transmissão conecta o nó à rede, enquanto a unidade de energia é crucial para a operação do sensor (AKYILDIZ *et al.*, 2002).

2.2 OUTLIERS

Na análise de dados, frequentemente nos deparamos com observações que se destacam significativamente do padrão geral dos dados. Essas observações são conhecidas como outliers ou valores atípicos. A definição intuitiva de um outlier é "uma observação que se desvia tanto das outras observações a ponto de suscitar suspeitas de que foi gerada por um mecanismo diferente"(HAWKINS, D. M., 1980). A detecção de outliers é crucial em muitas áreas de pesquisa, pois essas observações podem indicar erros de medição, variações naturais extremas, ou eventos raros e significativos. Um oulier também pode ser descrito como "uma observação (ou conjunto de observações) que parece inconsistente com o restante do conjunto de dados"(BARNETT; LEWIS, 1994).

Ambas as abordagens serão utilizadas para embasar as análises nos capítulos posteriores, é importante entender que um dado anômalo pode ter diversas motivações

para ocorrer, com isso, surge a importância de analisar e classificar a sua origem.

2.2.1 Tipos de Outliers

A identificação e tratamento de outliers são fundamentais para análises estatísticas e tomadas de decisão baseadas em dados. Esta seção apresenta uma visão geral desses tipos de anomalias.

2.2.1.1 Outliers Gerados por Distribuições de Cauda Pesada

Alguns dados anômalos surgem naturalmente em amostras de distribuições com cauda pesada, como a distribuição t de Student. Essas distribuições são caracterizadas por caudas que diminuem lentamente, o que aumenta a probabilidade de observar valores extremos. Hawkins classifica as distribuições em dois tipos principais: propensas a outliers e resistentes. As distribuições propensas a outliers têm maior probabilidade de gerar valores extremos, enquanto as resistentes tendem a produzir dados mais concentrados (HAWKINS, D. M., 1980).

2.2.1.2 Outliers Causados por Distribuições Contaminantes

Outro mecanismo que gera dados anormais envolve a presença de duas distribuições: uma distribuição básica que gera observações "boas"e uma distribuição contaminante que gera "contaminantes". Quando a distribuição contaminante tem caudas mais pesadas do que a distribuição básica, os contaminantes tendem a se separar visivelmente das boas observações, tornando-se outliers. Esse mecanismo pode ser subdividido em dois tipos:

- **Mecanismo (iia)**: Especifica que em uma amostra de tamanho *n*, exatamente *n–k* observações vêm da distribuição básica e *k* vêm da distribuição contaminante (HAWKINS, D. M., 1980).
- **Mecanismo (iib)**: As observações contaminadas ocorrem em pontos específicos do tempo, como em séries temporais, onde um evento específico causa uma mudança abrupta nos dados (HAWKINS, D. M., 1980).

2.2.1.3 Outliers Multivariados

Para dados multivariados, a detecção de outliers se complica, pois envolve a análise de várias dimensões simultaneamente. Hawkins discute métodos para lidar com outliers multivariados, como a análise de componentes principais (PCA) e testes baseados em distribuições multivariadas. Ele propõe que os outliers multivariados podem seguir várias distribuições alternativas que diferem em média, variância ou ambos. Os modelos considerados incluem:

- **Modelo** H_1: Outliers seguem uma distribuição normal multivariada com uma média diferente.
- **Modelo** H_2: Outliers têm variância diferente da distribuição básica.
- **Modelo** H_3: Outliers seguem uma distribuição com diferentes covariâncias (HAWKINS, D. M., 1980).

2.2.1.4 Outliers em Séries Temporais

Outliers em séries temporais podem ser classificados em dois tipos principais, conforme discutido por Fox (FOX, 1972):

- **Tipo 1**: Apenas uma observação é contaminada, sem afetar as subsequentes.
- **Tipo 2**: Uma anomalia ocorre em um ponto específico, afetando todas as observações subsequentes.

2.2.1.5 Outliers em Modelos de Regressão

Em modelos de regressão, a presença de outliers pode distorcer significativamente os resultados. Hawkins sugere técnicas para identificar e tratar outliers em regressão linear, como a utilização de resíduos recursivos e métodos de seleção stepwise. Ele destaca a importância de avaliar a presença de outliers tanto nos dados independentes quanto nas variáveis dependentes para garantir a precisão do modelo (HAWKINS, D. M., 1980).

2.2.2 Motivos para a Geração de Outliers

Os outliers podem surgir por diversos motivos, sendo os mais comuns erros humanos, falhas instrumentais, variações naturais em populações, comportamento fraudulento, mudanças no comportamento dos sistemas ou falhas nos sistemas (HAWKINS, D. M., 1980). Por exemplo, um erro tipográfico cometido por um operador pode ser corrigido após notificação, restaurando a observação a um registro normal. Falhas na leitura de instrumentos, por sua vez, podem ser simplesmente excluídas dos dados analisados. Em levantamentos populacionais, anomalias como a presença de indivíduos extremamente altos são consideradas naturais, mas ainda assim devem ser verificadas para garantir a ausência de erros antes de serem incluídas na análise.

Além disso, no caso de leituras de sensores os dados atípicos podem ser gerados por alterações climáticas, desastres ambientais, poluições e demais intemperes que a natureza possa ocasionar. Os dados anômalos registrados em ambientes críticos de segurança, sistemas de detecção de fraudes, análise de imagens ou monitoramento de intrusões devem ser detectados imediatamente e tratados com a devida atenção.

Uma vez identificado o problema, a leitura anômala pode ser armazenada separadamente para comparações futuras, mas geralmente não é mantida com os dados principais, já que esses sistemas tendem a modelar a normalidade e utilizar essa base para detectar anomalias.

2.2.3 Detecção de *Outliers*

A detecção de dados anômalos pode ser dividida em três grandes categorias de análise: dados com ruídos e erros (detecção de falhas), eventos (detecção de eventos) e ataques maliciosos (detecção de intrusos) (ZHANG *et al.*, 2010). Na maioria dos casos, os outliers gerados por erros são descartados. Portanto, a análise para entender as causas desses fenômenos tende a se concentrar na exploração e no tratamento dos dados classificados como eventos, que contêm informações relevantes e que poderiam ser perdidas se fossem tratados da mesma forma que os erros.

2.2.3.1 Erros

Os erros referem-se a dados ruidosos ou medições provenientes de sensores defeituosos. Esses outliers ocorrem frequentemente e são normalmente representados por mudanças arbitrárias e extremamente diferentes dos outros dados. É crucial identificar e corrigir esses dados, se possível, para manter a qualidade dos dados sensoriados (CHEN; KHER; SOMANI, 2006). As técnicas estatísticas, como modelos baseados em distribuições gaussianas, são comumente usadas para detectar e eliminar esses erros.

2.2.3.2 Eventos

Eventos são fenômenos específicos que alteram o estado do mundo real, como incêndios florestais, derramamentos químicos ou poluição do ar. Esses outliers geralmente duram por um período relativamente longo e mudam o padrão histórico dos dados dos sensores (KRISHNAMACHARI; IYENGAR, 2004). A detecção de eventos é fundamental para muitas aplicações de RSSFs, pois permite que ações sejam tomadas rapidamente em resposta a eventos significativos. Técnicas baseadas em correlações espaciais e temporais são frequentemente utilizadas para distinguir entre eventos reais e dados ruidosos (MARTINCIC; SCHWIEBERT, 2006).

2.2.3.3 Ataques

Os ataques maliciosos em Redes sem Fio visam interromper o funcionamento da rede ou obter acesso não autorizado aos dados. Exemplos incluem ataques de negação de serviço, ataques de buraco negro e escutas clandestinas (PERRIG; STANKOVIC; WAGNER, 2004). Detectar outliers causados por ataques é crucial para garantir

a segurança e a integridade da rede. Técnicas de detecção de intrusão frequentemente utilizam algoritmos de aprendizado de máquina e modelos estatísticos para identificar padrões suspeitos de comportamento dos nós da rede (BHUSE; GUPTA, 2006).

2.3 TÉCNICAS DE TRATAMENTO DE OUTLIERS EM REDES DE SENSORES SEM FIO

A detecção e tratamento de outliers em redes de sensores sem fio (RSSF) são essenciais para garantir a precisão e a confiabilidade dos dados coletados. Este capítulo examina as principais técnicas utilizadas, que se dividem em abordagens estatísticas, baseadas em vizinhança, clustering, classificação e decomposição espectral.

2.3.1 Abordagens Estatísticas

2.3.1.1 Modelos Paramétricos

Os modelos paramétricos assumem que os dados seguem uma distribuição específica, como a normal. Técnicas baseadas em modelos gaussianos identificam outliers ao comparar os dados com a distribuição esperada. Por exemplo, medições que se desviam significativamente da média são consideradas outliers (ZHANG *et al.*, 2010; CHEN; KHER; SOMANI, 2006).

2.3.1.2 Modelos Não-Paramétricos

Estas técnicas não assumem uma distribuição específica dos dados. Exemplos incluem o uso de histogramas e estimadores de densidade de kernel para detectar outliers com base na densidade dos dados em comparação a um modelo de referência. Essas abordagens são úteis quando a distribuição dos dados não é conhecida ou é complexa (HODGE; AUSTIN, 2004; PALPANAS *et al.*, 2003).

2.3.2 Abordagens Baseadas em Vizinhança

Estas técnicas analisam uma instância de dados em relação aos seus vizinhos mais próximos, baseando-se em medidas de similaridade.

2.3.2.1 Técnicas de Distância

Utilizam medidas como a distância Euclidiana para determinar se uma instância é um outlier com base na sua distância em relação aos vizinhos mais próximos. A técnica k-NN (k-nearest neighbors) é amplamente utilizada, onde um ponto é considerado outlier se estiver distante de seus k vizinhos mais próximos (ZHANG *et al.*, 2010; BRANCH *et al.*, 2006).

2.3.2.2 Análise de Similaridade Temporal

Avaliam as correlações temporais dos dados para identificar anomalias que se desviam das tendências temporais esperadas (CHEN; KHER; SOMANI, 2006).

2.3.3 Abordagens de Clustering

As técnicas de clustering agrupam dados em clusters baseados na similaridade. Dados que não pertencem a nenhum cluster ou pertencem a clusters significativamente menores são considerados outliers.

2.3.3.1 K-means e DBSCAN

O K-means é um algoritmo popular que agrupa dados em k clusters, minimizando a variabilidade dentro de cada cluster. O DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é outro algoritmo amplamente utilizado que detecta outliers como pontos que não se ajustam a nenhuma região densa, ou seja, regiões de alta densidade de dados (KRISHNAMACHARI; IYENGAR, 2004; RAJASEGARAR *et al.*, 2006).

2.3.3.2 Clustering Hierárquico

O clustering hierárquico cria uma hierarquia de clusters, permitindo a análise de dados em diferentes níveis de granularidade. Este método constrói uma árvore de clusters, conhecida como dendrograma, onde cada nó representa um cluster composto por seus subclusters (MARTINCIC; SCHWIEBERT, 2006).

2.3.4 Abordagens de Classificação

Essas técnicas utilizam modelos de aprendizado supervisionado ou não supervisionado para classificar dados como normais ou outliers.

2.3.4.1 Máquinas de Vetores de Suporte (SVM)

As SVMs utilizam hiperplanos para separar classes de dados. Dados fora da região definida pelo hiperplano são considerados outliers. As SVMs são eficazes na detecção de outliers devido à sua capacidade de manejar grandes espaços de características e identificar margens máximas entre classes (PERRIG; STANKOVIC; WAGNER, 2004; RAJASEGARAR; LECKIE; PALANISWAMI, 2007).

2.3.4.2 Redes Bayesianas

As redes Bayesianas empregam modelos probabilísticos para representar dependências entre variáveis. Elas classificam dados com base nas probabilidades cal-

culadas, sendo úteis para lidar com incertezas e variabilidades nos dados (BHUSE; GUPTA, 2006).

2.3.5 Abordagens de Decomposição Espectral

Estas técnicas utilizam a análise de componentes principais (PCA) para reduzir a dimensionalidade dos dados e encontrar modos normais de comportamento.

2.3.5.1 Análise de Componentes Principais (PCA)

A PCA identifica componentes principais que capturam a maior variabilidade dos dados. Dados que violam esta estrutura são considerados outliers. A PCA é eficaz para detectar outliers em conjuntos de dados de alta dimensionalidade, pois reduz a complexidade do problema (CHATZIGIANNAKIS; PAPAVASSILIOU; GRAMMATIKOU, 2006).

2.3.6 Abordagem de Redes Neurais

A detecção de outliers utilizando redes neurais, especificamente Redes Neurais Replicadoras (Replicator Neural Networks, RNNs), oferece uma abordagem poderosa e eficaz para identificar anomalias em grandes bancos de dados multivariados. A seguir, descrevemos a aplicação desta técnica, conforme apresentado por Hawkins (HAWKINS, S. *et al.*, 2002).

2.3.6.1 Redes Neurais Replicadoras (RNRs)

As Redes Neurais Replicadoras são um tipo de rede neural que utiliza os mesmos dados de entrada como dados de saída, permitindo a criação de um modelo comprimido dos dados durante o treinamento. O principal objetivo das RNR é minimizar o erro de reconstrução, que é a diferença entre os valores de entrada e os valores reconstruídos na saída.

2.3.6.2 Arquitetura da RNR

- **Camadas**: Uma RNR típica possui uma camada de entrada, três camadas ocultas e uma camada de saída, onde o número de unidades nas camadas de entrada e saída corresponde ao número de características dos dados.
- **Função de Ativação**: Utiliza funções de ativação específicas, como a tangente hiperbólica (tanh) para as camadas ocultas externas e uma função em degraus para a camada oculta central, o que ajuda a quantizar os dados em valores discretos e facilita a compressão dos dados.

2.3.6.3 Treinamento da RNR

- **Minimização do Erro de Reconstrução**: Durante o treinamento, os pesos da rede são ajustados para minimizar o erro médio quadrático entre os dados de entrada e de saída.
- **Taxa de Aprendizagem Adaptativa**: Utiliza uma taxa de aprendizagem que se ajusta automaticamente para garantir a convergência do erro de reconstrução.

2.3.6.4 Metodologia de Aplicação da RNR para Detecção de Outliers

- **Fator de Outlier (OF)**: O Fator de Outlier é definido como o erro médio de reconstrução de todos os recursos (características) de um registro de dados.

$$OF_i = \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - o_{ij})^2 \tag{1}$$

  onde n é o número de características, x_{ij} é o valor de entrada e o_{ij} é o valor de saída.

- **Tratamento de Variáveis Categóricas**: Divisão do conjunto de dados em subconjuntos com base nos valores das variáveis categóricas e treinamento de uma RNN individual para cada subconjunto.
- **Amostragem e Treinamento**: Para grandes conjuntos de dados, é utilizada uma amostragem para selecionar um subconjunto representativo dos dados para treinar a RNN.
- **Aplicação da RNR Treinada**: A RNR treinada é usada para calcular o Fator de Outlier para todos os pontos de dados, permitindo a identificação e priorização dos outliers com base no erro de reconstrução.

3 CONJUNTO DE DADOS E CONTEXTO

Este capítulo descreve a base de dados utilizada nas técnicas de análise de dados. Essa base de dados é composta por um conjunto de estações meteorológicas, equipadas com diversos tipos de sensores, instaladas em Chicago, Estados Unidos.

3.1 DESCRIÇÃO DO DATASET

O dataset utilizado neste estudo provém das estações meteorológicas automatizadas instaladas nas praias ao longo da orla do Lago Michigan em Chicago. Este dataset é mantido pelo *Chicago Park District* e captura medições horárias. Esses sensores geralmente capturam as medições indicadas de hora em hora enquanto estão em operação. É importante notar que, durante a pandemia, devido à falta de manutenção, os sensores passaram por períodos sem medições.As principais variáveis medidas pelos sensores incluem:

- **Temperatura do Ar (℃)**: Captura a temperatura ambiente nas proximidades das praias.
- **Umidade**: Mede a quantidade de vapor de água presente no ar.
- **Pressão Barométrica(hPa)**: Indica a pressão atmosférica na área.
- **Precipitação (mm)**: Registra a quantidade de chuva em intervalos específicos.
- **Radiação Solar (W/m^2)**: Mede a intensidade da radiação solar.
- **Velocidade do Vento (m/s)**: Monitora a velocidade do vento nas praias.

Esses sensores são cruciais para fornecer dados precisos e em tempo real que ajudam na gestão dos recursos das praias e na segurança pública (DISTRICT, 2015).

3.1.1 Sensores Utilizados

Os dados do dataset são coletados por diversas estações meteorológicas distribuídas ao longo da costa do Lago Michigan. Cada estação de monitoramento é identificada por uma coluna chamada *beach station*, que indica a localização específica da estação de coleta de dados. Cada estação está equipada com múltiplos sensores que capturam diferentes parâmetros ambientais. As estações incluem, mas não se limitam a:

- **North Avenue Beach Station**: Monitora condições ambientais em uma das praias mais populares de Chicago.
- **Oak Street Beach Station**: Coleta dados cruciais para a área central de Chicago.

- **Montrose Beach Station**: Focada em uma área de alta atividade recreativa.
- **63rd Street Beach Station**: Serve a comunidade no sul de Chicago.
- **Calumet Beach Station**: Monitora as condições em uma das áreas mais ao sul da orla de Chicago.

Cada uma dessas estações está equipada com sensores avançados para capturar dados de temperatura, umidade, pressão barométrica, precipitação, radiação solar e velocidade do vento, fornecendo um panorama abrangente das condições ambientais ao longo da costa.

3.1.2 Contexto e Importância

As medições meteorológicas em tempo real são fundamentais para uma variedade de aplicações, incluindo:

- **Segurança Pública**: Dados precisos sobre condições meteorológicas podem ajudar a prevenir acidentes e orientar os banhistas sobre as condições do tempo e da água.
- **Gestão de Recursos**: Informações sobre clima e condições ambientais ajudam na manutenção e planejamento das atividades nas praias.
- **Pesquisa e Análise**: Os dados coletados podem ser utilizados para pesquisas científicas, análise de tendências climáticas e estudos ambientais.

3.1.3 Desafios e Tratamento de Outliers

A coleta de dados em ambientes externos como praias está sujeita a várias fontes de erro e variabilidade, incluindo interferências ambientais, falhas nos sensores e erros de comunicação. A detecção e o tratamento de outliers são essenciais para garantir a integridade dos dados.

- **Erros de Sensores**: Falhas ou mal funcionamento dos sensores podem resultar em leituras anômalas que precisam ser identificadas e tratadas.
- **Interferências Ambientais**: Condições adversas, como tempestades e mudanças bruscas no clima, podem introduzir ruídos nos dados coletados.
- **Erros de Comunicação**: Problemas na transmissão de dados podem causar inconsistências que devem ser filtradas.

Para lidar com esses desafios, são empregadas técnicas de detecção de outliers, como modelos estatísticos e algoritmos de aprendizado de máquina, que ajudam a identificar e corrigir essas anomalias, assegurando a qualidade dos dados para análises e tomadas de decisão (DISTRICT, 2016).

4 APLICAÇÃO DAS TÉCNICAS À BASE DE DADOS

Neste capítulo, serão aplicados, na base de dados exposta no capítulo 3, os principais métodos para tratamento de outliers mencionados na revisão da literatura. A análise será conduzida com base na acurácia dos algoritmos, e posteriormente, os resultados serão comparados entre si. O objetivo é identificar a maneira mais eficaz e prática para tratar os dados anômalos, garantindo a integridade e a precisão do conjunto de dados.

4.1 TRATAMENTO DE DADOS

Para garantir a melhor aplicabilidade das técnicas, alguns tratamentos foram realizados na base de dados. A retirada de dados nulos e faltantes é um aspecto fundamental, considerando que a base de dados corresponde a um longo período de medições. É crucial verificar a integridade dos dados para evitar distorções nas análises subsequentes.

Além disso, para uma visualização apropriada das técnicas, foi escolhido um período de tempo reduzido para aplicação. Inicialmente, os modelos foram aplicados a todo o período amostral, mas para facilitar o entendimento e aprofundar nas técnicas, será utilizado o período de um mês. Especificamente, serão comparados os meses de julho dos anos de 2019, 2022 e 2023. O mês de julho foi escolhido por representar o verão na cidade de Chicago, quando as estações de medição estão em pleno funcionamento e há uma maior variabilidade nas leituras dos sensores. Essa escolha permite uma análise mais robusta e relevante das técnicas de detecção de outliers, uma vez que os dados refletem condições de operação intensiva das estações (DISTRICT, 2015, 2016).

4.2 MODELOS PARAMÉTRICOS

4.2.1 Z-Score

O Z-score é uma medida estatística que indica o número de desvios padrão que um dado valor está distante da média da amostra ou população. Ele é amplamente utilizado para detectar outliers em conjuntos de dados, especialmente quando os dados seguem uma distribuição normal. O Z-score é calculado usando a fórmula Equação (2):

$$Z_i = \frac{(X_i - \mu)}{\sigma}. \tag{2}$$

onde:

- X é o valor da observação,
- μ é a média da amostra,

- σ é o desvio padrão da amostra.

Valores de Z-score altos (positivos ou negativos) indicam que a observação está distante da média e pode ser considerada um outlier. Comumente, um valor de Z-score maior que 3 ou menor que -3 é considerado um outlier (ROUSSEEUW; HUBERT, 2011).

4.2.1.1 Aplicação em RSSF

Os sensores, especialmente em redes de sensores sem fio (RSSF), são frequentemente usados para monitorar variáveis ambientais e operacionais. Esses sensores coletam dados continuamente, e a identificação de outliers é crucial para garantir a precisão e confiabilidade dos dados. O Z-score é uma técnica eficaz para detectar esses outliers.

1. **Monitoramento de Temperatura:** Sensores de temperatura instalados em diversos ambientes podem sofrer interferências devido a fatores externos como falhas nos sensores ou condições climáticas extremas. Usando o Z-score, é possível identificar leituras anômalas que se desviam significativamente da média, sinalizando possíveis problemas ou eventos excepcionais.
2. **Detecção de Umidade e Pressão:** Sensores de umidade e pressão também podem registrar valores anômalos devido a falhas técnicas ou mudanças súbitas no ambiente. Ao aplicar o Z-score, essas anomalias podem ser rapidamente identificadas, permitindo intervenções oportunas para corrigir possíveis problemas.
3. **Precipitação e Radiação Solar:** Sensores que medem a precipitação e a radiação solar são susceptíveis a variações significativas devido a eventos climáticos extremos. A aplicação do Z-score ajuda a isolar esses eventos extremos dos dados normais, melhorando a análise e a interpretação dos dados coletados.

4.2.1.2 Método do Z-score

O método do Z-score envolve os seguintes passos:

1. **Cálculo da Média (μ) e do Desvio Padrão (σ)**: Calcule a média e o desvio padrão dos dados. Estes serão usados para normalizar os dados e calcular o Z-score para cada observação.
2. **Cálculo do Z-score para Cada Observação**: Para cada observação X_i, calcule o Z-score usando a fórmula acima.
3. **Identificação de Outliers**: Compare os valores de Z-score com um limiar predefinido (no caso escolhido foi 3 e -3). Observações com Z-scores além desses limites são consideradas outliers.

4.2.1.3 Aplicação aos Dados

A detecção de outliers foi essencial para garantir a integridade dos dados, especialmente considerando fatores como falhas nos sensores e condições ambientais adversas.

Abaixo estão os passos detalhados aplicados aos dados:

1. **Cálculo da Média e do Desvio Padrão**: Para cada variável medida pelas estações (temperatura do ar, umidade, pressão barométrica, precipitação, radiação solar e velocidade do vento), calculei a média (μ) e o desvio padrão (σ).

2. **Cálculo do Z-score**: Para cada observação, utilizei a fórmula do Z-score para normalizar os dados. Isso envolveu subtrair a média da observação e dividir pelo desvio padrão Equação (3):

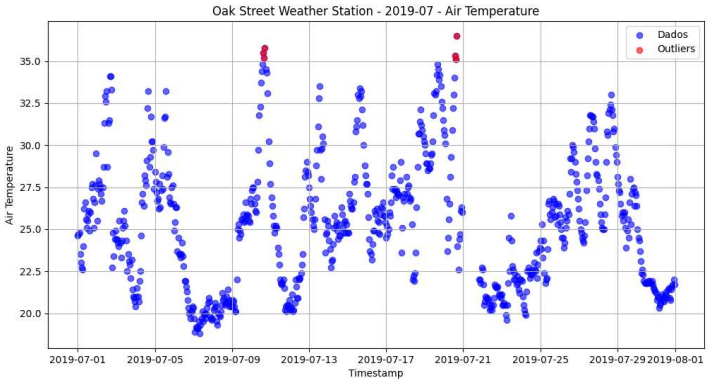
$$Z_i = \frac{(X_i - \mu)}{\sigma}. \tag{3}$$

Após aplicar a técnica de Z-score aos dados, deve-se prosseguir com a análise para verificar se a técnica aplicada fez sentido. Uma das várias técnicas para essa análise é a análise gráfica, que permite visualizar os outliers destacando-os dos pontos médios. Esse método facilita a identificação visual de outliers, como pode ser observado nas Figuras 1, 2, 3 e 4.

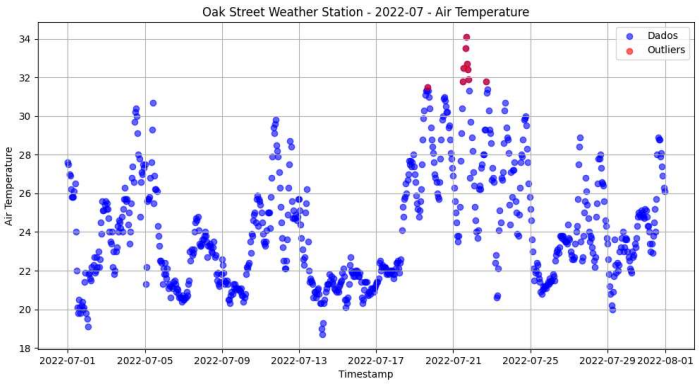
A análise gráfica é crucial porque fornece uma representação visual clara das anomalias, ajudando a confirmar se os outliers identificados pelo Z-score são realmente anômalos ou se são simplesmente variações normais dentro do conjunto de dados.

Esse método de visualização não só valida a eficácia do Z-score na identificação de outliers, mas também ajuda a contextualizar essas anomalias, permitindo uma análise mais aprofundada sobre as possíveis causas e impactos desses valores atípicos nos dados coletados.

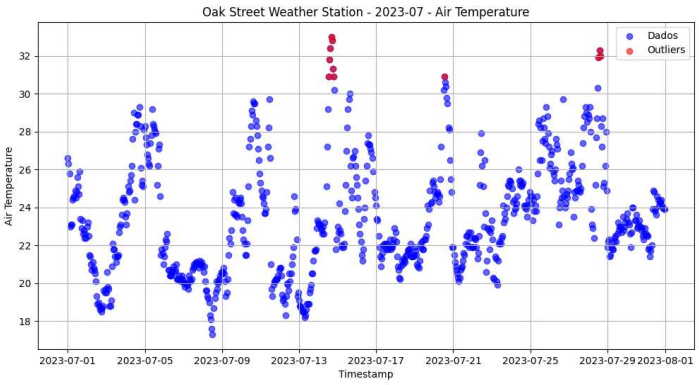
Destacando a Figura 2, é possível observar que a detecção dos outliers representam pontos em um mesmo dia em que as medições foram excepcionais, destacando que o método apresenta uma significativa acurácia. Porém, deve-se atentar que os dados destacados não necessariamente representam erros de medição e sim eventos anômalos, como por exemplo um grande intensidade de chuva na região em que o sensor está alocado.



2019



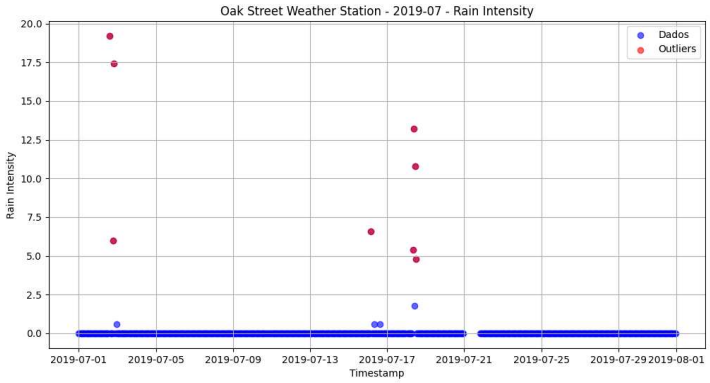
2022



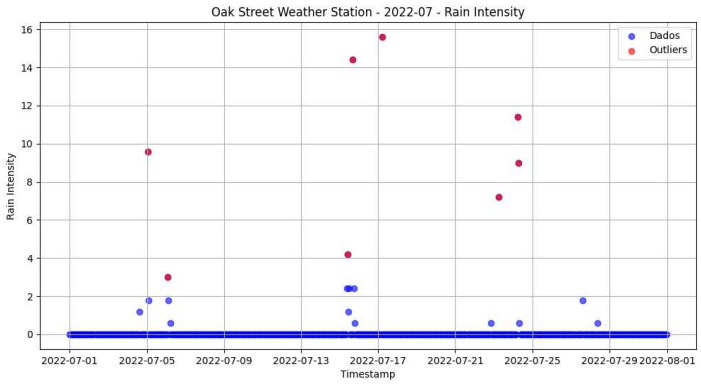
2023

Figura 1 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

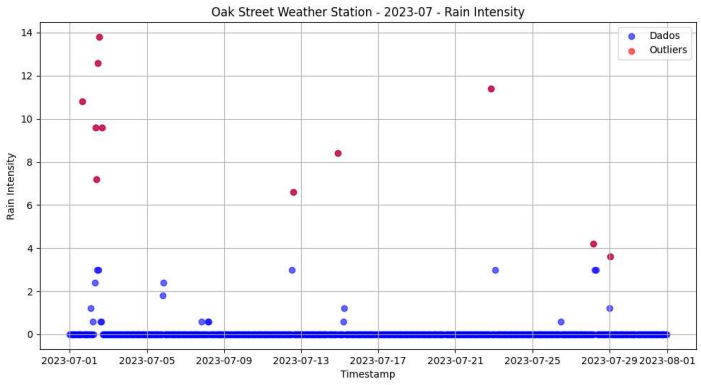
Fonte: Arquivo Pessoal.



2019



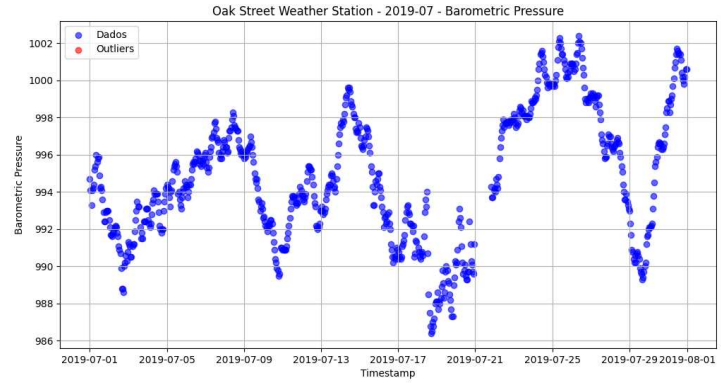
2022



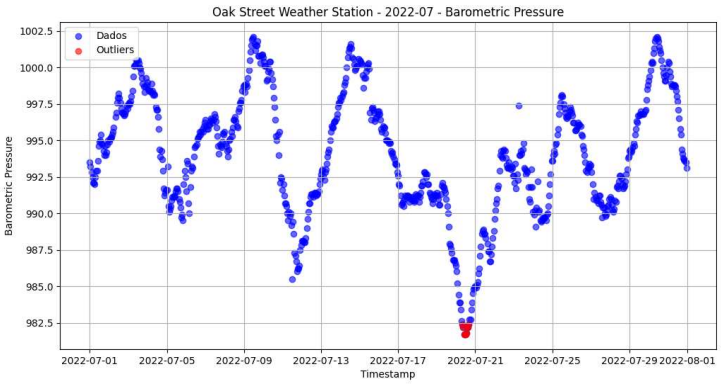
2023

Figura 2 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

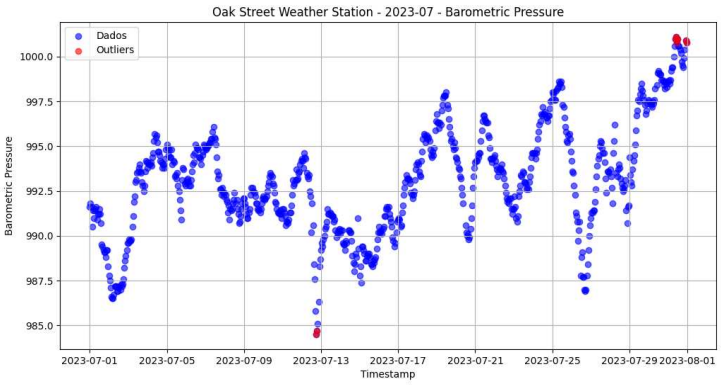
Fonte: Arquivo Pessoal.



2019



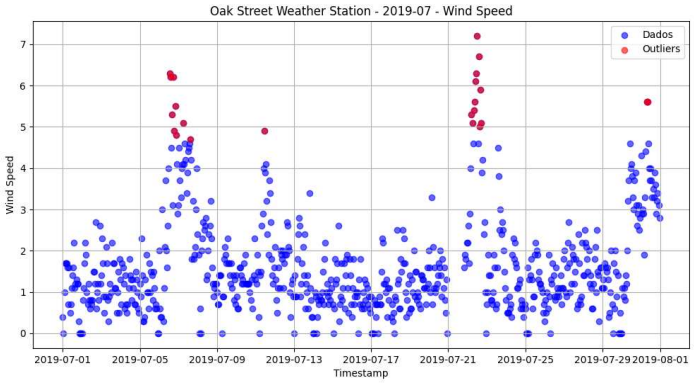
2022



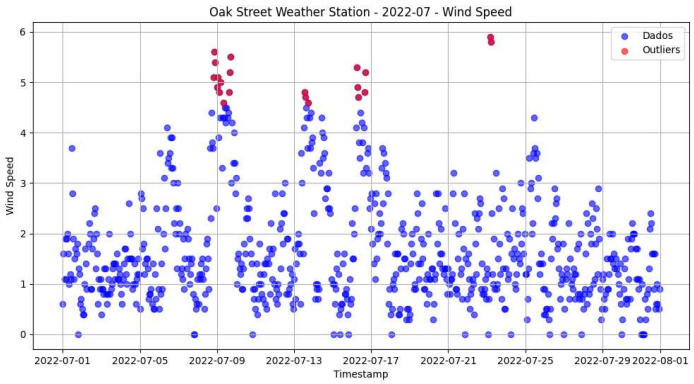
2023

Figura 3 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

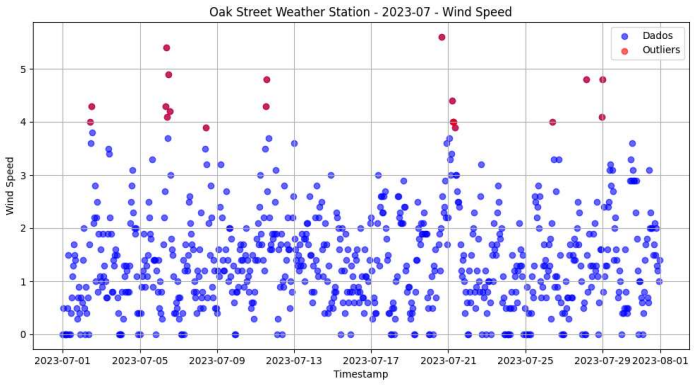
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 4 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

4.3 ABORDAGENS BASEADAS EM VIZINHANÇA

4.3.1 KNN

O K-Nearest Neighbors (KNN) é um algoritmo de aprendizado supervisionado utilizado tanto para classificação quanto para regressão. A ideia central do KNN é classificar uma nova observação com base nas *k* observações mais próximas no espaço

de características. O método considera a distância entre as observações, geralmente utilizando a distância Euclidiana, para determinar a similaridade entre os pontos de dados.

4.3.1.1 Funcionamento do KNN

O algoritmo KNN funciona através dos seguintes passos:

1. **Escolha do Parâmetro** k: Defina o número de vizinhos mais próximos (k) que serão considerados na análise.
2. **Cálculo das Distâncias**: Calcule a distância entre a nova observação e todas as observações no conjunto de dados.
3. **Identificação dos Vizinhos Mais Próximos**: Selecione as k observações mais próximas com base nas distâncias calculadas.

A distância Euclidiana, frequentemente usada no KNN, é calculada usando a fórmula Equação (4):

$$d(i,j) = \sqrt{\sum_{m=1}^{M} (x_i^m - x_j^m)^2}\,. \tag{4}$$

onde $d(i,j)$ é a distância entre os pontos i e j, M é o número de características, e x_i^m e x_j^m são os valores das m-ésimas características dos pontos i e j, respectivamente (COVER; HART, 1967).

4.3.1.2 Aplicação do KNN na Detecção de Outliers em Sensores

Na detecção de outliers em dados de sensores, o KNN pode ser utilizado para identificar observações que se desviam significativamente dos seus vizinhos mais próximos. Essa abordagem é eficaz porque sensores em uma rede geralmente produzem leituras similares em condições normais, e desvios significativos podem indicar anomalias.

A seguir, o exemplo de como foi aplicado KNN na detecção de outliers em um conjunto de dados de sensores, usando a biblioteca *scikit-learn*:

```
# Função para identificar outliers usando K-NN
def identificar_outliers_knn(data, columns, n_neighbors=50,
contamination=0.05):
    lof = LocalOutlierFactor(n_neighbors=n_neighbors,
    contamination=contamination)
    for column in columns:
        # Preparar os dados para o LOF
```

```
        column_data = data[[column]].dropna()
        if len(column_data) < n_neighbors:
            continue
        # Ajustar e prever outliers
        data.loc[column_data.index, f'{column}_outlier_KNN'] =
        lof.fit_predict(column_data)
        # Identificar outliers (LOF retorna -1 para outliers)
        data.loc[data.index, f'{column}_outlier_KNN'] =
        data[f'{column}_outlier_KNN'] == -1

    return data
```

Esta função aplica o algoritmo KNN para detectar outliers em colunas específicas de um dataframe *data*. Os seguintes parâmetros foram escolhidos:

- $n_{neighbors}$ = 50: Este parâmetro define o número de vizinhos mais próximos que o algoritmo considera ao calcular a pontuação de anomalia de cada ponto. O valor de 50 foi escolhido como um valor padrão para garantir que o algoritmo tenha uma quantidade suficiente de pontos de referência para determinar se um ponto é um outlier, através de tentativas de ajustar o parâmetro o valor de 50 demonstrou-se satisfatório para a análise. Um número maior de vizinhos proporciona uma avaliação mais robusta, mas pode ser ajustado conforme a densidade e o tamanho dos dados.

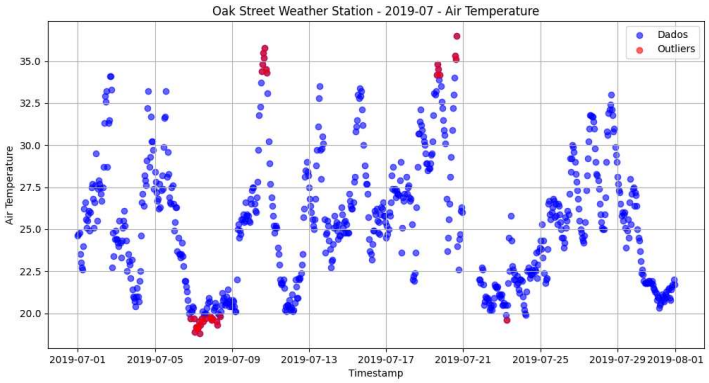
- *contamination* = 0.05: Este parâmetro representa a proporção de outliers esperados no conjunto de dados. Um valor de 0.05 indica que esperamos que 5% dos dados sejam outliers. Este é um valor comum para muitos conjuntos de dados, e apresentou um desempenho satisfatório quando aplicado ao conjunto.

4.3.1.3 Vantagens e Desvantagens

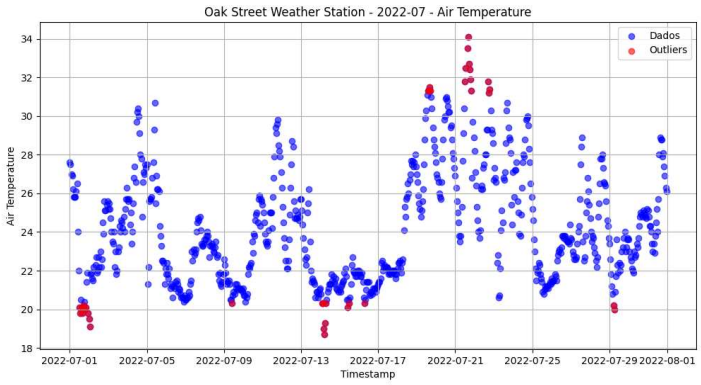
As vantagens incluem a simplicidade e a facilidade de implementação, além de ser eficaz para conjuntos de dados com estruturas complexas. No entanto, as desvantagens envolvem a sensibilidade à escolha do parâmetro k e à presença de ruído nos dados.

4.3.1.4 Aplicação aos Dados

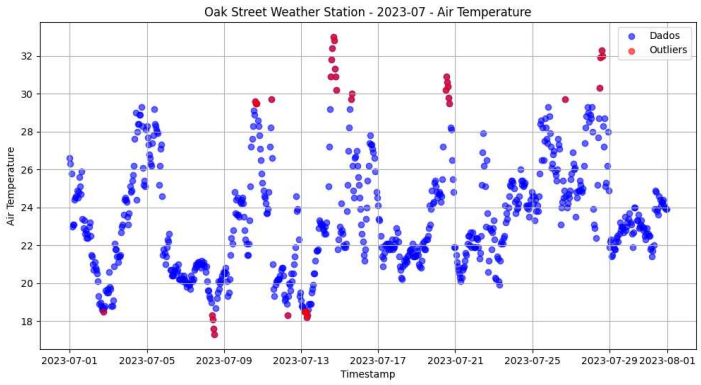
A aplicação do algoritmo KNN para análise de vizinhos foi realizada no dataset utilizando o código apresentado no Apêndice A. A implementação considerou os parâmetros citados anteriormente, e os resultados obtidos foram satisfatórios em termos de detecção de outliers. As métricas de desempenho serão discutidas em detalhes nas seções subsequentes. No entanto, a análise gráfica inicial, conforme ilustrado nas Figuras 9, 10, 11 e 12 já demonstra a eficácia do método. Na visualização gráfica, as médias das medições estão representadas em azul, enquanto os outliers identificados pelo método KNN estão destacados em vermelho, permitindo uma clara distinção entre os dados normais e anômalos.



2019



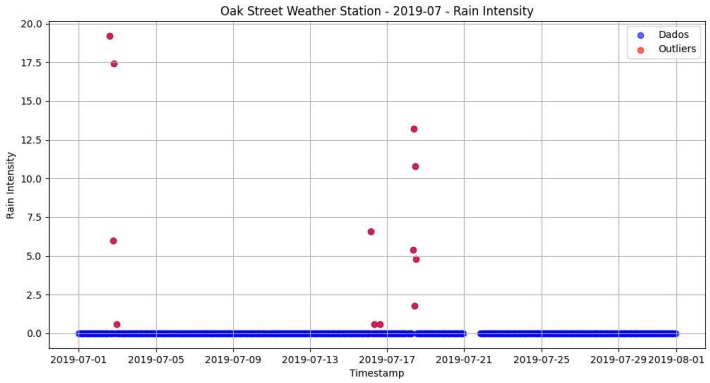
2022



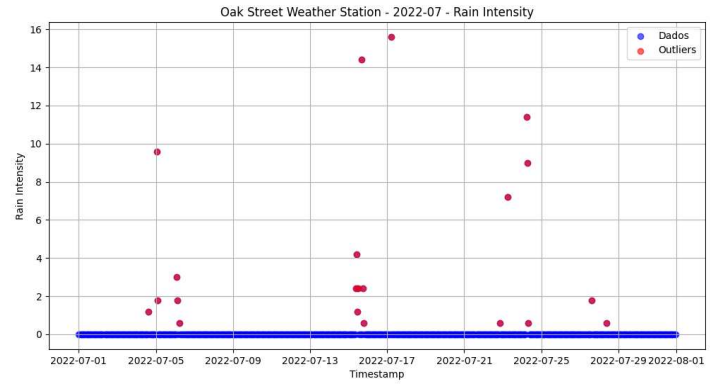
2023

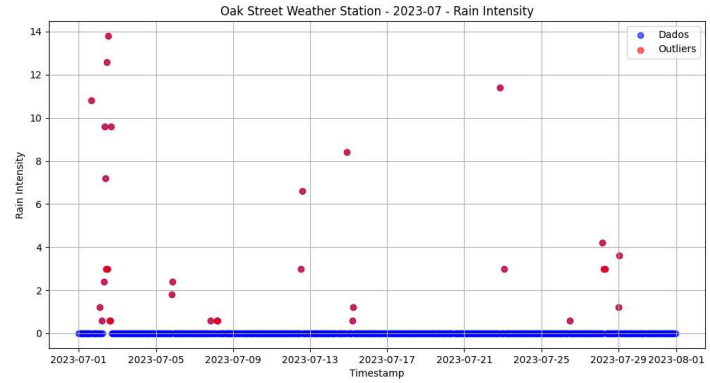
Figura 5 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.



2019

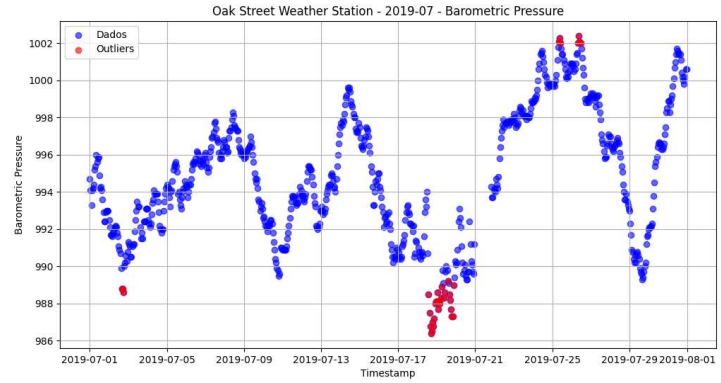




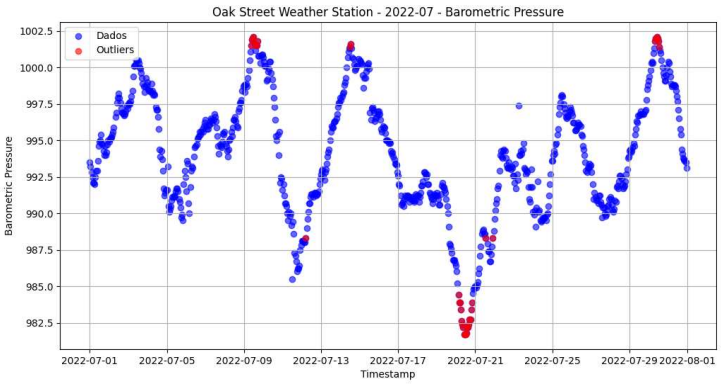
2023

Figura 6 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

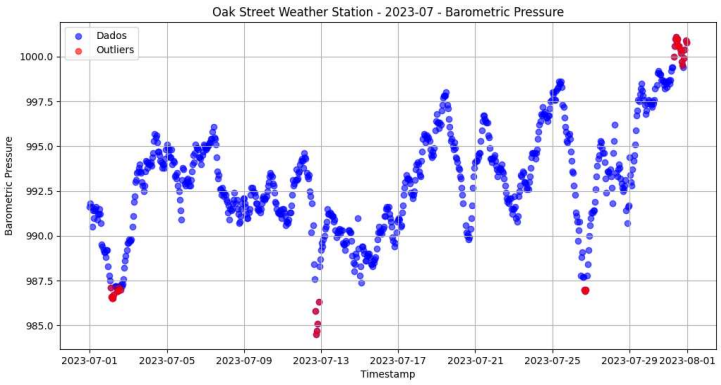
Fonte: Arquivo Pessoal.



2019



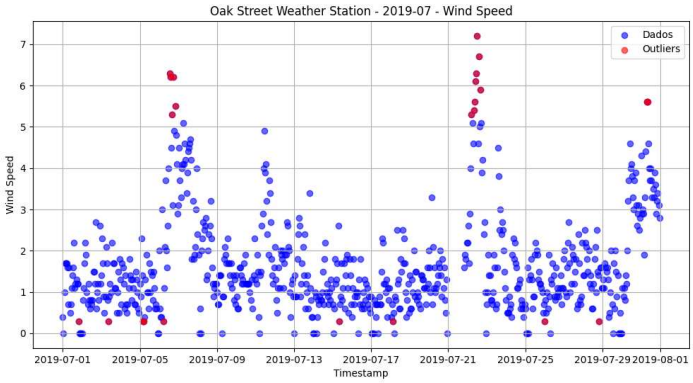
2022



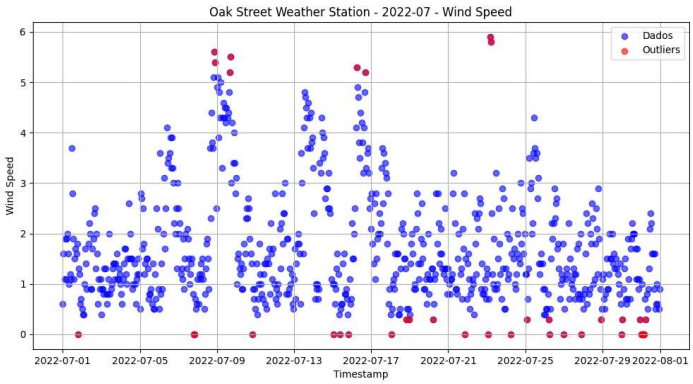
2023

Figura 7 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

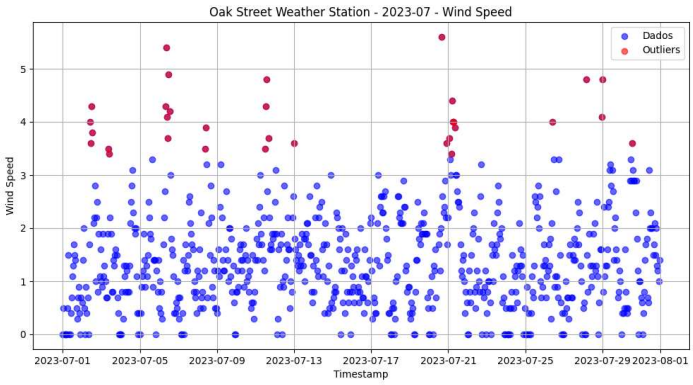
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 8 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

4.4 ABORDAGENS DE CLUSTERIZAÇÃO

4.4.1 K-means

O K-means é um algoritmo de aprendizado não supervisionado utilizado para particionar um conjunto de dados em *k* clusters, onde cada observação pertence ao cluster cujo centroide é o mais próximo. O objetivo do K-means é minimizar a variabilidade dentro de cada cluster, medindo a distância entre os pontos de dados e o centroide do cluster ao qual pertencem.

A detecção de outliers através de clusterização, como a realizada pelo K-means, baseia-se na ideia de que os outliers são pontos de dados que não se ajustam bem aos clusters formados. Este método é particularmente eficaz em detectar outliers globais, onde um ponto de dados é considerado anômalo em relação ao conjunto inteiro de dados (BREUNIG *et al.*, 2000).

4.4.1.1 Funcionamento do K-means

O algoritmo K-means funciona através dos seguintes passos:

1. **Inicialização**: Escolha *k* centroides iniciais, que podem ser selecionados aleatoriamente ou utilizando métodos como *K-means++*.
2. **Atribuição de Clusters**: Atribua cada observação ao cluster cujo centroide é o mais próximo, com base na distância Euclidiana.
3. **Atualização dos Centroides**: Calcule a nova posição de cada centroide como a média das observações atribuídas ao cluster correspondente.
4. **Convergência**: Repita os passos de atribuição e atualização até que os centroides não mudem significativamente ou até que um número máximo de iterações seja alcançado.

A fórmula da distância Euclidiana, frequentemente usada no K-means, é:

$$d(i,j) = \sqrt{\sum_{m=1}^{M} (x_i^m - x_j^m)^2} \tag{5}$$

onde $d(i,j)$ é a distância entre os pontos i e j, M é o número de características, e x_i^m e x_j^m são os valores das m-ésimas características dos pontos i e j, respectivamente (MACQUEEN, 1967).

4.4.1.2 Aplicação do K-means na Detecção de Outliers em Sensores

Na detecção de outliers em dados de sensores, o K-means pode ser utilizado para identificar observações que estão significativamente distantes dos centroides dos clusters. Observações que possuem grandes distâncias em relação aos seus centroides podem ser consideradas outliers.

A seguir, a função utilizada para K-means na detecção de outliers em um conjunto de dados de sensores, utilizando a biblioteca *scikit-learn*:

```
# Função para identificar outliers usando K-NN
def identificar_outliers_kmeans(data, columns, n_clusters=1):
    for column in columns:
        # Preparar os dados para o KMeans
        column_data = data[[column]].dropna()
        if len(column_data) < n_clusters:
            continue
        # Ajustar KMeans
        kmeans = KMeans(n_clusters=n_clusters, random_state=43)
        data.loc[column_data.index, f'{column}_cluster'] =
        kmeans.fit_predict(column_data)
        # Calcular distâncias dos pontos aos seus centróides
        distances = kmeans.transform(column_data)
        min_distances = np.min(distances, axis=1)
        # Identificar outliers com base na distância
        threshold_distance = np.percentile(min_distances, 99)
        data.loc[column_data.index, f'{column}_outlier_KMeans'] =
        min_distances > threshold_distance
    return data
```

Esta função aplica o algoritmo K-means para detectar outliers em colunas específicas de um dataframe *data*. O parâmetro $n_{clusters}$ define o número de clusters a serem utilizados na análise, e o limiar para considerar um ponto como outlier é definido como o 99 percentil das distâncias aos centroides.

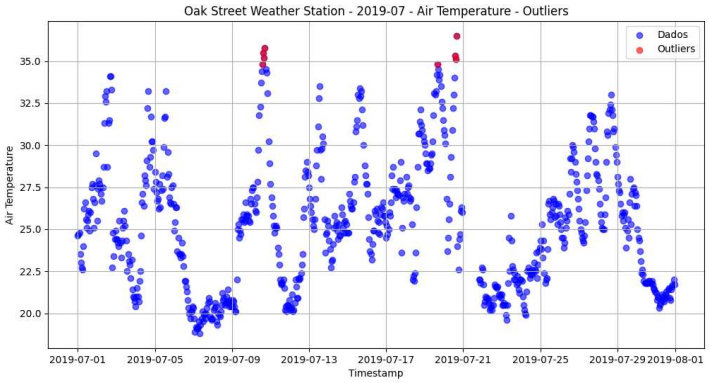
4.4.1.3 Vantagens e Desvantagens

Essa técnica possui as seguintes vantagens: simplicidade e eficiência, sendo adequada para grandes conjuntos de dados. É eficaz em identificar agrupamentos naturais, facilitando a detecção de outliers. E tem como desvantagens a necessidade de especificar o número de clusters previamente, a sensibilidade à inicialização dos centróides e a vulnerabilidade a outliers e ruídos nos dados.

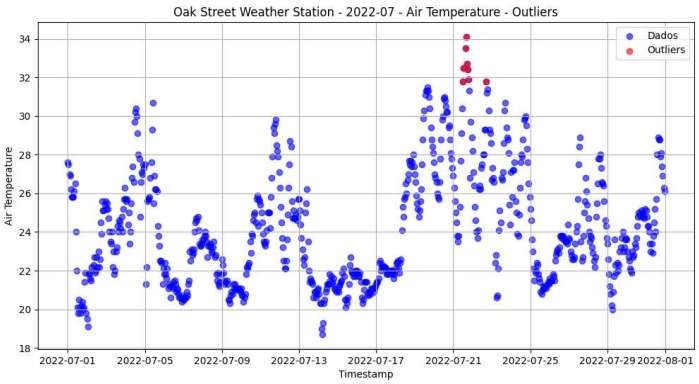
4.4.1.4 Resultados da Aplicação do K-means

A aplicação do algoritmo K-means para análise de clusters foi realizada no dataset utilizando o código apresentado no Apêndice B. Os parâmetros considerados foram explicitados anteriormente. Na análise do K-means, pela primeira vez, é possível observar as distinções entre outliers, que serão melhor demonstradas graficamente.

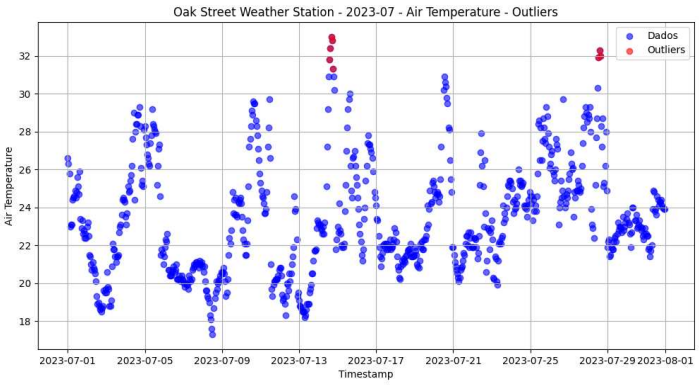
O número de clusters escolhido foi de um cluster. Essa escolha foi feita com o objetivo de identificar outliers de forma mais clara e direta. Ao utilizar apenas um cluster, todos os pontos de dados que não se ajustam bem ao centroide do único cluster formado são considerados outliers. Esse método é especialmente útil quando o objetivo é destacar anomalias sem a influência de múltiplos centroides, o que pode ocorrer em análises com mais clusters. Além disso, essa abordagem simplifica a visualização e a interpretação inicial dos dados anômalos, permitindo um foco mais detalhado na identificação e compreensão dos outliers (CHAWLA; GIONIS, 2013; ROUSSEEUW; HUBERT, 2011).



2019



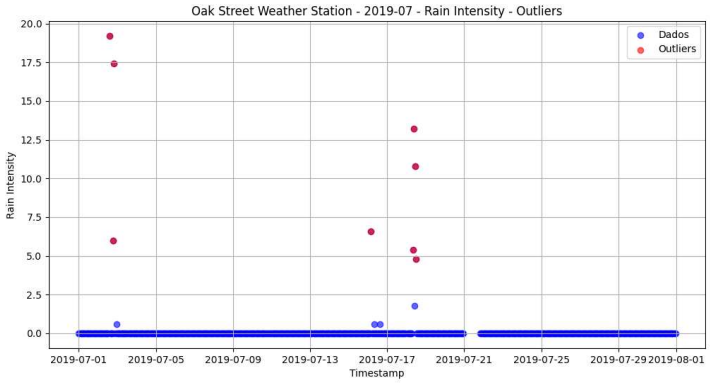
2022



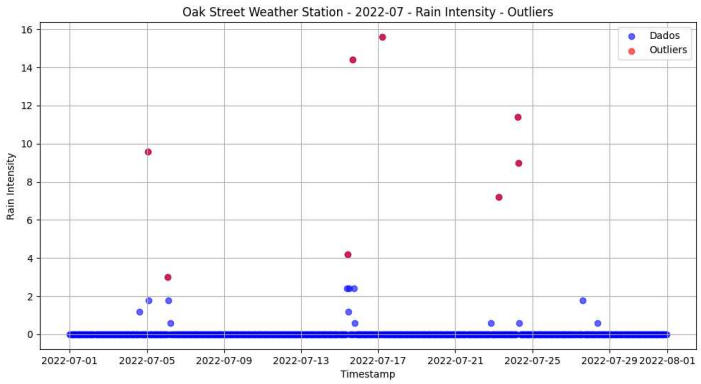
2023

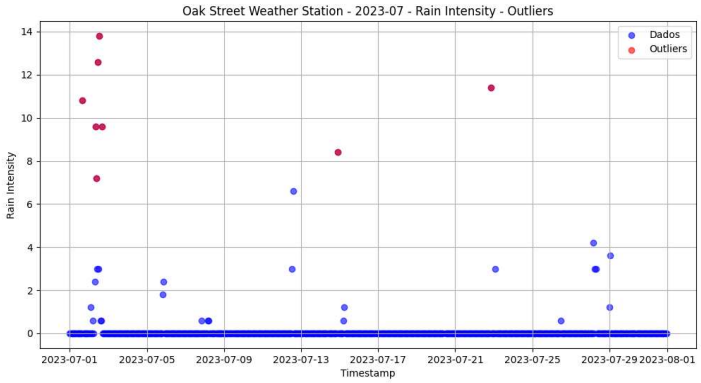
Figura 9 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.



2019

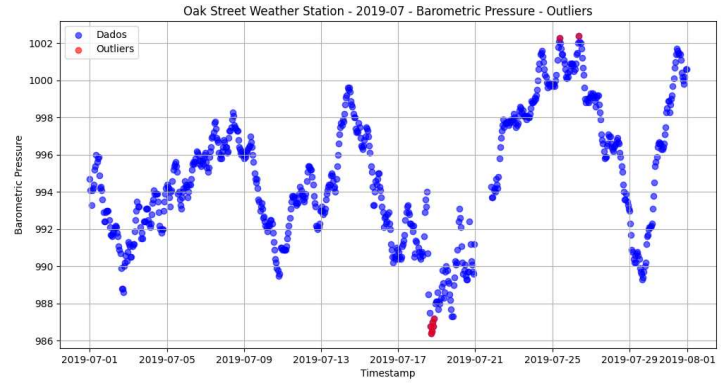




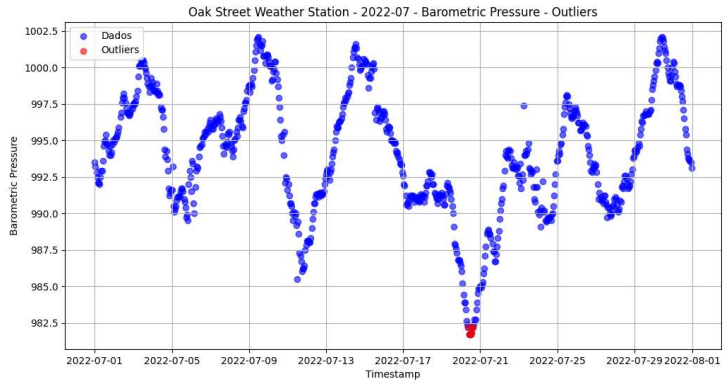
2023

Figura 10 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

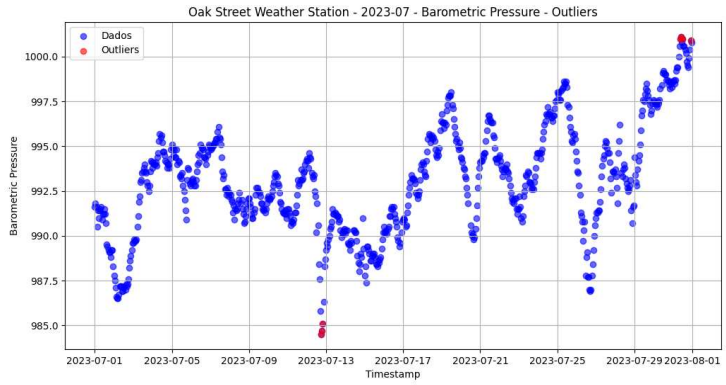
Fonte: Arquivo Pessoal.



2019



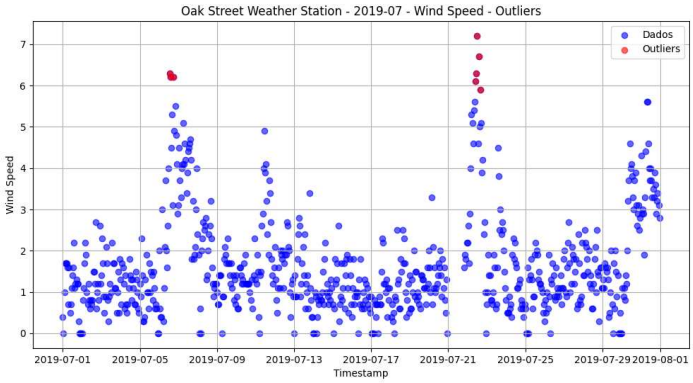
2022



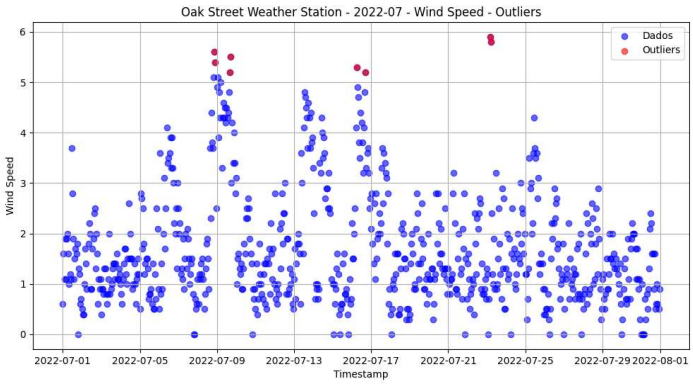
2023

Figura 11 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

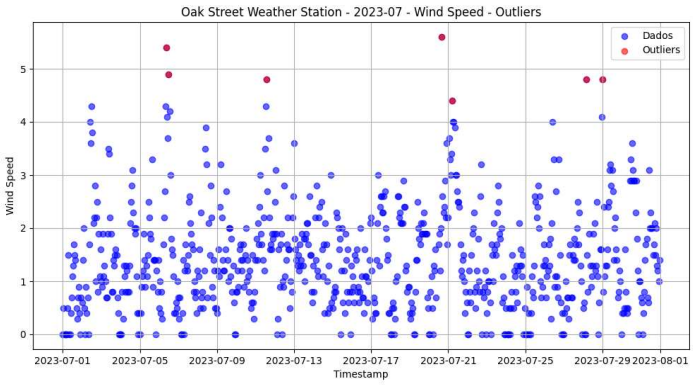
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 12 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

4.4.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é um algoritmo de clustering baseado em densidade que identifica clusters e detecta outliers em conjuntos de dados. Desenvolvido por (ESTER *et al.*, 1996), o DBSCAN é capaz de descobrir clusters de formas arbitrárias e encontrar pontos anômalos que não se ajustam a nenhum cluster. O algoritmo requer dois parâmetros principais: a distância de vizinhança ϵ (eps) e o número mínimo de pontos (*minPts*) necessários para formar um cluster.

4.4.2.1 Funcionamento do DBSCAN

O algoritmo DBSCAN funciona através dos seguintes passos:

1. **Definição dos Pontos de Núcleo**: Um ponto é considerado um ponto de núcleo se tiver pelo menos *minPts* pontos dentro de uma distância ϵ.
2. **Formação de Clusters**: Todos os pontos que estão dentro da vizinhança ϵ de um ponto de núcleo são atribuídos ao mesmo cluster. Este processo é repetido recursivamente para cada ponto de núcleo adicional no cluster.
3. **Identificação de Pontos de Borda**: Pontos que não são pontos de núcleo, mas que estão dentro da vizinhança ϵ de um ponto de núcleo, são considerados pontos de borda.
4. **Identificação de Outliers**: Pontos que não são nem pontos de núcleo nem pontos de borda são classificados como outliers (pontos anômalos).

4.4.2.2 Parâmetros do DBSCAN

O algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é amplamente utilizado para a detecção de anomalias devido à sua capacidade de identificar clusters de forma eficiente e robusta, mesmo em dados ruidosos. Dois parâmetros principais devem ser ajustados para que o DBSCAN funcione corretamente: *eps* (epsilon) e *min_samples*.

O parâmetro *eps* define a distância máxima entre dois pontos para que um seja considerado vizinho do outro. Este parâmetro é crucial para determinar a densidade de pontos necessária para formar um cluster. A escolha de *eps* é geralmente baseada na distribuição dos dados e na análise exploratória.

- **Ensaios e Erros**: Frequentemente, é necessário realizar múltiplos experimentos para ajustar *eps* de forma que balanceie entre a formação de clusters densos e a identificação de pontos de ruído.

Para este caso específico, *eps* foi ajustado para 0.3 baseado na análise preliminar dos dados e através de tentativas, garantindo que pequenos grupos de pontos próximos sejam considerados como parte do mesmo cluster, enquanto pontos mais distantes sejam considerados outliers.

O parâmetro *min_samples* define o número mínimo de pontos necessários para formar um cluster. Este parâmetro determina a robustez do cluster formado e a sensibilidade do algoritmo para detectar outliers. O valor de *min_samples* foi definido como 5 para garantir que os clusters formados sejam densos e robustos, refletindo a natureza dos dados dos sensores ambientais. Este valor é suficientemente grande para evitar a formação de clusters pequenos e sensíveis ao ruído, mas também não é tão grande que a maioria dos pontos sejam classificados como ruído.

- **Natureza dos Dados**: Para dados de sensores ambientais, como os analisados, os fenômenos observados geralmente apresentam uma certa consistência e densidade mínima. Portanto, escolher um valor adequado para *min_samples* é crucial para capturar corretamente essa densidade.
- **Equilíbrio entre Sensibilidade e Robustez**: Um valor muito baixo de *min_samples* pode resultar em muitos clusters pequenos e sensíveis ao ruído, enquanto um valor muito alto pode fazer com que DBSCAN classifique muitos pontos como ruído. A escolha de 5 para *min_samples* foi feita para equilibrar a sensibilidade e robustez, garantindo que clusters formados sejam significativos e representativos dos fenômenos observados.

O código utilizado para aplicar o DBSCAN é apresentado abaixo:

```
def identificar_outliers_dbscan(data, columns, eps=0.5, min_samples=5):
    for column in columns:
        # Preparar os dados para o DBSCAN
        column_data = data[[column]].dropna()
        if len(column_data) < min_samples:
            continue
        # Ajustar DBSCAN
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        data.loc[column_data.index, f'{column}_cluster'] =
        dbscan.fit_predict(column_data)

        # Identificar outliers (DBSCAN retorna -1 para outliers)
        data.loc[column_data.index, f'{column}_outlier_DBSCAN'] =
        data[f'{column}_cluster'] == -1
    return data
```

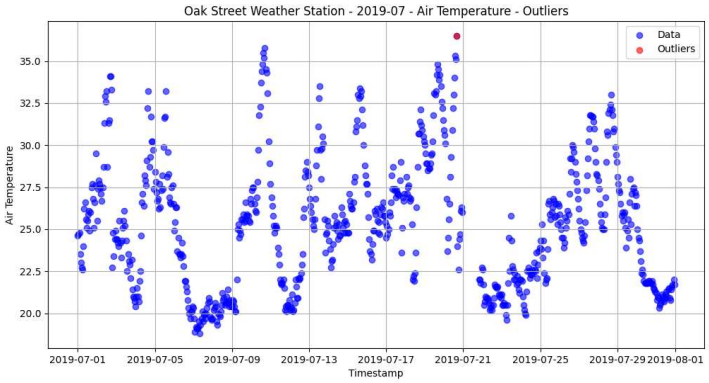
4.4.2.3 Vantagens e Desvantagens

Essa técnica possui as seguintes vantagens: não requer a especificação prévia do número de clusters, identifica clusters de formatos arbitrários e é robusta a ruídos, diferenciando-os de outliers. E tem como desvantagens a dificuldade em determinar um valor adequado para o parâmetro eps e a baixa eficácia em conjuntos de dados com grandes diferenças de densidade entre clusters. Além disso, DBSCAN não é totalmente determinístico, podendo produzir resultados diferentes em execuções distintas devido à escolha aleatória do ponto inicial.

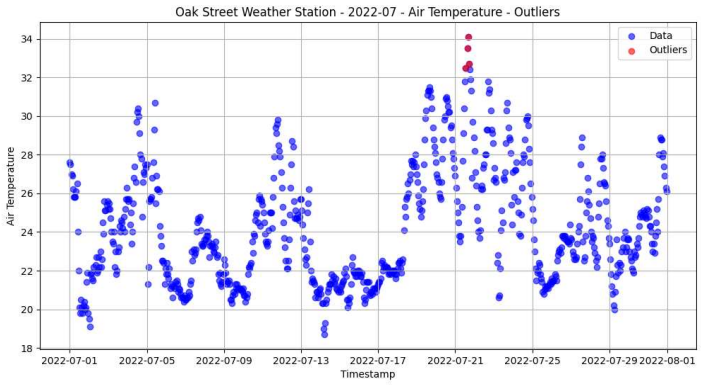
4.4.2.4 Resultados da Aplicação do DBSCAN

A aplicação do DBSCAN para análise de clusters e detecção de outliers foi realizada no dataset utilizando a função apresentada acima, o código completo está descrito no Apêndice C. A análise gráfica inicial, conforme ilustrado nas Figuras 13, 14, 15 e 16, permite observar o comportamento do método na detecção. Neste caso, é importante destacar que menos outliers foram encontrados, o que, pode demonstrar uma maior permissividade por outliers eventos e que possuem uma tendência. Destaca-se a Figura 15, em que a detecção dos dados excepcionais foi maior, assim como nos demais métodos.

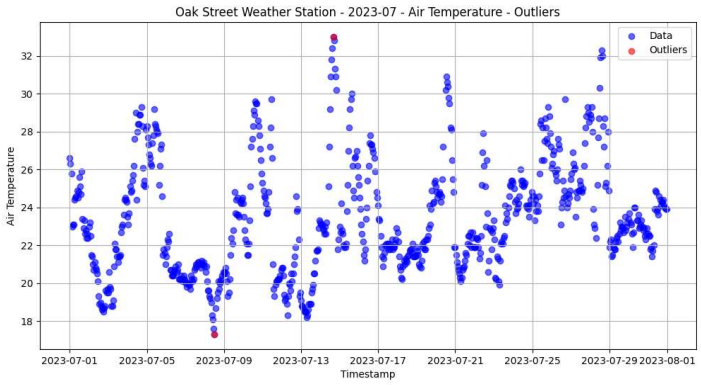
O DBSCAN trata-se de um modelo de clusterização, assim, a Figura 17 representa o gráfico referente ao sensor de intensidade da chuva com a visualização de seus clusters, aqui são definidos com mais clareza em quais clusters os dados fazem parte e em preto a marcação dos outliers. Dessa forma, pode-se com compreender o comportamento da medição e tratar os valores anômalos.



2019



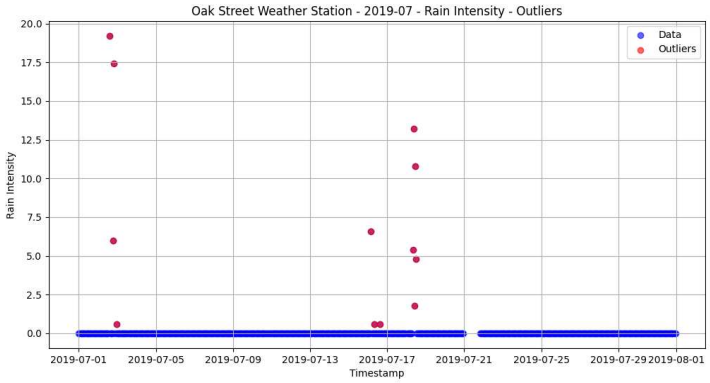
2022



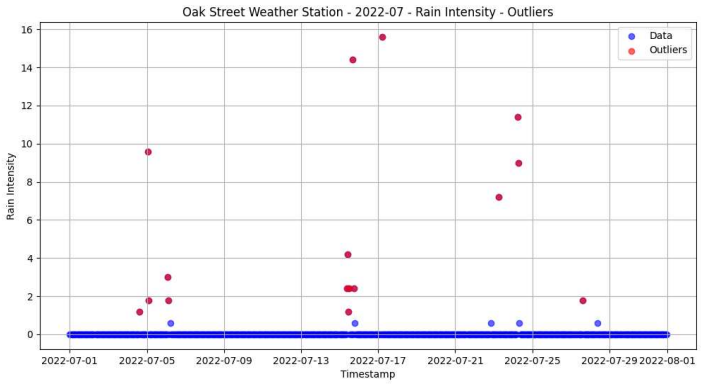
2023

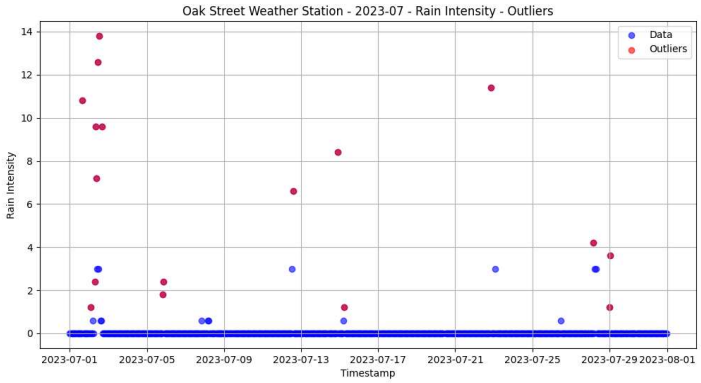
Figura 13 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.



2019

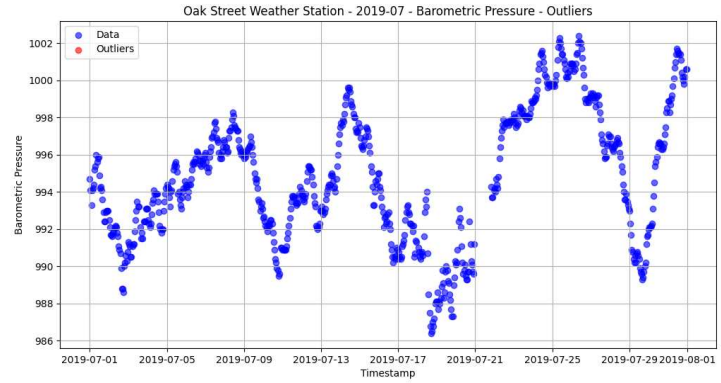




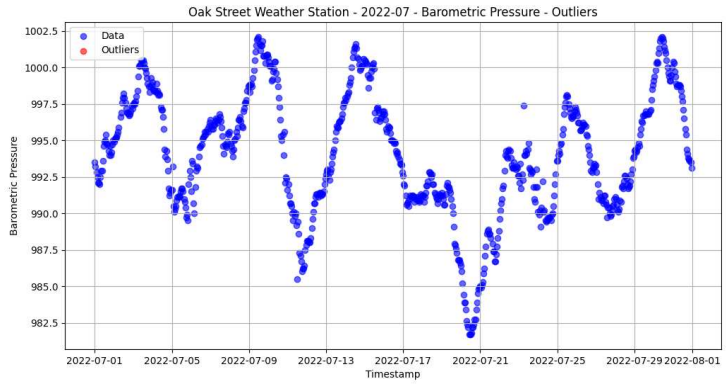
2023

Figura 14 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

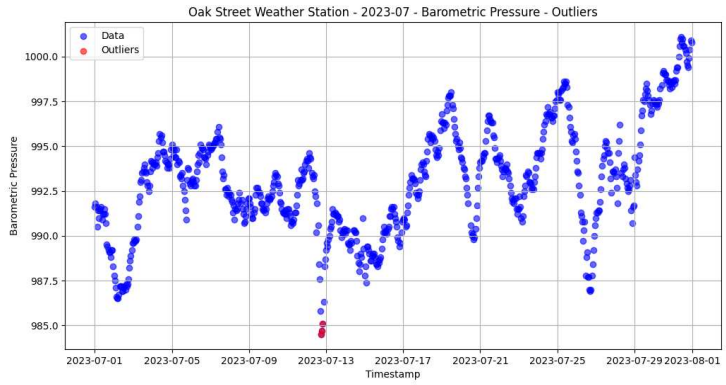
Fonte: Arquivo Pessoal.



2019



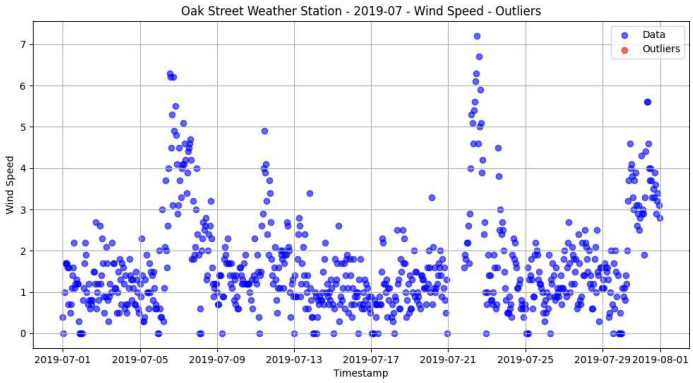
2022



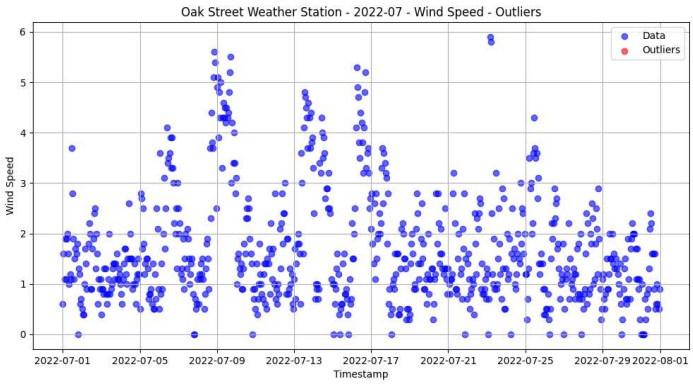
2023

Figura 15 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

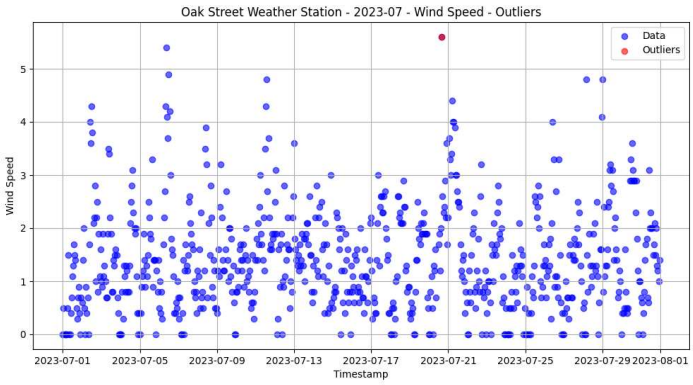
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 16 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

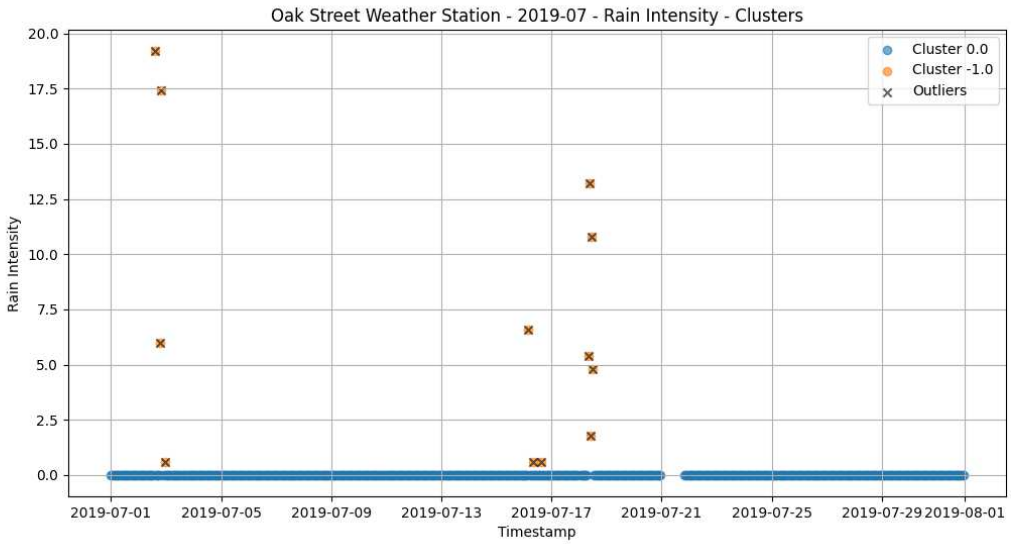


Figura 17 – Representação em clusters dos dados referentes ao Sensor de Chuva para o ano de 2019.

Fonte: Arquivo Pessoal.

4.5 ABORDAGENS DE CLASSIFICAÇÃO

4.5.1 SVM

Support Vector Machine (SVM) é uma técnica de aprendizado supervisionado amplamente utilizada tanto para classificação quanto para regressão. Na detecção de outliers, uma variante chamada One-Class SVM é comumente empregada. Este método é particularmente eficaz em cenários onde os dados de treinamento contêm apenas exemplos da classe normal, e o objetivo é identificar anomalias ou outliers que não se conformam com o padrão aprendido.

4.5.1.1 Funcionamento do SVM na Detecção de Outliers

O One-Class SVM funciona mapeando os dados de entrada para um espaço de alta dimensionalidade usando uma função de kernel. Em seguida, ele tenta encontrar um hiperplano que separe a maioria dos dados do espaço de origem, maximizando a margem entre os pontos de dados e o hiperplano. Os pontos que estão fora dessa margem são considerados outliers (SCHÖLKOPF *et al.*, 2001).

A função de decisão para a detecção de outliers é dada por:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) - \rho \tag{6}$$

onde α_i são os coeficientes dos vetores de suporte, $K(x_i, x)$ é a função de kernel, e ρ é o bias do modelo. Um ponto x é considerado um outlier se $f(x) < 0$.

4.5.1.2 Aplicação do SVM na Detecção de Outliers em Sensores

A aplicação do One-Class SVM para detectar outliers em dados de sensores envolve os seguintes passos:

1. **Treinamento do Modelo**: Utilizar um conjunto de dados contendo apenas exemplos normais para treinar o modelo One-Class SVM.
2. **Predição de Outliers**: Aplicar o modelo treinado aos dados de teste para identificar outliers.
3. **Interpretação dos Resultados**: Analisar os resultados para verificar quais pontos foram identificados como outliers e entender o motivo dessas anomalias.

O código a seguir demonstra a aplicação do One-Class SVM utilizando a biblioteca *scikit-learn*:

```
# Função para identificar outliers usando One-Class
# SVM com GridSearchCV
def identificar_outliers_svm(data, columns):
    for column in columns:
        # Preparar os dados para o One-Class SVM
        column_data = data[[column]].dropna()
        if len(column_data) < 2:
            continue
        # Encontrar os melhores parâmetros
        melhores_parametros = encontrar_melhores_parametros(column_data)
        nu = melhores_parametros['nu']
        gamma = melhores_parametros['gamma']
        # Imprimir os parâmetros escolhidos
        print(f'Parâmetros escolhidos para {column} -
        nu: {nu}, gamma: {gamma}')
        # Ajustar One-Class SVM com os melhores parâmetros
        oc_svm = OneClassSVM(nu=nu, kernel='rbf', gamma=gamma)
        data.loc[column_data.index, f'{column}_outlier_SVM'] =
        oc_svm.fit_predict(column_data)
        # Identificar outliers (One-Class SVM retorna -1
        # para outliers e 1 para inliers)
        data.loc[column_data.index, f'{column}_outlier_SVM'] =
        data[f'{column}_outlier_SVM'] == -1

    return data
```

4.5.1.3 Escolha dos Parâmetros do One-Class SVM

Para aplicar o One-Class SVM de maneira eficaz, é crucial escolher os parâmetros adequados. Os principais parâmetros a serem ajustados são o *kernel* e o *nu* e *gamma*.

O parâmetro *kernel* define a função de kernel usada para mapear os dados de entrada para um espaço de alta dimensionalidade. O kernel radial basis function (RBF) é frequentemente usado devido à sua capacidade de lidar com relações não lineares entre os dados.

O parâmetro *nu* define uma estimativa superior da fração de outliers no conjunto de dados e uma estimativa inferior da fração de vetores de suporte. Os parâmetros *nu* e *gamma* são fundamentais para o desempenho do One-Class SVM na detecção de anomalias. O parâmetro nu representa uma cota superior na fração de outliers esperada e uma cota inferior na fração de pontos de suporte. especificidade do modelo na identificação de outliers.

O kernel RBF foi escolhido por sua eficácia em capturar relações complexas nos dados dos sensores ambientais, permitindo uma melhor separação entre pontos normais e anômalos (SCHÖLKOPF *et al.*, 2001).

Os parâmetros foram escolhidos de maneira independente para cada medição, utilizando como métrica a acurácia através da biblioteca *GridSearchCV* e variando os parâmetros entre uma escala para verificar como o comportamento de cada detecção se comportaria. Como pode ser observado no código:

```
# Função para encontrar os melhores parâmetros de One-Class
# SVM usando GridSearchCV
def encontrar_melhores_parametros(column_data):
    param_grid = {
        'nu': [0.01, 0.05, 0.1, 0.2],
        'gamma': ['scale', 'auto', 0.1, 0.5, 1]
    }
    grid_search = GridSearchCV(OneClassSVM(kernel='rbf'),
    param_grid, cv=5, scoring='accuracy')
    grid_search.fit(column_data, np.ones(len(column_data)))
    return grid_search.best_params_
```

Através da lista de parâmetros os dados são treinados e a acurácia medida, dessa forma pode-se selecionar para cada caso a melhor combinação de *nu* e *gamma*.

4.5.1.4 Vantagens e Desvantagens

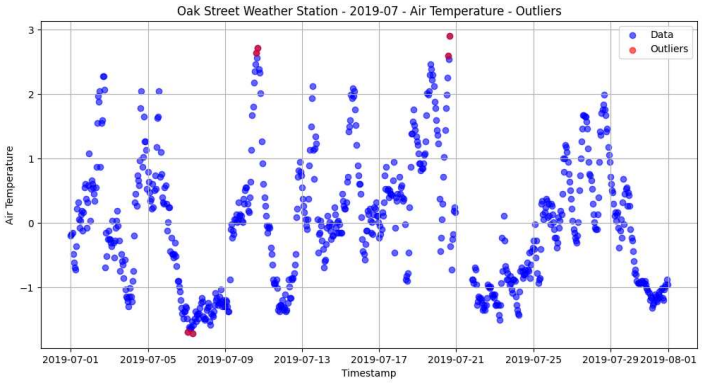
Essa técnica possui as seguintes vantagens: é eficaz em espaços de alta dimensionalidade, tem um bom desempenho quando há uma clara margem de separação

entre classes e pode ser utilizada tanto para detecção de outliers quanto para classificação e regressão. E tem como desvantagens a sensibilidade à escolha do kernel e dos parâmetros, a exigência de tempo para ajuste e otimização, além de ser menos eficiente em termos de tempo e memória em grandes conjuntos de dados, podendo ser influenciada por outliers no conjunto de treinamento.

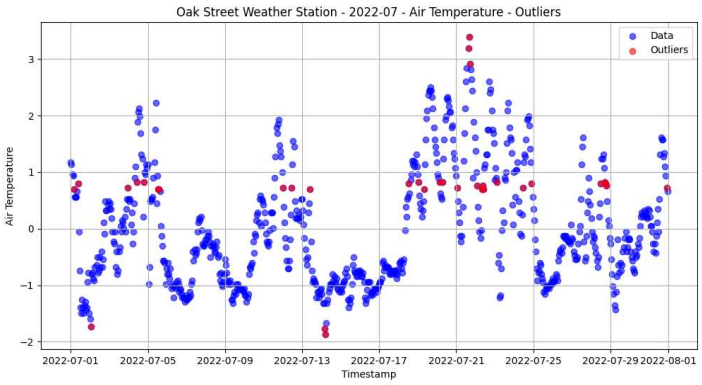
4.5.1.5 Resultados da Aplicação do SVM

A aplicação do One-Class SVM foi realizada no dataset, e o código detalhado pode ser visto no Apêndice D. A eficácia deste método depende diretamente do ajuste preciso de seus parâmetros. Após a aplicação variável dos parâmetros e treinamento do método, alguns cenários apresentaram detecções de outliers em regiões médias.

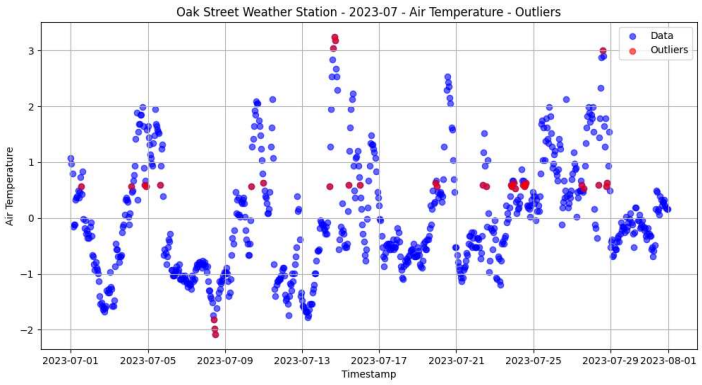
Esta observação destaca uma importante avaliação para a técnica: para leituras dos mesmos sensores e utilizando os mesmos parâmetros, a detecção foi diferente em cenários semelhantes. Em destaque, as Figuras 22 e 23 mostram que no ano de 2023, o cenário de detecção se inverteu, o que é digno de nota e merece uma análise mais aprofundada. Esta variação pode ser atribuída a mudanças nas condições ambientais ou no comportamento dos sensores, indicando a necessidade de um ajuste contínuo e específico dos parâmetros do modelo para garantir a precisão na detecção de outliers.



2019



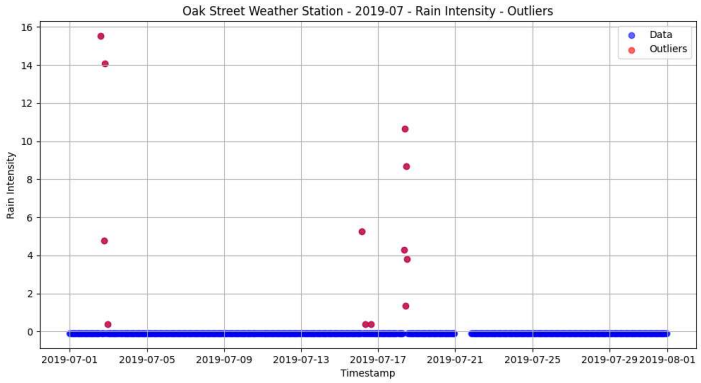
2022



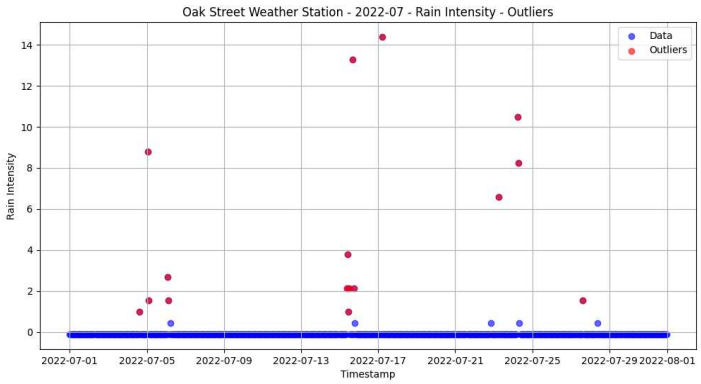
2023

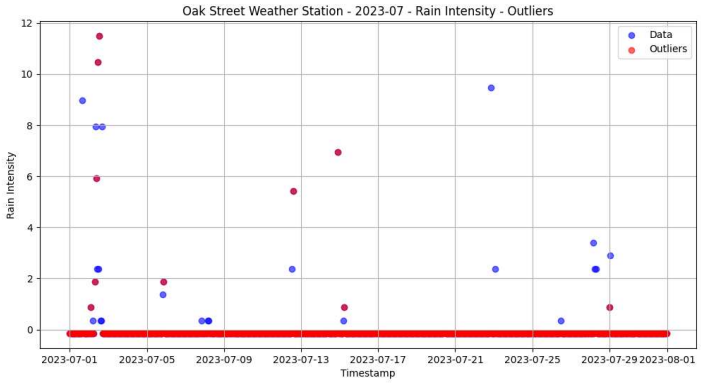
Figura 18 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.



2019

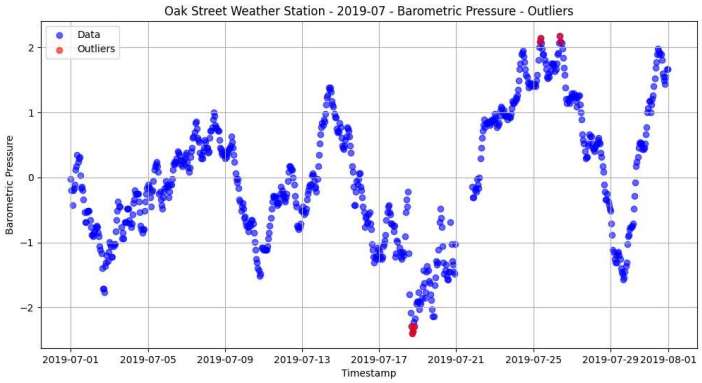




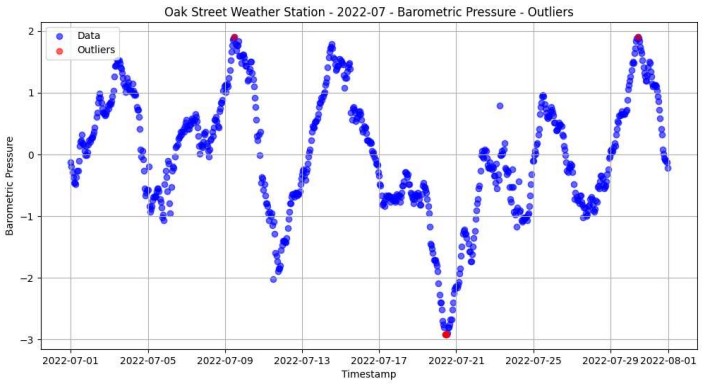
2023

Figura 19 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

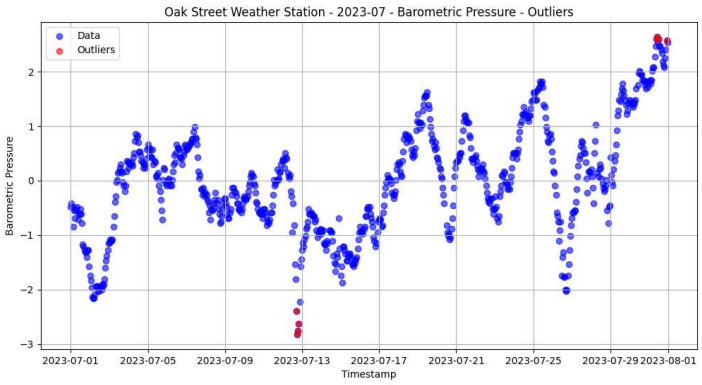
Fonte: Arquivo Pessoal.



2019



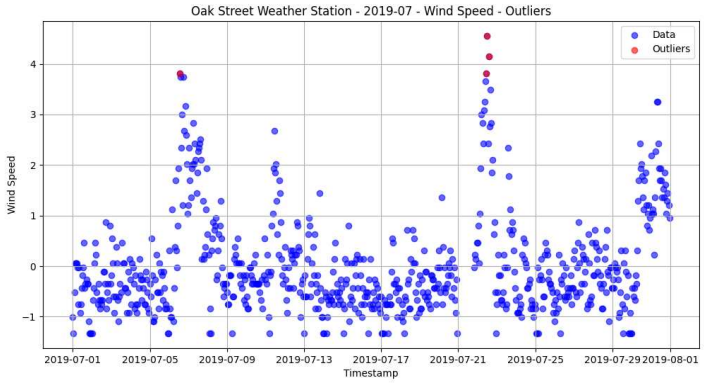
2022



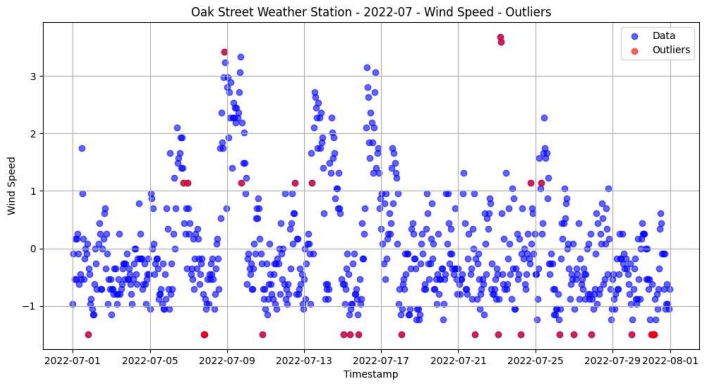
2023

Figura 20 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

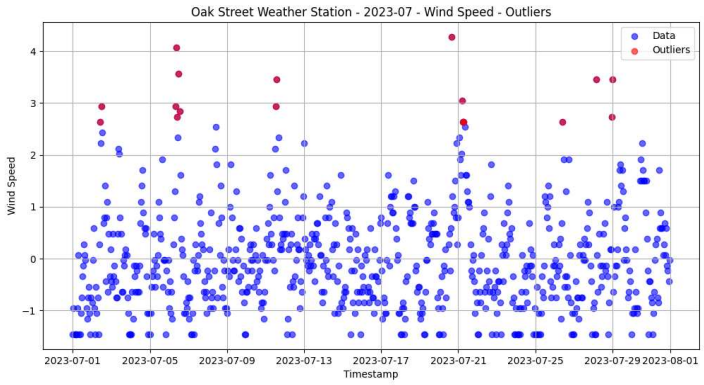
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 21 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

4.6 REDES NEURAIS

4.6.1 Aplicação de Redes Neurais para Detecção de Outliers

Redes neurais, especificamente Autoencoders, têm se mostrado eficazes na detecção de outliers em conjuntos de dados complexos. Um Autoencoder é uma rede neural utilizada para aprender uma representação (codificação) eficiente dos dados,

geralmente para redução de dimensionalidade ou para detecção de anomalias. O Autoencoder é treinado para reconstruir seus dados de entrada, e a diferença entre a entrada e a saída reconstruída pode ser usada para identificar outliers.

4.6.1.1 Construção do Autoencoder

A construção do Autoencoder para detecção de outliers envolve as seguintes etapas:

1. **Camada de Entrada**: Define a dimensão de entrada dos dados.
2. **Codificação**: Três camadas densas com 32, 16 e 8 neurônios, respectivamente, utilizando a função de ativação *ReLU*.
3. **Decodificação**: Três camadas densas com 16, 32 neurônios e uma camada de saída com a mesma dimensão da entrada, utilizando a função de ativação *sigmoid*.

O código a seguir demonstra a construção do Autoencoder:

```
# Função para construir o Autoencoder
def construir_autoencoder(dimension_input):
    input_layer = Input(shape=(dimension_input,))
    encoded = Dense(32, activation='relu')(input_layer)
    encoded = Dense(16, activation='relu')(encoded)
    encoded = Dense(8, activation='relu')(encoded)

    decoded = Dense(16, activation='relu')(encoded)
    decoded = Dense(32, activation='relu')(decoded)
    decoded = Dense(dimension_input, activation='sigmoid')(decoded)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder
```

4.6.1.2 Parâmetros do Autoencoder

- **Número de Neurônios**: As camadas codificadoras e decodificadoras possuem 32, 16 e 8 neurônios. Este esquema reduz gradualmente a dimensionalidade dos dados, permitindo ao modelo capturar as características principais.
- **Épocas**: O modelo é treinado por 50 épocas, um número suficiente para que a rede aprenda a reconstruir os dados de entrada sem sobreajustar.

- **Tamanho do Lote**: Um tamanho de lote de 32 é usado para garantir que o modelo treine eficientemente, mantendo um equilíbrio entre a velocidade de treinamento e a estabilidade da atualização dos pesos.

4.6.1.3 Funções de Ativação

- **ReLU (Rectified Linear Unit)**: Utilizada nas camadas ocultas para introduzir não-linearidade, permitindo que a rede aprenda relações complexas nos dados.
- **Sigmoid**: Utilizada na camada de saída para garantir que os valores reconstruídos estejam no intervalo [0, 1], facilitando a comparação com os dados normalizados de entrada.

4.6.1.4 Aplicação do Autoencoder na Detecção de Outliers

A aplicação do Autoencoder para detectar outliers em dados de sensores envolve as seguintes etapas:

1. **Preparação dos Dados**: Normalização dos dados e remoção de valores nulos.
2. **Construção do Autoencoder**: Utilizando a função *construir_autoencoder*.
3. **Treinamento do Autoencoder**: Treinamento do modelo com os dados de entrada.
4. **Reconstrução e Cálculo do Erro**: Cálculo do erro de reconstrução para identificar outliers.
5. **Definição de Limiar**: Utilização do percentil 95 do erro de reconstrução como limiar para identificar outliers.

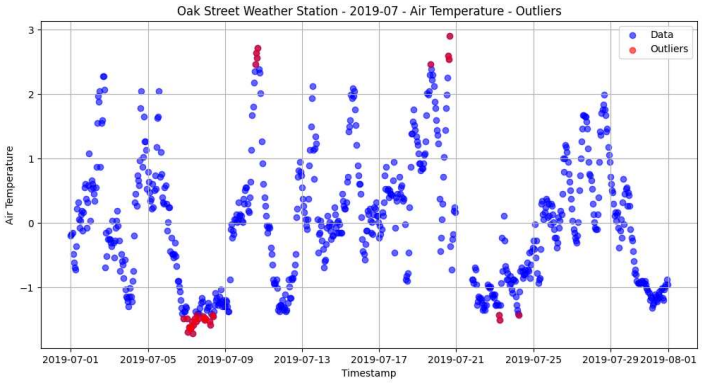
O código que demonstra a aplicação do Autoencoder para detecção de outliers pode ser visto no Apêndice E.

4.6.1.5 Resultados da Aplicação do Autoencoder

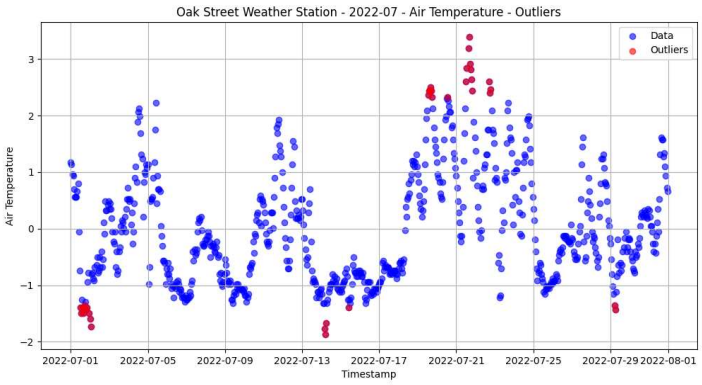
A aplicação do Autoencoder no dataset demonstrou a eficácia do método na detecção de outliers. Na visualização gráfica, os pontos normais são representados em azul, enquanto os outliers identificados estão destacados em vermelho. Os resultados mostram que o Autoencoder é capaz de identificar não apenas valores extremos, mas também pontos que não ocorrem frequentemente, fornecendo uma abordagem robusta para a detecção de anomalias. Como pode ser visto nas figuras 22, 23, 24 e 25

Além disso, o uso de Autoencoders encontrou mais outliers do que outros métodos, tornando-o especialmente eficiente para uso em outros modelos ,como por

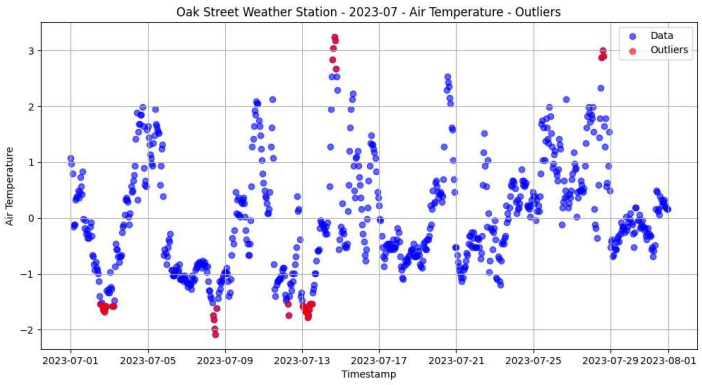
exemplo, predição baseados em outliers. Essa capacidade aprimorada de detecção pode melhorar significativamente a precisão dos modelos preditivos, já que os outliers identificados podem ser usados como indicadores importantes em diversas aplicações. Autoencoders, especialmente aqueles baseados em RNRs (Redes Neurais Recorrentes), são eficazes em lidar com dados temporais e sequenciais, capturando dependências complexas que outros métodos podem não detectar. Essa combinação de robustez e eficácia faz dos Autoencoders uma escolha valiosa para tarefas de detecção de anomalias e predição.



2019



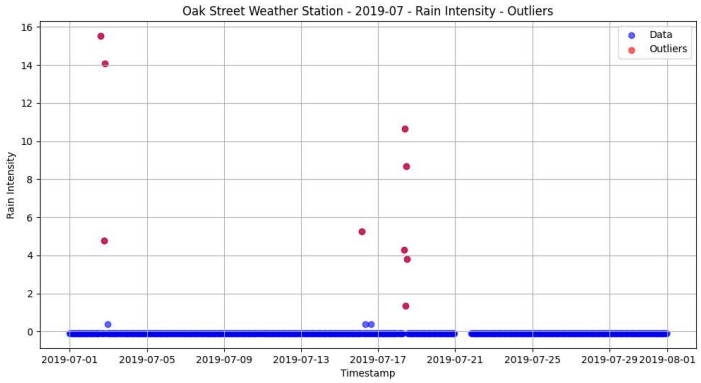
2022



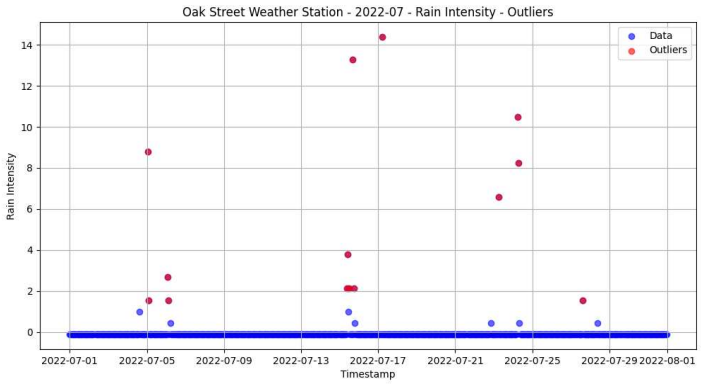
2023

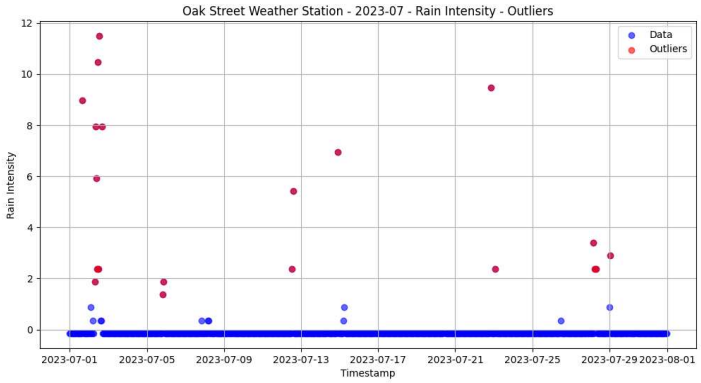
Figura 22 – Gráfico com os dados coletados pelo Sensor de Temperatura no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.



2019

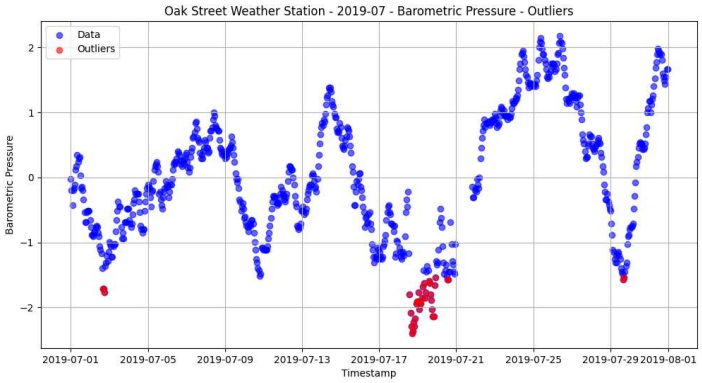




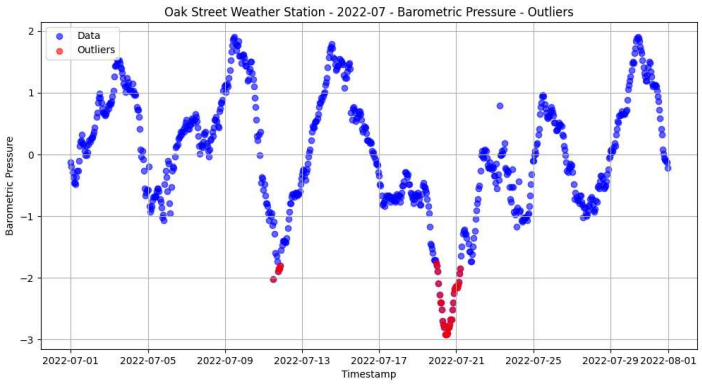
2023

Figura 23 – Gráfico com os dados coletados pelo Sensor de Chuva no mês de julho, e os Outliers destacados.

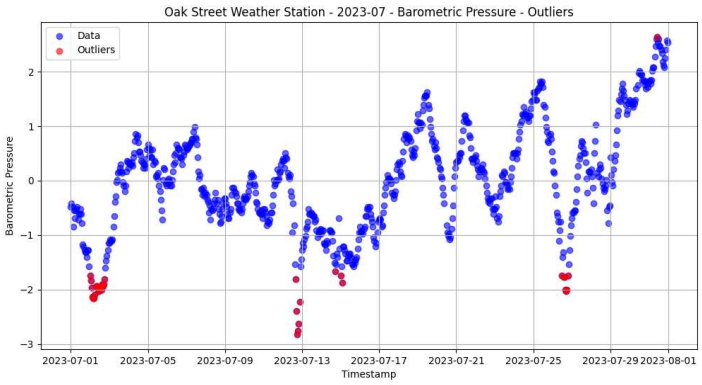
Fonte: Arquivo Pessoal.



2019



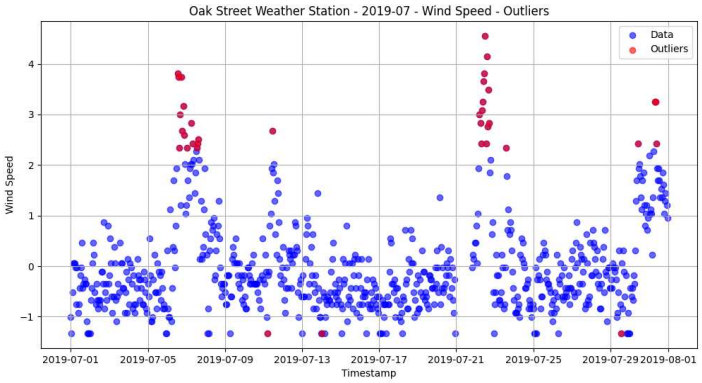
2022



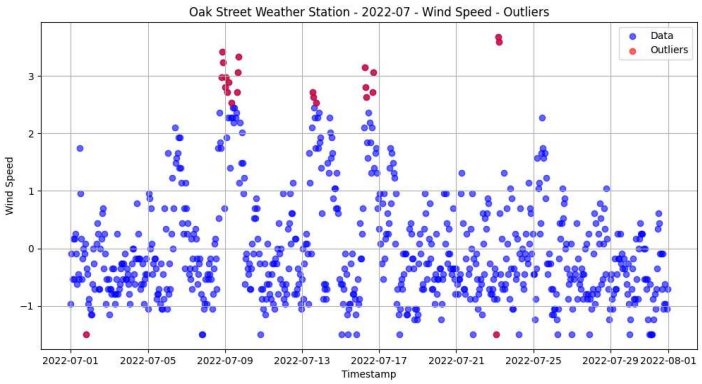
2023

Figura 24 – Gráfico com os dados coletados pelo Sensor de pressão atmosférica no mês de julho, e os Outliers destacados.

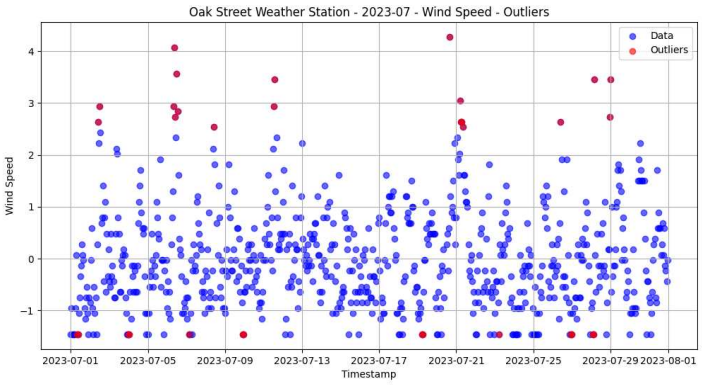
Fonte: Arquivo Pessoal.



2019



2022



2023

Figura 25 – Gráfico com os dados coletados pelo Sensor de velocidade do vento no mês de julho, e os Outliers destacados.

Fonte: Arquivo Pessoal.

5 ANÁLISES FINAIS

5.1 COMPARAÇÃO DE MÉTODOS E ANÁLISES ENTRE ESTAÇÕES METEOROLÓGICAS

Neste capítulo, são comparados os métodos de detecção de outliers utilizados e suas respectivas detecções. Além disso, são realizadas análises comparativas entre as diferentes estações meteorológicas, embora tenha sido escolhida apenas uma estação específica para exemplificação detalhada. Essa abordagem permite verificar se os outliers detectados são anomalias locais ou se representam eventos detectados e compartilhados entre múltiplas estações.

5.1.1 Comparação dos Métodos de Detecção de Outliers

Os métodos de detecção de outliers utilizados incluem K-means, DBSCAN, One-Class SVM e Autoencoders. Cada método possui suas particularidades e adequações conforme a natureza dos dados analisados. A seguir, são apresentadas as principais características e resultados obtidos para cada um dos métodos.

5.2 SÍNTESE DOS MÉTODOS E RESULTADOS

Z-Score e K-Means: O método Z-Score foi simples de implementar e proporcionou resultados satisfatórios, especialmente em termos de tempo de processamento. A aplicação do K-Means também mostrou-se eficiente, sendo um método robusto que facilitou a identificação de outliers com base na distância média entre clusters. Ambos os métodos demonstraram uma boa capacidade de detectar anomalias nos dados, sendo particularmente úteis para análises rápidas e eficazes.

K-Nearest Neighbors (KNN): O KNN, apesar de ser mais computacionalmente intensivo, proporcionou uma análise detalhada das proximidades entre os pontos, permitindo identificar outliers com base em vizinhos mais próximos. Este método foi eficaz para detectar anomalias em dados densamente povoados.

DBSCAN: A utilização do DBSCAN permitiu a detecção de outliers de forma eficiente em dados com densidades variadas. A capacidade do DBSCAN de lidar com clusters de forma flexível e identificar ruídos mostrou-se vantajosa, especialmente em contextos onde a densidade de dados varia significativamente.

Support Vector Machine (SVM): O SVM, especificamente o One-Class SVM, foi aplicado para identificar outliers com base em uma fronteira de decisão aprendida a partir dos dados. Este método apresentou resultados variáveis, com a detecção de anomalias em diferentes regiões do espaço de características, indicando sua sensibilidade aos parâmetros ajustados.

Autoencoder (RNR): A implementação do Autoencoder permitiu a detecção de outliers com base em erros de reconstrução. Este método, embora mais complexo, mostrou-se potente na identificação de anomalias subtis, especialmente em dados de alta dimensionalidade.

5.2.1 Análise Entre Estações Meteorológicas

A análise entre diferentes estações meteorológicas visa identificar se os outliers detectados são eventos locais ou globais. Embora a análise detalhada tenha sido exemplificada com dados de uma única estação, a comparação entre estações permitirá uma compreensão mais abrangente dos eventos anômalos detectados.

5.2.1.1 Análise Local vs Global

Para determinar se um outlier é uma anomalia local ou um evento global, é necessário comparar as leituras anômalas entre múltiplas estações. Se um outlier for detectado em uma única estação, é provável que se trate de uma anomalia local, possivelmente devido a falhas ou interferências específicas daquela estação. Por outro lado, se múltiplas estações detectarem um evento anômalo simultaneamente, é mais provável que o evento seja um fenômeno ambiental real compartilhado entre as estações.

5.2.1.2 Exemplificação com Figuras

As figuras a seguir ilustram os resultados das detecções de outliers em diferentes estações meteorológicas. A Figura 26 apresenta a detecção de outliers em 2019 para o sensor de chuva na estação da Rua 63, enquanto a Figura 27 destaca os outliers detectados na estação da Rua Oak no mesmo período.

Observa-se que, em alguns casos, a detecção de outliers variou significativamente entre as medições das estações, sugerindo mudanças nos padrões de leitura dos sensores ou eventos ambientais específicos. Por exemplo, entre os dias 01 e 05, o sensor da Rua Oak (Figura 27) detectou pontos extremos de precipitação em 20 mm, enquanto o sensor da Rua 63 (Figura 26) não registrou tais medições. Essa discrepância pode ser classificada como um outlier local e serve como análise para situações de microclima, onde pode ocorrer uma chuva intensa em uma região da cidade, mas não em outra.

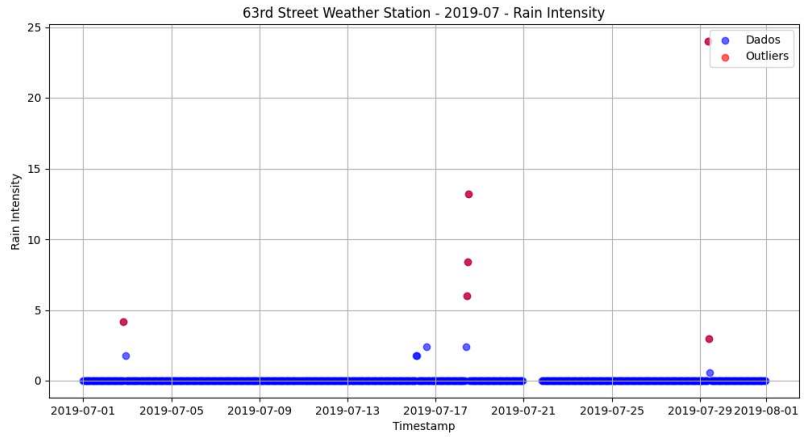


Figura 26 – Detecção de outliers na estação meteorológica da Rua 63 em 2019.

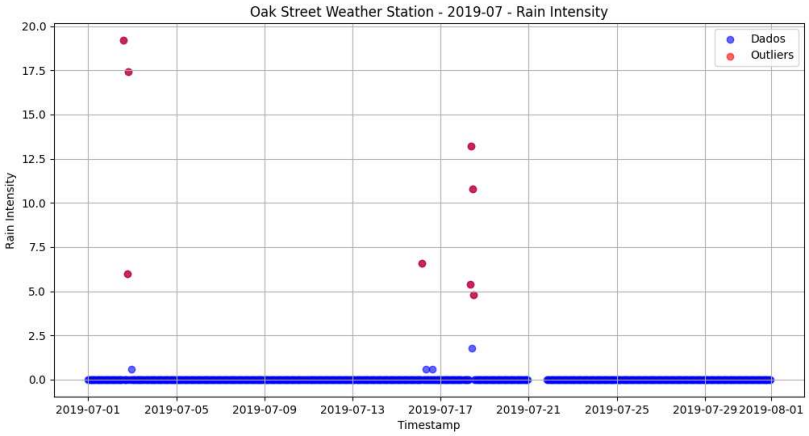


Figura 27 – Detecção de outliers na estação meteorológica da Rua Oak em 2019.

Essas análises são fundamentais para validar a robustez dos métodos de detecção de outliers e compreender a natureza das anomalias detectadas. A variabilidade nas detecções entre as estações sugere a necessidade de uma abordagem cuidadosa ao interpretar os resultados, levando em consideração fatores locais e condições específicas de cada estação.

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, aplicamos diversos métodos de detecção de outliers em dados coletados de sensores Wi-Fi em estações meteorológicas. As técnicas abordadas incluíram Z-Score, K-Nearest Neighbors (KNN), K-Means, DBSCAN, Support Vector Machine (SVM) e Autoencoder, cada uma com suas respectivas justificativas e parâmetros ajustados. A seguir, apresentamos uma síntese dos métodos utilizados e seus resultados, ressaltando a importância da análise de outliers locais e globais.

6.1 SUCESSO DO TRABALHO E APLICABILIDADE

O trabalho demonstrou sucesso na aplicação das técnicas de detecção de outliers, atingindo os objetivos propostos de identificar anomalias em dados de sensores climáticos. Os resultados foram satisfatórios, mostrando que tanto métodos simples quanto mais robustos podem ser utilizados efetivamente para detecção de outliers. Entre os métodos aplicados, o Z-Score, K-Means e Autoencoder se destacaram.

O método Z-Score foi simples de implementar e proporcionou resultados satisfatórios, especialmente em termos de tempo de processamento. Ele se mostrou eficaz para dados que seguem uma distribuição normal, identificando rapidamente os pontos anômalos.

A aplicação do K-Means mostrou-se eficiente, sendo um método robusto que facilitou a identificação de outliers com base na distância média entre clusters. Este método foi particularmente útil para identificar anomalias em dados agrupados de forma clara.

A implementação do Autoencoder permitiu a detecção de outliers com base em erros de reconstrução. Este método, embora mais complexo, mostrou-se potente na identificação de anomalias sutis, especialmente em dados de alta dimensionalidade.

Um dos pontos altos deste trabalho foi a elaboração de um *Jupyter Notebook* contendo todos os códigos utilizados para as diferentes técnicas de detecção. Este *Jupyter Notebook* se mostrou uma ferramenta poderosa, não só para a execução dos métodos neste trabalho, mas também como uma base que pode ser replicada e adaptada para diversos outros projetos. Pesquisadores e profissionais da área podem utilizar este arquivo como ponto de partida para suas próprias análises, ajustando os parâmetros conforme necessário para seus dados específicos.

6.2 ANÁLISE DE OUTLIERS LOCAIS E GLOBAIS

A análise de outliers locais e globais foi uma parte crucial deste estudo. A distinção entre anomalias que afetam apenas uma estação (outliers locais) e aquelas que são detectadas em múltiplas estações (outliers globais) forneceu insights valiosos.

Identificar outliers locais é essencial para entender fenômenos específicos de microclimas, enquanto outliers globais podem indicar tendências climáticas mais amplas. Além disso, o trabalho comparou mais de um sensor, o que permitiu realizar distinções precisas entre outliers locais e globais. Essa abordagem multi-sensorial garantiu uma análise mais robusta e confiável, destacando variações e anomalias que poderiam passar despercebidas em uma análise unidimensional.

6.3 SUGESTÕES PARA TRABALHOS FUTUROS

Para futuros trabalhos, recomenda-se explorar a combinação de diferentes métodos de detecção de outliers, implementando técnicas de *ensemble learning* para aumentar a robustez e precisão das detecções. Além disso, a análise temporal pode ser expandida para modelos preditivos, utilizando os outliers detectados para identificar tendências climáticas globais. A integração de dados de múltiplas fontes, incluindo dados de satélites e outros sensores ambientais, pode fornecer uma visão mais abrangente dos fenômenos climáticos.

A aplicação de técnicas avançadas de *deep learning*, como redes neurais convolucionais (CNNs), pode ser explorada para a detecção de padrões complexos e anomalias em séries temporais de dados climáticos. Adicionalmente, a criação de um sistema automatizado de monitoramento e alerta baseado em detecção de outliers pode ser desenvolvido para fornecer informações em tempo real sobre eventos climáticos extremos.

A análise de microclimas também é uma área promissora para trabalhos futuros. Microclimas são condições climáticas específicas de pequenas áreas, que podem diferir significativamente do clima geral da região circundante devido a fatores como vegetação, topografia e características do solo (BRITANNICA, 2024). Estudar microclimas pode fornecer informações valiosas para a conservação da biodiversidade e para a adaptação às mudanças climáticas (METLINK, 2024).

Em suma, este trabalho demonstrou que a aplicação de uma variedade de técnicas de detecção de outliers em dados de sensores sem fio é viável e eficaz, contribuindo significativamente para a área de monitoramento e análise climática. A partir dos resultados obtidos, futuras pesquisas podem se beneficiar da base estabelecida para aprimorar ainda mais a detecção e análise de anomalias em dados ambientais. Além disso, provou-se que os outliers devem ser analisados em suas abordagens e não apenas excluídos durante os tratamentos.

REFERÊNCIAS

AKYILDIZ, Ian F.; SU, Weilian; SANKARASUBRAMANIAM, Yogesh; CAYIRCI, Erdal. Wireless sensor networks: a survey. **Computer Networks**, 2002. DOI: 10.1016/s1389-1286(01)00302-4.

BARNETT, Vic; LEWIS, T. **Outliers in Statistical Data**. [*S.l.*: *s.n.*], 1994.

BHUSE, V.; GUPTA, A. Anomaly intrusion detection in wireless sensor networks. **Journal of High Speed Networks**, v. 15, n. 1, p. 33–51, 2006.

BRANCH, J.; HEBER, G. T.; RUIZ, L. B.; CAVALCANTI, D.; RODRIGUES, P. R. Lightweight detection of network anomalies using smart sampling. **Computer Communications**, v. 29, n. 11, p. 1965–1979, 2006.

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. LOF: Identifying Density-Based Local Outliers. **Proceedings of the ACM SIGMOD International Conference on Management of Data**, p. 93–104, 2000.

BRITANNICA, Encyclopaedia. **Microclimate**. 2024. Disponível em: https://www.britannica.com/science/microclimate. Acesso em: 10 mai. 2024.

CHATZIGIANNAKIS, I.; PAPAVASSILIOU, S.; GRAMMATIKOU, M. PCA-based anomaly detection in wireless sensor networks. *In*: PROCEEDINGS of the 14th IEEE Mediterranean Electrotechnical Conference. [*S.l.*: *s.n.*], 2006. P. 1340–1345.

CHAWLA, S.; GIONIS, A. k-means–: A Unified Approach to Clustering and Outlier Detection. **Proceedings of the SIAM International Conference on Data Mining**, p. 189–197, 2013. Disponível em: https://doi.org/10.1137/1.9781611972832.21. Acesso em: 21 mai. 2024.

CHEN, J.; KHER, S.; SOMANI, A. Distributed fault detection of wireless sensor networks. *In*: PROCEEDINGS of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks. [*S.l.*: *s.n.*], 2006. P. 65–72.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967.

DISTRICT, Chicago Park. **Beach Weather Stations - Automated Sensors**. 2015. Disponível em: `https://data.cityofchicago.org/Parks-Recreation/Beach-Weather-Stations-Automated-Sensors/k7hf-8y75`. Acesso em: 30 mai. 2024.

DISTRICT, Chicago Park. **Beach Weather Stations - Automated Sensors - 2016**. 2016. Disponível em: `https://data.cityofchicago.org/Parks-Recreation/Beach-Weather-Stations-Automated-Sensors-2016-Inte/kpqj-2v8e`. Acesso em: 30 mai. 2024.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. **Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining**, p. 226–231, 1996.

FOX, A. J. Outliers in time series. **Journal of the Royal Statistical Society**, v. 34, n. 3, p. 350–363, 1972.

HAWKINS, Douglas M. **Identification of outliers**. [*S.l.*: *s.n.*], 1980.

HAWKINS, Simon; HE, Hongxing; WILLIAMS, Graham; BAXTER, Rohan. Outlier Detection Using Replicator Neural Networks. *In*: SPRINGER-VERLAG BERLIN HEIDELBERG. DAWAK 2002. [*S.l.*: *s.n.*], 2002. P. 170–180.

HODGE, V. J.; AUSTIN, J. A Survey of Outlier Detection Methodologies. **Artificial Intelligence Review**, v. 22, n. 2, p. 85–126, 2004.

KRISHNAMACHARI, B.; IYENGAR, S. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. **IEEE Transactions on Computers**, v. 53, n. 3, p. 241–250, 2004.

MACQUEEN, J. B. Some Methods for Classification and Analysis of Multivariate Observations. **Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability**, v. 1, p. 281–297, 1967. Disponível em: `https://projecteuclid.org/euclid.bsmsp/1200512992`. Acesso em: 22 mai. 2024.

MARTINCIC, F.; SCHWIEBERT, L. Distributed event detection in sensor networks. *In*: PROCEEDINGS of the International Conference on Systems and Networks Communication. [*S.l.*: *s.n.*], 2006. P. 43–48.

METLINK. **Microclimate**. 2024. Disponível em: https://www.metlink.org/microclimate. Acesso em: 11 mai. 2024.

PALPANAS, T.; PAPADOPOULOS, D.; KALOGERAKI, V.; GUNOPULOS, D. Online amnesic approximation of streaming time series. *In*: PROCEEDINGS 2003 IEEE International Conference on Data Engineering. [*S.l.*: *s.n.*], 2003. P. 338–349.

PERRIG, A.; STANKOVIC, J.; WAGNER, D. Security in wireless sensor networks. **Communications of the ACM**, v. 47, n. 6, p. 53–57, 2004.

RAJASEGARAR, S.; LECKIE, C.; PALANISWAMI, M. Elliptical anomalies in wireless sensor networks. *In*: PROCEEDINGS of the IEEE Conference on Local Computer Networks. [*S.l.*: *s.n.*], 2007. P. 548–555.

RAJASEGARAR, S.; LECKIE, C.; PALANISWAMI, M.; BEZDEK, J. C. Anomaly detection and localization in energy constrained wireless sensor networks. *In*: PROCEEDINGS of the 4th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing. [*S.l.*: *s.n.*], 2006. P. 19–24.

ROUSSEEUW, P. J.; HUBERT, M. Robust statistics for outlier detection. **WIREs Data Mining and Knowledge Discovery**, v. 1, p. 73–79, 2011. Disponível em: https://doi.org/10.1002/widm.2. Acesso em: 11 abr. 2024.

SCHÖLKOPF, B.; PLATT, J. C.; SHAWE-TAYLOR, J.; SMOLA, A. J.; WILLIAMSON, R. C. Estimating the Support of a High-Dimensional Distribution. **Neural Computation**, v. 13, p. 1443–1471, 2001. Disponível em: https://doi.org/10.1162/089976601750264965. Acesso em: 10 abr. 2024.

ZHANG, Yang; ZHANG, Yu; MERATNIA, Nirvana; HAVINGA, Paul J.M. Outlier Detection Techniques for Wireless Sensor Networks: A Survey. **IEEE Communications Surveys and Tutorials**, 2010. DOI: 10.1109/surv.2010.021510.00088.

APÊNDICE A – IMPLEMENTAÇÃO ALGORITMO KNN

```
# Função para identificar outliers usando K-NN
# Função para filtrar os dados por um sensor específico e por ano/mês
def filtrar_dados(data, sensor_name, year, month):
    filtered_data = data[(data['Station Name'] == sensor_name) &
                         (data['Measurement Timestamp'].dt.year == year) &
                         (data['Measurement Timestamp'].dt.month == month)]
    return filtered_data

# Função para identificar outliers usando KNN (LocalOutlierFactor)
# para julho de 2019, 2022 e 2023
def identificar_outliers_knn(data, columns, n_neighbors=50, contamination=0.05):
    lof = LocalOutlierFactor(n_neighbors=n_neighbors, contamination=contamination)

    for column in columns:
        # Preparar os dados para o LOF
        column_data = data[[column]].dropna()
        if len(column_data) < n_neighbors:
            continue

        # Ajustar e prever outliers
        data.loc[column_data.index, f'{column}_outlier_KNN'] =
        lof.fit_predict(column_data)
        # Identificar outliers (LOF retorna -1 para outliers)
        data.loc[data.index, f'{column}_outlier_KNN'] =
        data[f'{column}_outlier_KNN'] == -1

    return data

# Função para plotar dados e outliers
def plotar_dados(filtered_data, columns, sensor_name, year, month, output_dir):
    for column in columns:
        filtered_data['is_outlier'] = filtered_data[f'{column}_outlier_KNN']

        plt.figure(figsize=(12, 6))
        plt.scatter(filtered_data['Measurement Timestamp'], filtered_data[column],
        color='blue', label='Dados', alpha=0.6)
```

```
        # Destacar os outliers com um marcador diferente
        outliers = filtered_data[filtered_data['is_outlier']]
        plt.scatter(outliers['Measurement Timestamp'], outliers[column],
        color='red', label='Outliers', alpha=0.6)

        plt.xlabel('Timestamp')
        plt.ylabel(column)
        plt.title(f'{sensor_name} - {year}-{month:02d} - {column}')
        plt.legend()
        plt.grid(True)
        plot_filename = os.path.join(output_dir,
        f'{sensor_name}_{year}_{month:02d}_{column}.png')
        plt.savefig(plot_filename)
        plt.close()

    print(f'Plots salvos na pasta: {output_dir}')

# Lista de colunas relevantes
relevant_columns = ['Air Temperature', 'Wet Bulb Temperature', 'Humidity',
'Rain Intensity', 'Wind Speed', 'Barometric Pressure', 'Solar Radiation']
# Obter lista de nomes de sensores únicos
sensor_names = data['Station Name'].unique()

# Criar a pasta para salvar os plots
output_dir = 'plots_knn_mes'
os.makedirs(output_dir, exist_ok=True)
# Iterar sobre a lista de sensores e identificar outliers para cada sensor
for sensor_name in sensor_names:
    for year in [2019, 2022, 2023]:
        # Filtrar dados para o sensor específico e ano/mês
        filtered_data = filtrar_dados(data, sensor_name, year, 7)
        if not filtered_data.empty:
            # Identificar outliers nas colunas relevantes usando KNN
            filtered_data =
            identificar_outliers_knn(filtered_data.copy(), relevant_columns)
            plotar_dados(filtered_data, relevant_columns,
            sensor_name, year, 7, output_dir)
```

APÊNDICE B – IMPLEMENTAÇÃO ALGORITMO K-MEANS

```
# Função para identificar outliers usando K-means
def identificar_outliers_kmeans(data, columns, n_clusters=3):
    for column in columns:
        kmeans = KMeans(n_clusters=n_clusters)
        data[f'{column}_cluster'] = kmeans.fit_predict(data[[column]])

        # Calcular distâncias dos pontos aos seus centroides
        distances = np.min(kmeans.transform(data[[column]]), axis=1)

        # Identificar outliers como pontos com distâncias acima de um limiar
        threshold = np.percentile(distances, 95)
        # Definir limiar como o 95º percentil
        data[f'{column}_outlier_KMeans'] = distances > threshold
    return data

# Identificar outliers nas colunas relevantes usando K-Means
data = identificar_outliers_kmeans(data, relevant_columns)
for column in relevant_columns:
    # Separar outliers e não-outliers
    data['is_outlier'] = data[f'{column}_outlier_KMeans']

    # Calculando a média dos valores, excluindo os outliers
    mean_values = data[~data['is_outlier']].groupby('Measurement Timestamp')
    [column].mean().reset_index()

    # Plotando os dados
    plt.figure(figsize=(12, 6))

    # Plotando as médias
    plt.plot(mean_values['Measurement Timestamp'], mean_values[column],
    label=f'Mean {column}', color='blue')

    # Plotando os outliers
    outliers = data[data['is_outlier']]
    plt.scatter(outliers['Measurement Timestamp'], outliers[column],
    color='red', label='Outliers', alpha=0.6)
```

```
    # Configurações adicionais do plot
    plt.xlabel('Timestamp')
    plt.ylabel(column)
    plt.title(f'Mean {column} and Outliers (K-Means)')
    plt.legend()
    plt.grid(True)

    # Salvando o plot
    plot_filename = os.path.join(output_dir, f'{column}_kmeans_plot.png')
    plt.savefig(plot_filename)
    plt.close()

print(f'Plots salvos na pasta: {output_dir}')
```

APÊNDICE C – IMPLEMENTAÇÃO ALGORITMO DBSCAN

```
# Função para filtrar os dados por um sensor específico e por ano/mês
def filtrar_dados(data, sensor_name, year, month):
    filtered_data = data[(data['Station Name'] == sensor_name) &
                         (data['Measurement Timestamp'].dt.year == year) &
                         (data['Measurement Timestamp'].dt.month == month)]
    return filtered_data

# Função para identificar outliers usando DBSCAN para julho de 2019, 2022 e 2023
def identificar_outliers_dbscan(data, columns, eps=0.5, min_samples=5):
    for column in columns:
        # Preparar os dados para o DBSCAN
        column_data = data[[column]].dropna()
        if len(column_data) < min_samples:
            continue

        # Ajustar DBSCAN
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        data.loc[column_data.index, f'{column}_cluster'] =
        dbscan.fit_predict(column_data)

        # Identificar outliers (DBSCAN retorna -1 para outliers)
        data.loc[column_data.index, f'{column}_outlier_DBSCAN'] =
        data[f'{column}_cluster'] == -1

    return data

# Função para plotar clusters destacando os outliers
def plotar_clusters(filtered_data, columns, sensor_name, year, month, output_dir):
    for column in columns:
        plt.figure(figsize=(12, 6))
        unique_clusters = filtered_data[f'{column}_cluster'].unique()

        for cluster in unique_clusters:
            cluster_data = filtered_data[filtered_data[f'{column}_cluster']
            == cluster]
            plt.scatter(cluster_data['Measurement Timestamp'],
            cluster_data[column],
```

```
            label=f'Cluster {cluster}', alpha=0.6)

        # Destacar os outliers com um marcador diferente
        outliers = filtered_data[filtered_data[f'{column}_outlier_DBSCAN']]
        plt.scatter(outliers['Measurement Timestamp'], outliers[column],
        color='red',
        label='Outliers', alpha=0.6, marker='x')

        plt.xlabel('Timestamp')
        plt.ylabel(column)
        plt.title(f'{sensor_name} - {year}-{month:02d} - {column} - Clusters')
        plt.legend()
        plt.grid(True)
        plot_filename = os.path.join(output_dir,
        f'{sensor_name}_{year}_{month:02d}_{column}_clusters.png')
        plt.savefig(plot_filename)
        plt.close()

    print(f'Plots de clusters salvos na pasta: {output_dir}')

# Função para plotar dados e outliers
def plotar_outliers(filtered_data, columns, sensor_name, year, month, output_dir):
    for column in columns:
        filtered_data['is_outlier'] = filtered_data[f'{column}_outlier_DBSCAN']

        plt.figure(figsize=(12, 6))
        plt.scatter(filtered_data['Measurement Timestamp'], filtered_data[column],
        color='blue', label='Data', alpha=0.6)

        # Destacar os outliers com um marcador diferente
        outliers = filtered_data[filtered_data['is_outlier']]
        plt.scatter(outliers['Measurement Timestamp'], outliers[column],
        color='red',
        label='Outliers', alpha=0.6, marker='x')

        plt.xlabel('Timestamp')
        plt.ylabel(column)
        plt.title(f'{sensor_name} - {year}-{month:02d} - {column} - Outliers')
        plt.legend()
```

```
        plt.grid(True)
        plot_filename = os.path.join(output_dir,
        f'{sensor_name}_{year}_{month:02d}_{column}.png')
        plt.savefig(plot_filename)
        plt.close()

    print(f'Plots de outliers salvos na pasta: {output_dir}')

# Lista de colunas relevantes
relevant_columns = ['Air Temperature', 'Wet Bulb Temperature',
'Humidity', 'Rain Intensity',
'Wind Speed', 'Barometric Pressure', 'Solar Radiation']

# Obter lista de nomes de sensores únicos
sensor_names = data['Station Name'].unique()

# Criar as pastas para salvar os plots
output_dir_clusters = 'plots_dbscan_clusters'
output_dir_outliers = 'plots_dbscan_mes'
os.makedirs(output_dir_clusters, exist_ok=True)
os.makedirs(output_dir_outliers, exist_ok=True)

# Iterar sobre a lista de sensores e identificar outliers para cada sensor
for sensor_name in sensor_names:
    for year in [2019, 2022, 2023]:
        # Filtrar dados para o sensor específico e ano/mês
        filtered_data = filtrar_dados(data, sensor_name, year, 7)
        if not filtered_data.empty:
            # Identificar outliers nas colunas relevantes usando DBSCAN
            filtered_data = identificar_outliers_dbscan(filtered_data.copy(),
            relevant_columns)
            plotar_clusters(filtered_data, relevant_columns, sensor_name, year,
            7, output_dir_clusters)
            plotar_outliers(filtered_data, relevant_columns, sensor_name, year,
            7, output_dir_outliers)
```

APÊNDICE D – IMPLEMENTAÇÃO ALGORITMO SVM

```
# Função para filtrar os dados por um sensor específico e por ano/mês
def filtrar_dados(data, sensor_name, year, month):
    filtered_data = data[(data['Station Name'] == sensor_name) &
                         (data['Measurement Timestamp'].dt.year == year) &
                         (data['Measurement Timestamp'].dt.month == month)]
    return filtered_data

# Função para normalizar os dados
def normalizar_dados(data, columns):
    scaler = StandardScaler()
    data[columns] = scaler.fit_transform(data[columns])
    return data

# Função para encontrar os melhores
parâmetros de One-Class SVM usando GridSearchCV
def encontrar_melhores_parametros(column_data):
    param_grid = {
        'nu': [0.01, 0.05, 0.1, 0.2],
        'gamma': ['scale', 'auto', 0.1, 0.5, 1]
    }
    grid_search = GridSearchCV(OneClassSVM(kernel='rbf'),
    param_grid, cv=5, scoring='accuracy')
    grid_search.fit(column_data, np.ones(len(column_data)))
    return grid_search.best_params_

# Função para identificar outliers usando One-Class SVM com GridSearchCV
def identificar_outliers_svm(data, columns):
    for column in columns:
        # Preparar os dados para o One-Class SVM
        column_data = data[[column]].dropna()
        if len(column_data) < 2:
            continue

        # Encontrar os melhores parâmetros
        melhores_parametros = encontrar_melhores_parametros(column_data)
        nu = melhores_parametros['nu']
        gamma = melhores_parametros['gamma']
```

```
        # Imprimir os parâmetros escolhidos
        print(f'Parâmetros escolhidos para {column}
        - nu: {nu}, gamma: {gamma}')

        # Ajustar One-Class SVM com os melhores parâmetros
        oc_svm = OneClassSVM(nu=nu, kernel='rbf', gamma=gamma)
        data.loc[column_data.index,
        f'{column}_outlier_SVM'] = oc_svm.fit_predict(column_data)

        # Identificar outliers (One-Class SVM retorna
        -1 para outliers e 1 para inliers)
        data.loc[column_data.index,
        f'{column}_outlier_SVM'] =
        data[f'{column}_outlier_SVM'] == -1

    return data

# Função para plotar dados e outliers
def plotar_outliers(filtered_data, columns, sensor_name,
year, month, output_dir):
    for column in columns:
        filtered_data['is_outlier'] = filtered_data[f'{column}_outlier_SVM']

        plt.figure(figsize=(12, 6))
        plt.scatter(filtered_data['Measurement Timestamp'], filtered_data[column],

        # Destacar os outliers com um marcador diferente
        outliers = filtered_data[filtered_data['is_outlier']]
        plt.scatter(outliers['Measurement Timestamp'], outliers[column],
        color='red',
        label='Outliers', alpha=0.6)

        plt.xlabel('Timestamp')
        plt.ylabel(column)
        plt.title(f'{sensor_name} -
        {year}-{month:02d} - {column} - Outliers')
        plt.legend()
        plt.grid(True)
```

```
        plot_filename = os.path.join(output_dir, f'{sensor_name}_
        {year}_{month:02d}_{column}.png')
        plt.savefig(plot_filename)
        plt.close()

    print(f'Plots de outliers salvos na pasta: {output_dir}')

# Lista de colunas relevantes
relevant_columns = ['Air Temperature',
'Wet Bulb Temperature', 'Humidity', 'Rain Intensity',
'Wind Speed', 'Barometric Pressure', 'Solar Radiation']

# Obter lista de nomes de sensores únicos
sensor_names = data['Station Name'].unique()

# Criar a pasta para salvar os plots
output_dir_outliers = 'plots_svm_mes'
os.makedirs(output_dir_outliers, exist_ok=True)

# Iterar sobre a lista de sensores e identificar outliers para cada sensor
for sensor_name in sensor_names:
    for year in [2019, 2022, 2023]:
        # Filtrar dados para o sensor específico e ano/mês
        filtered_data = filtrar_dados(data, sensor_name, year, 7)
        if not filtered_data.empty:
            # Normalizar os dados
            filtered_data =
            normalizar_dados(filtered_data.copy(),
            relevant_columns)

            # Identificar outliers nas colunas relevantes usando One-Class SVM
            filtered_data = identificar_outliers_svm
            (filtered_data.copy(), relevant_columns)

            plotar_outliers(filtered_data,
            relevant_columns, sensor_name, year, 7, output_dir_outliers)
```

APÊNDICE E – IMPLEMENTAÇÃO ALGORITMO RNR

```
# Função para filtrar os dados por um sensor específico e por ano/mês
def filtrar_dados(data, sensor_name, year, month):
    filtered_data = data[(data['Station Name'] == sensor_name) &
                         (data['Measurement Timestamp'].dt.year == year) &
                         (data['Measurement Timestamp'].dt.month == month)]
    return filtered_data

# Função para normalizar os dados
def normalizar_dados(data, columns):
    scaler = StandardScaler()
    data[columns] = scaler.fit_transform(data[columns])
    return data

# Função para construir o Autoencoder
def construir_autoencoder(dimension_input):
    input_layer = Input(shape=(dimension_input,))
    encoded = Dense(32, activation='relu')(input_layer)
    encoded = Dense(16, activation='relu')(encoded)
    encoded = Dense(8, activation='relu')(encoded)

    decoded = Dense(16, activation='relu')(encoded)
    decoded = Dense(32, activation='relu')(decoded)
    decoded = Dense(dimension_input, activation='sigmoid')(decoded)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder

# Função para identificar outliers usando Autoencoder
def identificar_outliers_autoencoder(data, columns):
    for column in columns:
        # Preparar os dados para o Autoencoder
        column_data = data[[column]].dropna().values
        if len(column_data) < 10:
            continue

        # Normalizar os dados
```

```
        scaler = StandardScaler()
        column_data = scaler.fit_transform(column_data)

        # Construir o Autoencoder
        autoencoder = construir_autoencoder(column_data.shape[1])

        # Treinar o Autoencoder
        autoencoder.fit(column_data, column_data,
        epochs=50, batch_size=32, shuffle=True,
        validation_split=0.1, verbose=0)

        # Reconstruir os dados e calcular o erro de reconstrução
        reconstruido = autoencoder.predict(column_data)
        erros = np.mean(np.square(reconstruido - column_data), axis=1)

        # Definir um limiar para o erro de reconstrução (usando percentil 95)
        limiar = np.percentile(erros, 95)
        data[f'{column}_outlier_AE'] = False
        data.loc[data.index[np.where
        (erros > limiar)],
        f'{column}_outlier_AE'] = True

    return data

# Função para plotar dados e outliers
def plotar_outliers(filtered_data, columns, sensor_name,
year, month, output_dir):
    for column in columns:
        filtered_data['is_outlier'] = filtered_data[f'{column}_outlier_AE']

        plt.figure(figsize=(12, 6))
        plt.scatter(filtered_data['Measurement Timestamp'],

        filtered_data[column],
        color='blue', label='Data', alpha=0.6)

        # Destacar os outliers com um marcador diferente
        outliers = filtered_data[filtered_data['is_outlier']]
        plt.scatter(outliers['Measurement Timestamp'], outliers[column],
```

```
            color='red', label='Outliers', alpha=0.6)

            plt.xlabel('Timestamp')
            plt.ylabel(column)
            plt.title(f'{sensor_name} - {year}-{month:02d} - {column} - Outliers')
            plt.legend()
            plt.grid(True)
            plot_filename = os.path.join(output_dir, f'{sensor_name}_{year}_
            {month:02d}_{column}.png')
            plt.savefig(plot_filename)
            plt.close()

        print(f'Plots de outliers salvos na pasta: {output_dir}')

# Lista de colunas relevantes
relevant_columns = ['Air Temperature',
'Wet Bulb Temperature', 'Humidity',
'Rain Intensity', 'Wind Speed',
'Barometric Pressure', 'Solar Radiation']

# Obter lista de nomes de sensores únicos
sensor_names = data['Station Name'].unique()

# Criar a pasta para salvar os plots
output_dir_outliers = 'plots_autoencoder_mes'
os.makedirs(output_dir_outliers, exist_ok=True)

# Iterar sobre a lista de sensores e identificar outliers para cada sensor
for sensor_name in sensor_names:
    for year in [2019, 2022, 2023]:
        # Filtrar dados para o sensor específico e ano/mês
        filtered_data = filtrar_dados(data, sensor_name, year, 7)
        if not filtered_data.empty:
            # Normalizar os dados
            filtered_data =
            normalizar_dados(filtered_data.copy(),
            relevant_columns)

            # Identificar outliers nas colunas relevantes usando Autoencoder
```

```
filtered_data = identificar_outliers_autoencoder
(filtered_data.copy(),
relevant_columns)
plotar_outliers(filtered_data,
relevant_columns, sensor_name, year,
7, output_dir_outliers)
```